On autonomous target tracking for UAVs

Panagiotis Theodorakopoulos

Thèse

préparée au

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du

Doctorat de l'Université de Toulouse

Spécialité : Automatique

par

Panagiotis THEODORAKOPOULOS

On autonomous target tracking for UAVs

Soutenue le 4 mai 2009

devant le jury composé de :

Anibal OLLERO	Rapporteur
Claude PEGARD	Rapporteur
Rachid ALAMI	Examinateur
Pascal BRISSET	Examinateur
Jean Louis CALVET	Examinateur
Simon LACROIX	Directeur de thèse

LAAS-CNRS
7, Avenue du Colonel Roche
31077 Toulouse Cedex 4

Acknowledgements

First of all, I would like to thank the Greek State Scholarship Foundation (I.K.Y.) for offering me the chance to prepare this Ph.D. in France, by financing this project.

Secondly, I would like to express my gratitude to my supervisor, Simon Lacroix, whose great sense of humor, expertise, patience, warm leadership skills, and well-targeted ideas helped me get through this long term project. Simon, thank you a lot, I've learned a lot from you and if I started this Ph.D. all over again, I would still do it with you.

Thirdly, I would like to express my gratitude to Raja Chatila and Rachid Alami for accepting me in the robotics group, to Jean-Louis Calvet for participating at the defense jury, to Anibal Ollero and Claude Pegard for reviewing the thesis manuscript and finally to Pascal Brisset for participating in the jury as well as to the whole ENAC team for producing such a useful tool as the Paparazzi autopilot/simulator.

Fourthly, I would also like to thank Bertrand Vandeportaele for his valuable support during the flights, his technical advices and mostly for letting me know what it means to be a real engineer - I hope that you will invite me to see that house of yours when it is completed.

A big 'thank you' to Dimitris Fragkoulis for his support, good times spent around our common washing machine and the laughs we had when we had to drive each other to the airport - I just learned that we could have used one of the airport parking spots instead.

I also would like to thank Gautier Hattenberger for his help not only in the early phases of this thesis but also for his support as a member of the Paparazzi autopilot team, as well as Michel Courdesses for offering some good advises during the early phases of this thesis.

I would like to say a glorious thank you to all the people (and plants) that I've spent some time with in room B67. Viviane Cadenat for her jokes and good times, Tereza Vidal for her big smile coming from the other side of the desk, Joan Sola for his engaging ideas, talks and proofreading help, Louis F.Marin for the funny times and proofreading corrections, Norman Juchler for throwing balls of paper at me, eating all my yogurts and organizing cool parties. Finally, I should include in this list, the strange cactus named 'Finis-ta-these' that kept the atmosphere in the room clean and the spirits high.

A warm thank you to Jerome Manhes, Sara Fleury and the legendary Matthieu Herrb for helping our robots remain functional. I would like to give a special thank you to Anthony Mallet for sharing some talks with me in the Japanese garden as well as answering my programming questions.

Special thank you to: Sébastien Dalibard and Manish Sreenivasa for their corrections and engaging talks, Mokhtar Gharbi and Xavier Broquère for their help and funny talks, the great Yi Li for every lunch we had together and for helping me get out of the building the days that I left my badge home, Alireza Nakhaei Sarvedani for the laughing moments and Mathieu Poirier for all the drinks and the talks we had together about women. It would be impossible to leave out Sophie Achte, Camille Cazeuneve and Natacha Ouwanssi as they supported me in all my strange administration demands.

I would like to thank more than anything my parents Aristotelis and Katerina Theodorakopoulos for having worked hard during their lives so that their children can have the chance to think and enjoy an excellent quality of life. To my sister for finishing her studies and making me jealous so that I had to finish my Ph.D. too, to my grand mother for teaching me what courage means and to my uncle Thanassis for learning me how to think big.

En stor takk til Ida for å lære meg hvordan kjærlighet føles, for henne varmt smil og for

hennes støtte alle disse årene. Flyturen fortsetter wingman :)

Toulouse, May 2009

Contents

Glossary

AI Artificial Intelligence, 20
ANN Artificial Neural Networks, 43

FOV Field of View, 16

GIS Geographical Information Systems, 52
GNC Guidance, Navigation and Control, 20
GPS Global Positioning System, 125

LOS Line of Sight, 22

NFZ No fly zone, an area that the drone cannot fly over., 16
NMPC Non Linear Model Predictive Control, 39

PF Potential Field, 24
PN Proportional Navigation, 21

Shadow an area for the target where it is not seen by the UAV (shadows are caused by "visibility obstacles" or "occluding obstacles"), 17

Target The object moving on the ground to be tracked by the UAV, also called the "evader", 16

UAV Unmanned Aerial Vehicle, also called the pursuer, 15

Chapter 1

Introduction

1.1 Motivations

During the past ten years, there has been a significant increase in interest towards Unmanned Aerial Vehicles (UAVs) and this is probably because UAVs have come a long way since their early remote controlled days. Nowadays, UAVs can take off, fly and land almost without any human intervention. Following this trend, global production of UAVs has increased as well: while in 2004, the global production of unmanned aircraft systems (UAS) produced 477 different types of UAVs, some civilian but most military and by 2007 that figure had increased almost by 65% reaching the figure of 789 [International, 2007].

Some of these platforms are by all means impressive: they address many important issues such as flying capabilities, airspace integration, air worthiness and operational capacities. However, they seem to be missing one very important: autonomy. There is often a misconception that as a UAV has the capability to fly by itself during some parts of its mission, it is autonomous. The distinction should be made immediately: on the one hand, automatic flight offers a system the capability to perform actions according to a set of predefined rules in a highly predictable environment, a classical example being the automatic pilot. Whereas on the other hand, true autonomy offers a drone the capability to act without any supervision in an environment that changes and that can face a wide range of conditions. An autonomous system should provide an interface with a very high abstraction level, where the user could command the aircraft to "Explore(Area)" or "GoTo(City)" and the system could be left completely unsupervised to perform these tasks. In order to get an idea on the level of autonomy that modern UAVs provide, we should note that for most aerial platforms produced today, the plan following a communication loss is to command the aircraft to go to some deserted area and perform circles until communication is re-established or until all fuel are consumed and the aircraft hard lands.

UAVs have slowly evolved from piloted aircraft and no matter how fast they can fly or maneuver, they still bear a lot of similarities with their piloted counterparts: there is still somewhere a pilot taking the decisions and pulling the strings and the fact that he is not on the aircraft does not change much. On the other hand, robotics research has advanced significantly the past years and it can now offer many elaborate solutions in artificial vision, path planning, decision making, obstacle avoidance and environment modeling – all these work aiming at endowing mobile machines with autonomy. One of the leading motivations of this thesis is to examine the technologies leading to UAV autonomy and attempt to expand

Figure 1.1: Aerosonde: First robotic aircraft to cross the North Atlantic.

some of them.

The second motivation comes from the nature of most UAV applications. They come in a widening range: airborne surveillance, military information gathering, military offensive actions, automated search and rescue, gas pipe line monitoring and automated forest fire surveillance. The common point to most of these applications is that they are defined with respect to the ground environment: ground target tracking, be the target static, slowly moving or maneuvering at high speeds, is an essential task for UAVs. For such tasks, fixed wing UAVs have the advantage over helicopters to reach high speeds at low energy consumption, but their dynamics constrain the visibility of the ground target. This thesis considers this issue, and introduces various strategies to achieve ground visual target tracking, aiming at maximizing the target visibility.

1.2 Challenges and objectives

1.2.1 Problem characteristics

The overall objective of this thesis is to provide methods to endow a drone to autonomously track a moving ground target, under the following conditions:

- UAV dynamics: we consider fixed wing UAVs. This type of aircraft has motion limitations and dynamic constraints, and behaves as a non-holonomic vehicle. A fixed wing aircraft cannot stop and change direction as easily as a ground target can do and this raises difficulties, leading to the UAV being out maneuvered by the target. Also, we consider that the UAV motions are constrained by the presence of no fly zones.

- Presence of obstacles: UAVs are unlikely to be allowed to fly over all areas and in the case of a low flying micro drone this assumption is more than often true. One should consider that there exist areas over which the drone cannot venture either because they are tagged as no-fly-zones (NFZs), or because there are buildings to avoid. In any of the above cases one should find ways to pilot the drone in such a manner that it will maintain visibility without entering in the NFZ.

- Field of view restrictions: in many cases, UAVs and especially micro-drones do not have a full Field of View (FOV) leading to a greater complexity of the problem. We will examine the consequences of field of view restrictions for the UAV (panoramic full field of view with respect to limited field of view).

- Target dynamics and behavior: The target may be either moving on an open field or on a road network, and also has dynamic constraints (the model used in this thesis is that of a car). It can be neutral or evasive: in the latter case, it can exploit the presence of obstacles, denoted as "shadows", to avoid being seen by the UAV, making the problem akin to a *hide and seek game*.

1.2.2 Thesis objective

Overall, there is a need for the UAV to anticipate the movements of the target in order to be "a step ahead". This brings the interest to the level of the *decision process*: a target that is evasive is going to actively attempt to hide behind obstacles and use the road network in a predictable way. In order to maximize its visibility time, the UAV(s) must be able to predict such situations and to react or plan before they even happen in order to maintain visibility.

The objectives of the thesis is to provide a decision framework that maximizes the target visibility under all these constraints. Additionally, the case of two chasing UAVs will also be considered.

1.3 Contributions and Thesis Outline

The main contributions of this thesis are:

- The presentation of a control based navigation method that permits a UAV to track a ground target, in the absence of obstacles and shadows

- The presentation of a predictive method that permits a UAV to track a target based on its position, while avoiding obstacles and areas that the target can use as hiding places. The method considers the case of an evasive target that moves either in an off road environment or within a road network, and is extended to the case of two cooperative drones.

- The presentation of a discrete game theoretic approach that permits a UAV to track a target based on its position among NFZs and shadowing obstacles, and the extension of this approach to a two UAV collaboration scheme.

1.3.1 Outline

The thesis is structured in six chapters:

- In the second chapter, we review the state of the art on problems similar to the one at hand.

- In the third chapter, we present the problem details, *i.e.* the numerical models and constraints that we use.

- The fourth chapter presents a method that maximizes visibility under the assumption of a neutral target and a limited visibility, using a reactive navigation approach.
- The fifth chapter introduces a predictive method that tracks non cooperative ground targets, in an environment with obstacles and shadowing obstacles.
- The sixth chapter tackles the same problem, but using an approach based on classical game theory.
- Finally, all implementation tools used during this thesis are briefly presented in annex A.

1.3.2 Implementation Grid

Most of the control based methods presented in the state of the art as well as the one presented in the fourth chapter are rather control based and thus reactive. The methods presented in the planning section of the state of the art as well as the approaches proposed in chapters 5 and 6 are more planning based. The figure 1.2 presents how we locate our contributions in two dimensional "problem complexity" / "solution abstraction" space:

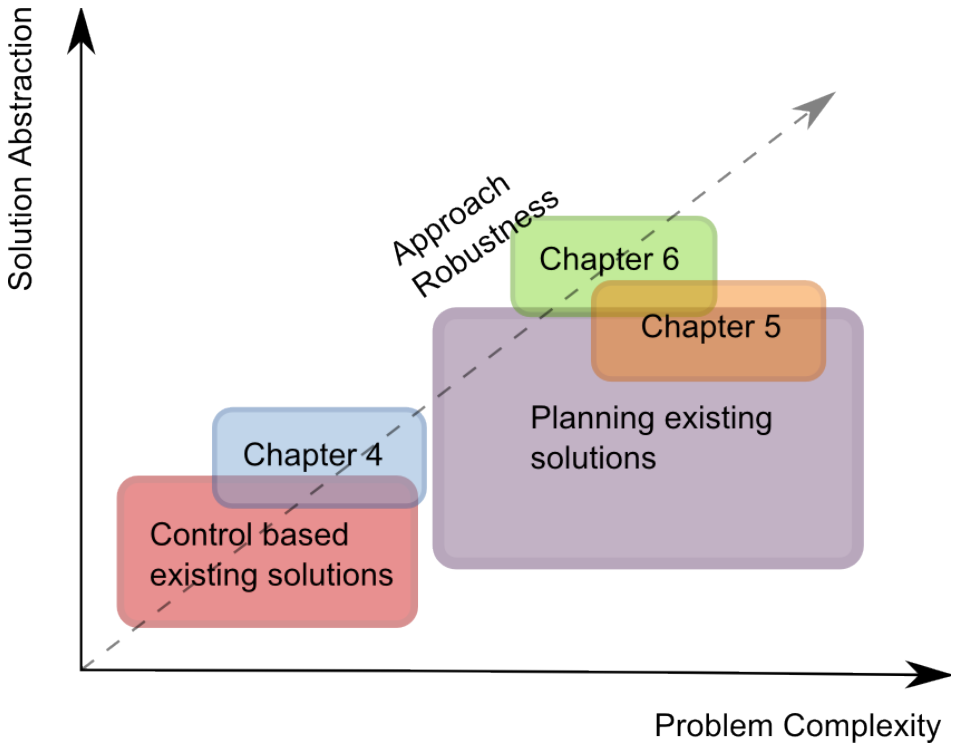

Figure 1.2: Location of our contributions with respect to the state of the art. Note that this figure does not measure the effectiveness of a method, it only illustrates the tackled problem complexity and the solution abstraction.

- "Problem complexity" qualifies the number of constraints to satisfy. For instance, a single UAV tracking a single passive target in an open environment is much less complex than 2 UAVs tracking an evasive target in an urban environment.

- "Solution abstraction" denotes the capacity of a proposed solution to be applied in a wide variety of applications. Low abstraction means a solution tailored to a very specific problem, *e.g.* a given UAV with a specific FOV, whereas a high abstraction solution mean that it can cope with numerous different problems, such as game theory, which can be applied in economic theories as well as in UAV ground tracking.

The approach proposed in chapter 4 consists of a double control loop that commands a trajectory that permits to the drone to maximize visibility. It tackles a problem a bit more constrained problem than the usual control-based approaches. It however is restricted to open environment, for UAVs with a limited FOV.

The approach presented in the fifth chapter is based on predicting and evaluating future UAV/target configurations to select the action that yields the most satisfying results. It is located a bit more left than the planning solutions on the figure because it has not been proven to work in very dense areas. It however does not require any preplanning, an adapts more easily to the problem as long as mission criteria remain the same.

Finally the game theory approach presented in chapter 6 relies on transforming the configuration to a game and then proceeds to find out the best game configurations by using classical game theory tools. It can easily be used for any kind of robotic applications. It is though more on the left than some other applications, because in complex environments, the Nash tabu search will not give always the best solution.

Chapter 2

State of the art

This chapter presents work that has been done in domains that are closely related to this thesis. One must remember that research on UAVs has involved experts from different and distinct domains. However, most early research programs on UAVs were traditionally attached to aeronautics university departments and as a result all experts involved at those programs where some-how related to Guidance, Navigation and Control fields. The result was that those researchers produced solutions heavily influenced by their research fields. As technology on UAVs was evolving, the cost and complexity of UAVs greatly diminished, and powerful and easy-to-use simulators can now be used by almost anyone having a personal computer. This makes it easier for researchers without any aeronautics background to enter the race: robotics and artificial intelligence researchers seized this opportunity. With their elaborate algorithms and with their iterative solutions, they not only provided new ways to tackle the UAV problems but they also produced new tools.

- The first body of research, is based on **Guidance, Navigation and Control**. It's main interest is how to change a trajectory in order to achieve maximum visibility of the target. Most of the scientific papers published in this domain also deal with the control aspects of the problem. However, very few deal with the presence of No-Fly zones or any other type of visibility obstructing obstacle. The typical scenario is the following: a target moves in an open field, while the UAV flies at a steady altitude and with steady speed, while trying to remain close to some ground target.

- The second body of research draws its forces from **planning based solutions** and moves a step further to autonomy.

- The third type of work presented in this chapter is **game theory**, that has been historically one of the most influential in the domain of decision making.

- This chapter would be incomplete without a presentation of **multi agent environments** and the two main type of interactions that can arise: **competitive interactions** in the sense that target and UAV try to outsmart each other and **cooperative interactions** in the sense that two or more UAVs may cooperate to maximize some common utility.

2.1 Guidance, Navigation and Control

The first Unmanned Aerial Vehicle to ever fly autonomously was the Hewitt-Sperry Automatic Aeroplane, also known as the "flying bomb" [Pearson,]: it performed its first flight on 12 September 1916. Its control was achieved by the use of gyroscopes and signaled the start of the UAV era. For the following forty years, the research on UAVs remained essentially a research on control theory. We could define this period as the ***control and guidance problem*** phase of UAV research. The question that researchers were seeking to answer was the following: *How can one make an unmanned aircraft closely follow a predetermined heading, altitude and speed ?*

As electronics and control theory evolved fast, this problem soon became trivial and led to a new research phase for UAVs: *What kind of system should we use on an aircraft, in order to be able to get from point A to point B successfully and without any form of direct human control ?* This problem was known as the ***navigation problem*** and it was especially interesting when point B was moving.

2.1.1 Waypoints VS Continuous Navigation

The guidance, control and navigation methods can be divided into three different approaches:

- In the waypoint approach, every trajectory is first divided into a series of **waypoints** that the drone must attain. Each waypoint is reached, one at a time, before the next waypoint is fetched from the list. As a result, any desired trajectory must first be divided in a series of waypoints before it is fed back to the control level.
- On the other hand, in **continuous navigation** approaches, waypoints are not necessary at all. Instead, a continuous stream of commands is passed directly to the control level. Most *visual servoing* [Bourquardez and Chaumette, 2007] and *missile guidance* approaches fell under this category. As every guidance law is tuned to optimize some problem specific criterion, there doesn't exist any generic method for Continuous Navigation control and as such each problem has to be treated individually.
- There exists also a mixed third type of approaches: **mixed strategy**. This approach makes use of both characteristics and shows quite good results in terms of trajectory following.

2.1.2 Missile Guidance

There are many similarities between UAV guidance and missile guidance. There exist many missile guidance laws that can assure that, a missile starting from a position A can most times hit an evading target B, with a reasonable probability.

Missiles are relatively simple flying machines that fly at a maximal (and most of the times, not controllable) speed. Even if there exist missiles that change their speed as part of their strategy, in most cases the missile guidance problem involves one control variable only: a force α_m applied normally on the missile's vector V_m. When the missile reaches a critical distance from the target, it detonates and this is known as a **capture**.

Several methods exist that show good results in terms of capture capabilities.

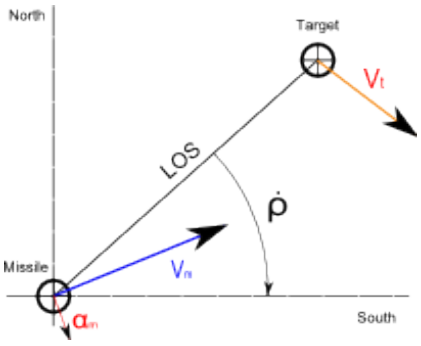

Figure 2.1: This is a 2D graph of Proportional Navigation applied on the lateral plan.

Proportional Navigation (PN) (Fig. 2.1) is one of the oldest and most classic method used by missiles in order to capture targets. The idea has been proved to work so well, that it remains still in use today, five decades after its first appearance in scientific papers.

PN is based on the fact that two vehicles are on a collision course when their Line-of-Sight (LOS) angle does not change. In order to achieve that, the normal angle α_m remains proportional to the rate of change of the LOS $\dot{\rho}$, using the following formula:

$$\alpha_m = N\dot{\rho}$$

N is a gain that permits to tune the behavior of the missile trajectory: a gain of $N = 1$ yields a curved trajectory, while a gain of $N \to \infty$ guides the missile straight to the position of the target with no curvature[1]

Pursuit Guidance (Fig. 2.2 (a)) is a very simple method that makes sure that the missile always flies towards the evading target. Even if this method can some times be impractical for closing the final distance and 'capturing' the target, it still remains an excellent method for approaching closer to the target.

Deviated pursuit Guidance (Fig. 2.2(b)) is quite similar to the pursuit Guidance with one significant difference. The missile angle leads the LOS angle with a fixed angle λ. This is like anticipating where the target will be and thus covering less distance than the target.

Three Point Guidance (Fig. 2.2(c)) makes sure that the missile will always lie somewhere between the target and the target tracker. It is known also as constant bearing guidance.

For a good introduction on these methods, the reader might want to look at the classical work of Siouris [G.M.Siouris, 2004]. For more details, in [M.Guelman, 1971] one can find a good study on Proportional Navigation in one of the oldest publications that exist on the subject. [D.Ghose, 1994] and [Yang and Yang, 1997] treat the problem of a maneuvering target and they show how proportional navigation achieves capturing evasive targets. Finally, in [U.S.Shukla and P.R.Mahapatra, 1990] an alternative method of PN is presented, where the applied force is not normal on the missile speed vector, but on the LOS vector instead.

One very interesting application of PN on a wheeled, ground robot is given in

[F.Belkhouche and B.Belkhouche, 2007], where the authors use a more relaxed version of the law:

[1]The curvature should not be confused with unwanted oscillations that may arise. A missile can move along a given curvature and still have fast oscillations.

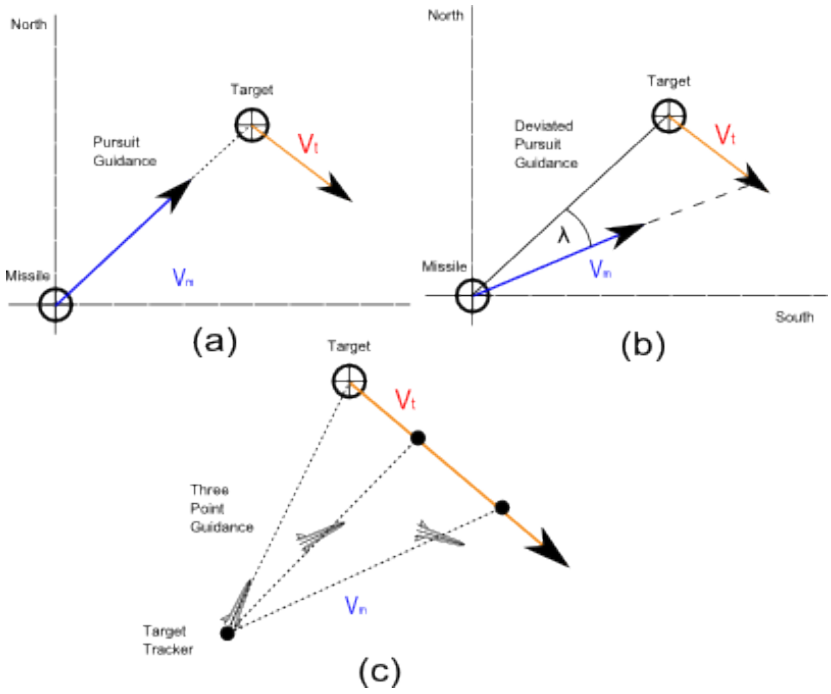

Figure 2.2: Different forms of missile guidance.

$$\theta_R = N\rho + \lambda$$

where θ_R is the robot heading with the reference frame, ρ is the LOS angle and λ is the lead angle that if different from zero can have the same effect as in deviated pursuit guidance. It should be noted that during the simulations the robot manages to find its way in the presence of obstacles.

2.1.3 Mixed strategy trajectory following

Nicolescu [Niculescu, 2001] proposed a lateral track law that manages to hold a UAV very close to its trajectory. This approach, behaves very well in straight lines. However, a method that could behave well both on straight lines and on curved trajectories was proposed by Park *et al* in [Park et al., 2004].

Park (Fig. 2.3) proposed a nonlinear guidance law that performs better than conventional Proportional Derivative controllers. This approach has been shown to be mathematically equivalent to a PN guidance law with a gain of $N = 2$.

The law of the guidance law is:

$$\alpha_{cmd} = 2\frac{V_d}{L_1}\sin\eta$$

where α_{cmd} is the lateral acceleration command, V_d is the drone speed, L_1 is the distance between the reference point and the drone and finally η is the angle between the drone speed vector and the UAV / reference point LOS. The method has been used for obstacle avoidance as we will see later in [Ducard et al., 2007].

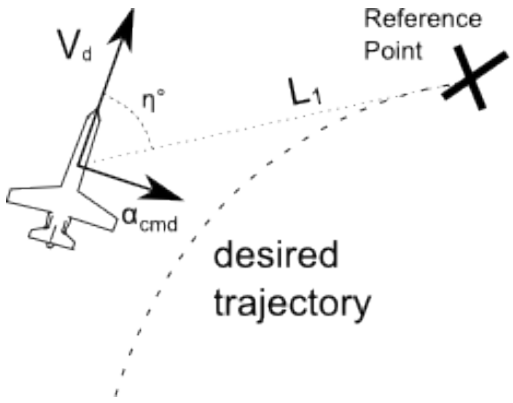

Figure 2.3: Diagram of Shangyuk Park's proposed guidance law.

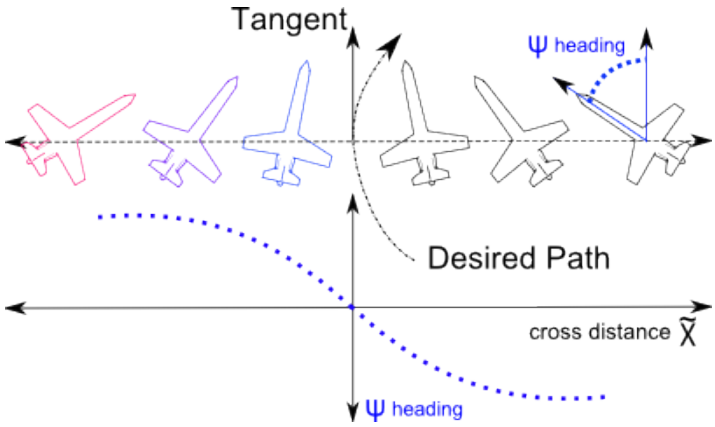

Figure 2.4: Diagram of the 'good helmsman's' Guidance.

2.1.4 Waypoint Guidance

One of the most classic methods in the UAV Guidance literature, comes from waypoint tracking control of ships [Pettersen and Lefeber, 2001]. The idea behind this principle is to model the behavior of a 'good helmsman' (Fig. 2.4), the person that has the task to steer a ship, a submarine or any type of maritime vessel. The idea was presented to the UAV community by Rysdyk in [Rysdyk, 2003], who fine tuned it for the aircraft case.

2.1.5 Potential Fields

These methods use repulsion from undesired configurations and attraction towards goals in order to compute a path. A useful method to generate motions is the *Potential Field (PF)* (Fig. 2.5) [Latombe, 1991]. The primary advantage is that PF approaches were developed for real time applications. The idea behind PFs is that every point, which has to be avoided is surrounded by a repulsion field while every desired goal point is surrounded by a attraction field. This way the robot will move around the obstacles that repulse it and move towards the points that attract it. Using this simple formulation the task of path planning becomes much easier: the robot always moves towards the area that attracts it the most and this can be done using any greedy local search method like *e.g.* steepest descent. A similar approach

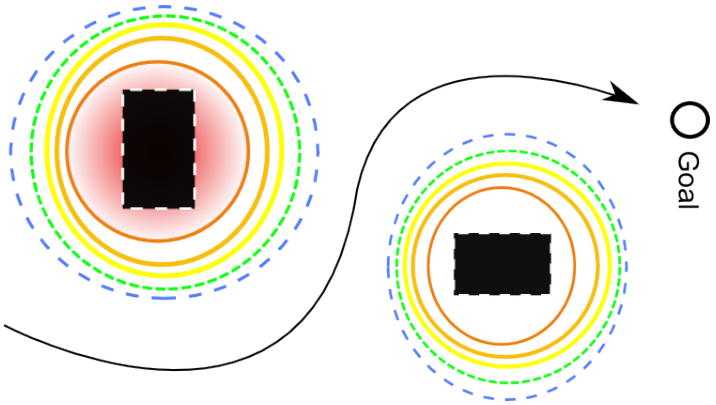

Figure 2.5: Trajectory of a robot that avoids two obstacles while heading for some distant goal with using a Potential Field method.

uses probability density functions as evaluations functions [Dogan, 2003][Kim et al., 2008].

However, the biggest caveat of this method is that robots can be easily trapped into *local minimum* and remain there in the absence of an external aid [Koren and Borenstein, 1991]. A good mechanism to break cyclic behaviors is proposed in [Dogan, 2003]: Every time a change in UAV heading is detected a counter measures how much the aircraft has turned: If a full 360 degrees heading change is detected, the behavior is noted as cyclic and the aircraft enters an exiting pattern. Another solution proposed for overcoming cyclic behavior in potential field methods is the change of altitude [Helble and Cameron, 2007].

2.1.6 On Non-holonomic Aircrafts and Time Optimal Paths

Aeroplane dynamics are constrained by various laws, spanning from aerodynamics to structural engineering. However, the most influential law for target tracking are those of lateral movement, as they can help the selection of a UAV optimal path.

A robot whose configuration space has l dimensions is a non-holonomic robot, if it is constrained by k non-integrable constraints G_i with $i \in k$ and $(0 < k < l)$ of the form :

$$G_i(q, q') = 0$$

An aeroplane behaves exactly as a Non-holonomic vehicle when it is examined with respect to its lateral movement. A Non-holonomic vehicle is a vehicle whose current posture (position and orientation) depends on the path it took to get there.

The model that seems to correspond the most is that of the **Dubins car** as it is known in the Robotics community from the ground breaking work of L.E.Dubins [Dubins, 1957], published in 1957. [2]

[2]However, what until recently was not known is that, another researcher under the name of Pecsvaradi published a paper in 1972, covering the same aspects, arriving at the same results and focusing *solely* on aeroplanes. However, as he published in a journal that was read almost only by researchers from the Control and Guidance community, his work did not reach the Robotics community, until very recently. The two researchers arrived at the same results but did it using two different approaches: Pecsvaradi used the Pontryagin Maximum Principle (PMP) and found what guidance law is appropriate, when an aircraft wishes to get from one point to another, in a time optimal manner. On the other hand, Dubins achieved the same result but this time using Topology elements, without mentioning the application. For a good introduction on PMP , one can read [L.M.Hocking, 2001] and for an introduction on Topology, one can look at [Lipschutz, 1968].

The term **Dubins Aircraft** was coined in [Chitsaz and LaValle, 2007], where three distinct cases where examined all using the Pontryagin Maximum Principle (PMP): what are the time optimal paths in order to attain three different goals of low, medium and high altitude. This type of work has been applied for years in the Robotics community in order to find optimal paths for ground robots. Good examples can be seen in [Bui et al., 1994] and [Tang and Ozguner, 2005].

The model of the Dubins aeroplane is given by:

$$\dot{x} = v\cos\psi \tag{2.1}$$
$$\dot{y} = v\sin\psi \tag{2.2}$$
$$\psi = -(g/v)\tan\phi \tag{2.3}$$
$$\tag{2.4}$$

where x and y are the Cartesian coordinates of the aeroplane, ψ is the heading angle of the aeroplane, v is the speed of the aeroplane, ϕ is the bank angle and g is the gravity acceleration. It should be noted that the bank angle is bounded:

$$|\phi| \leq \phi_{max}$$

For any aeroplane (or robot) following the above equations, any given initial and final configuration can be attained using six types of paths (Fig. 2.6), constructed from three arcs only. This means for every UAV, moving in the absence of obstacles, only three type of commands are necessary to attain any point on the horizon under any type of heading:

- An arc of a counterclockwise, left circle (L) $\psi = -(g/v)\tan\phi_{max}$
- An arc of a clockwise, right circle (R) $\psi = (g/v)\tan\phi_{max}$
- A straight line segment (S) $\psi = 0$

2.1.7 UAV Trajectory Optimization for Target Tracking

Target tracking includes a series of methods that permit a UAV to bring in or to maintain the target within its camera frame. This is done by adjusting accordingly the UAV trajectories and camera postures in order for the drone maintains target visibility. The key aspects of these methods are to maintain target proximity and proper camera orientation.

Lee *et al* [Lee et al., 2003] propose the following strategy in order to track a ground target: the UAV enters in two different modes accordingly to the speed of the ground target. If the target moves too slow, then the UAV enters a *loitering mode* that looks like a *rose curved trajectory* (Fig. 2.8) and by this it remains close to the target. If however the target starts to move faster, it enters into *sinusoidal mode*. During this mode the UAV follows a sinusoid wave pattern trajectory, whose frequency and amplitude depend on the targets speed and thus staying close to the target. The target's position is supposed to be known already, while a full FOV visibility is assumed for the camera. The authors have proved the functionality of the algorithm in a real flight test.

Stolle *et al* [Stolle and Rysdyk, 2003] make use of the helmsman's guidance law in order to track a ground target. They examine different types of camera visibilities and provide

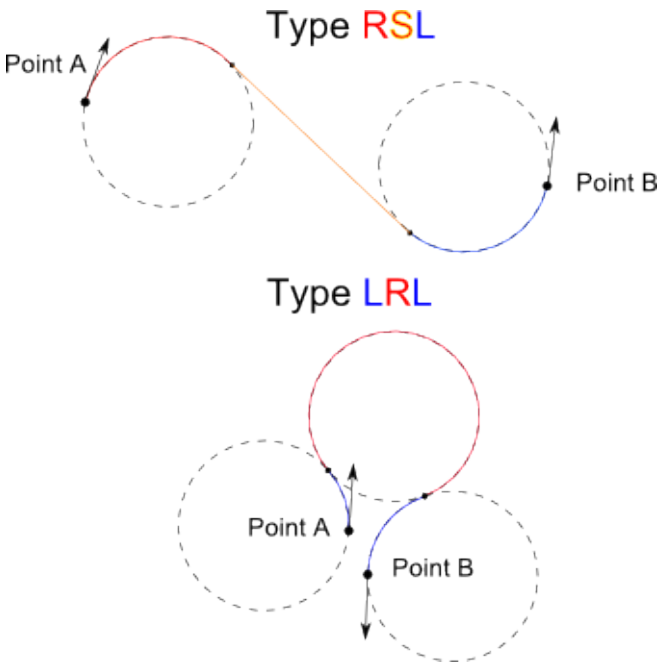

Figure 2.6: Two examples of optimal trajectories for a Dubins aeroplane. Note the different headings of the waypoints.

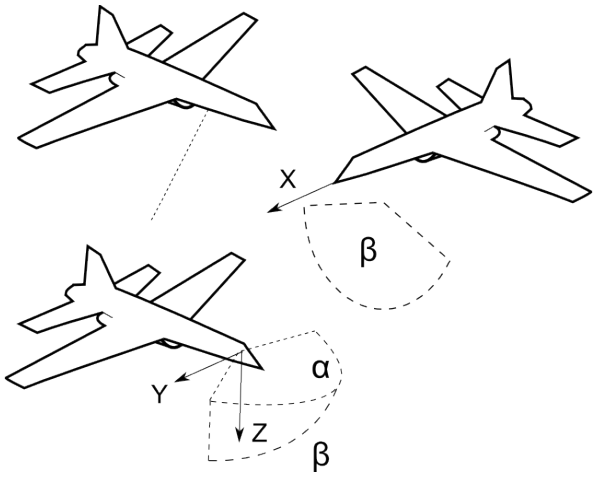

Figure 2.7: Different configurations of cameras. We can see cameras that are of static, roll only and pan-roll, depending of their capabilities to turn on certain axes. β is the roll (or tilt) axis and α is the pan axis

different type of trajectories. They show that for a full FOV camera, a *circular path* around the target will guarantee visibility. However, when the camera has limited camera angles (Fig. 2.7) and it is in the presence of wind, they propose to add a sideslip angle in order to maintain the target within the camera FOV without changing flying path. They remark the negative effects of the sun and the wind and they propose *segmented circular orbits* (Fig. 2.8) around the target of different radius in order to avoid the angles where visibility is deteriorated.

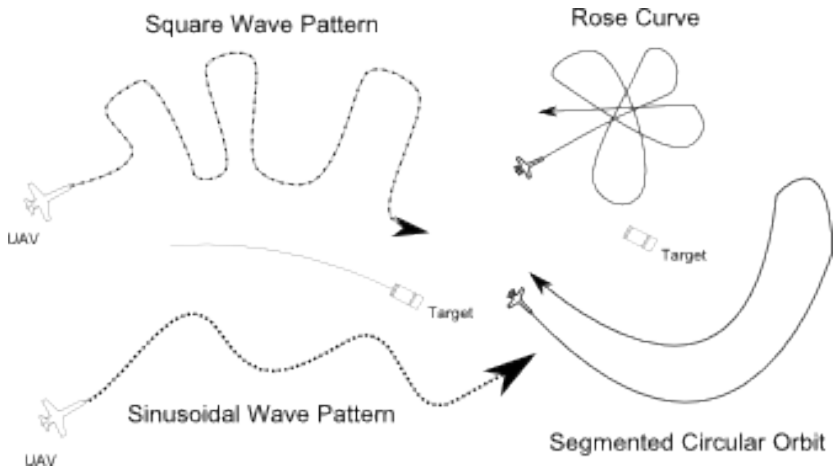

Figure 2.8: Different strategies for tracking a ground target.

Husby during his master thesis added a *square wave pattern* trajectory that achieves the same result as the sinusoidal method and demonstrated that the methods presented in [Stolle and Rysdyk, 2003] work equally well with moving targets.

The case of a roll only camera (Fig. 2.7) has been examined in [Thomasson, 1998] by Thomasson. He proved that for such a configuration only three type of trajectories are viable: *spirals, circles and ellipses*, all depending from the wind vector. He then provided the guidance laws required for a UAV to perform a search. The author of [Rysdyk, 2003] decided to work on a 'clock-angle' frame of reference, with each algorithm evaluating every bearing angle around the UAV. This time the camera is considered to be of pan and roll type (Fig. 2.7). The author looks for the trajectory that maintains the geometry between target and drone constant. He then arrives at results similar the roll only camera case. The idea of an ellipse trajectory that compensates wind presence has been examined in [Sun et al., 2008], where the authors described the pan, roll camera guidance law. Also in [Rafi et al., 2006] the authors propose a circular pattern around a target and offer the guidance algorithms to achieve that.

In [Quigley et al., 2005b] the authors proposed an attraction field generated by a super critical Hopf bifurcation in order to keep the UAV close to the target. A Hopf bifurcation is a phenomenon appearing to certain dynamical systems, causing a limit cycle attraction field. An interesting application of the Dubins aeroplane in order to follow a slower target (in this case a helicopter) has been presented in [Ryan and Hedrick, 2005] where the methods reminds us the work on sinusoidal guidance, seen earlier. In [Rathinam and Sengupta, 2004] the authors assume that a fast moving UAV will not have the time to scan the areas for threats. So they propose to delay the forward speed using circle trajectories in order to scan the area.

2.1.8 Synthesis of Guidance, Navigation and Control Bibliography

Missiles, drones and cars all behave as non-holonomic vehicles and as such any law that can guide successfully one type of engine can guide another one too – when we examine them in the lateral plane. Missile guidance laws provide a good set of tools that can permit a UAV to approach a target, a waypoint or a trajectory path. In essence this is what Sanghyuk guidance

achieves, as under some circumstances, it closely resembles to proportional navigation.

It was shown that for any UAV, three commands are enough to successfully plan a path. This can be of great use when graph search algorithms will be implemented.

In the domain of reactive path planning there exist numerous trajectory patterns that can achieve a good observability. However, these methods depend from the type of camera:

- For roll only cameras, Ellipses and Circles seem to be the trajectories that provide a good target visibility through the trajectory.
- For full FOV cameras, a circular path seems to track the target the best.
- For 180 degrees FOV cameras, a circular path may not be good enough in the presence of wind, as it can easily throw the target out of view. Two solutions are proposed: Either add a sideslip angle if that option is available on the drone or perform an ellipse based trajectory.

These trajectories are actively constructed with predefined patterns by examining the Cartesian coordinates of the drone. The knowledge of when to switch from one type to another is assumed to derive either from the position of the drone or from measuring its speed. Very few of these cases consider an obstacle that obstructs either the UAV path or the target image. As a result they can function only under special conditions.

Furthermore there doesn't exist a comparison that can show us which of these methods presents the best target tracking score under different target movements.

Another point that varies significantly is the way the target's position is acquired. In most cases the position is known via some external source and very few actually test the method with a real flight camera feedback.

Overall, there seem to exist several methods that address the visibility issues but there lacks a unification, a single method that can be used in all of the above combinations of camera types, target position input types and aeroplane types.

2.2 Planning based methods for UAVs

Some of the methods examined in the section are:

- The use of predictive methods in order to select the best future path.
- Research of all possible configurations in order to find the best path possible using search algorithms.
- Anticipation of target movement in order that the drone will position itself in such a way that it will be easier to track the target, even if the target tries to avoid capture. This capability falls under the category of adversarial searching and the applications arising are known as pursuit evasion games.

2.2.1 Path Finding for UAVs

Path or Motion Planning in Robotics, is the process where a machine produces a collision-free path from a point A to a goal point B. However, this section does not address only methods that pre-plan a path, but also expands to methods that simply search to find a near optimal path under several constraints as obstacle avoidance and target visibility (Fig. 2.9).

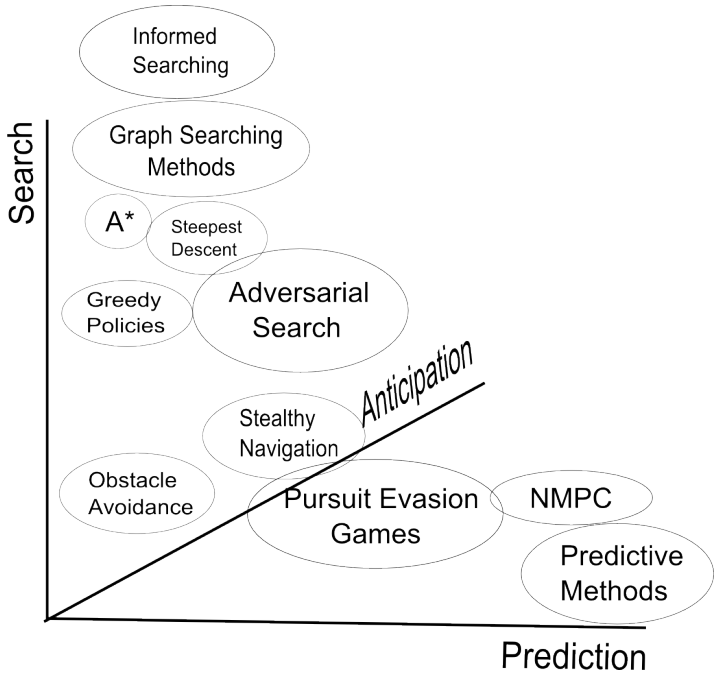

Figure 2.9: Summary of the tools and application domains presented in the Path Finding bibliography.

2.2.2 Search algorithms

Despite the real world fluctuations, in the AI world drones are considered to be flying in a 3-D space known as a *State Space*.

The problem as it is proposed by Path Planning is: *What kind of process should a machine use in order to find an optimal path starting from an initial state towards some desired goal.*

Most *search algorithms* used in the Robotics literature are mainly of *graph search* type. [Russell et al., 1995][Korf, 1985] The basic idea behind graph search algorithms is to use the **nodes** of a *search graph* to store information for every state space available. Each node also stores information regarding the *path cost* one would use to get to that node, based on problem related criteria. This way the question of path planning becomes a question of *finding the optimal path within the search tree.* After one node is examined and found that it is not the goal node, all or some of its descendant nodes are generated and examined in order to see, if any of them is the goal node. The total cost of each candidate path is the sum of all individual costs of every node participating in the path. Here, some complexity issues arise as the search tree expands (also known as the depth limit or diameter of the graph), so do the calculations necessary to complete the process. For an on board computer, this may bring up issues of memory usage and processor time allocation.

There exist two main categories of search algorithms:

- **Informed** methods use some form of heuristics in order to select which node will they open.
- **Uninformed** methods where no information exists concerning which node should the algorithm explore next. As a result nodes are opened one by one until an end goal is

reached.

Uninformed search algorithms

In this category we find the following algorithms:

- Uniform-cost search algorithm
- Breadth-first search algorithm
- Depth-first search algorithm
- Depth-first iterative deepening search algorithm

Informed search algorithms

- Branch and bound search algorithm
- Dijkstra's search algorithm.
- A* and all its descendants (IDA*, D* etc)
- Steepest Descent search algorithm

Example 2.1. ———————————————————————

Local search algorithm - Tabu search

The tabu search method is an optimization solution that uses local search methods. This specific search method overcomes local minimum by using a special memory structure known as the tabu list T. As the algorithm proceeds evaluating different solutions using some termination condition, it places in the tabu list all solutions that it has already examined. Once a solution is put in that list the algorithm can no longer reuse it, forcing thus the algorithm to search for new, alternative solutions. When the algorithm is applied to a graph this leads to constantly exploring new paths until a solution is found [Gendreau, 2003, Glover and Laguna, 1993].

———————————————————————

Example 2.2. ———————————————————————

Heuristics graph search - for Robots

A* algorithm *is a widespread search algorithms used in Robotics and AI. It is used to find the* ***least cost path*** *from an initial node to the desired end goal. It is a best first, informed search algorithm. While most algorithms will expand all nodes one by one until they arrive to the end goal, A* uses a shortcut in order to arrive to the end goal. This is achieved using an* ***evaluation function*** *$f(x)$ that helps select only the most promising nodes to expand. For this reason, the algorithm keeps in memory a whole front of nodes and keeps expanding only the best one at a time. Once a node has been examined and found not to be the end goal, it is*

removed from the front and its successors are placed in memory, restarting the whole process again.

The evaluation function $f(x)$ of every node consists of two parts:

- *$g(x)$ where the cost of the path from initial-node-to-the-present-node is stored.*
- *$h(x)$ a **Heuristic function** where the estimated-to-the-end-node cost is stored.*

In order for A to be optimal an **admissible heuristic** $h(x)$ is needed, something that can be achieved if the heuristic underestimates the actual cost left to reach the goal. In other words, this means that the heuristic must always be optimistic and a way to do that is by adopting **a relaxed version of the problem**. In the case of a robot that seeks to find the shortest path to a goal, the Cartesian distance between the node in question and the goal is admissible.*

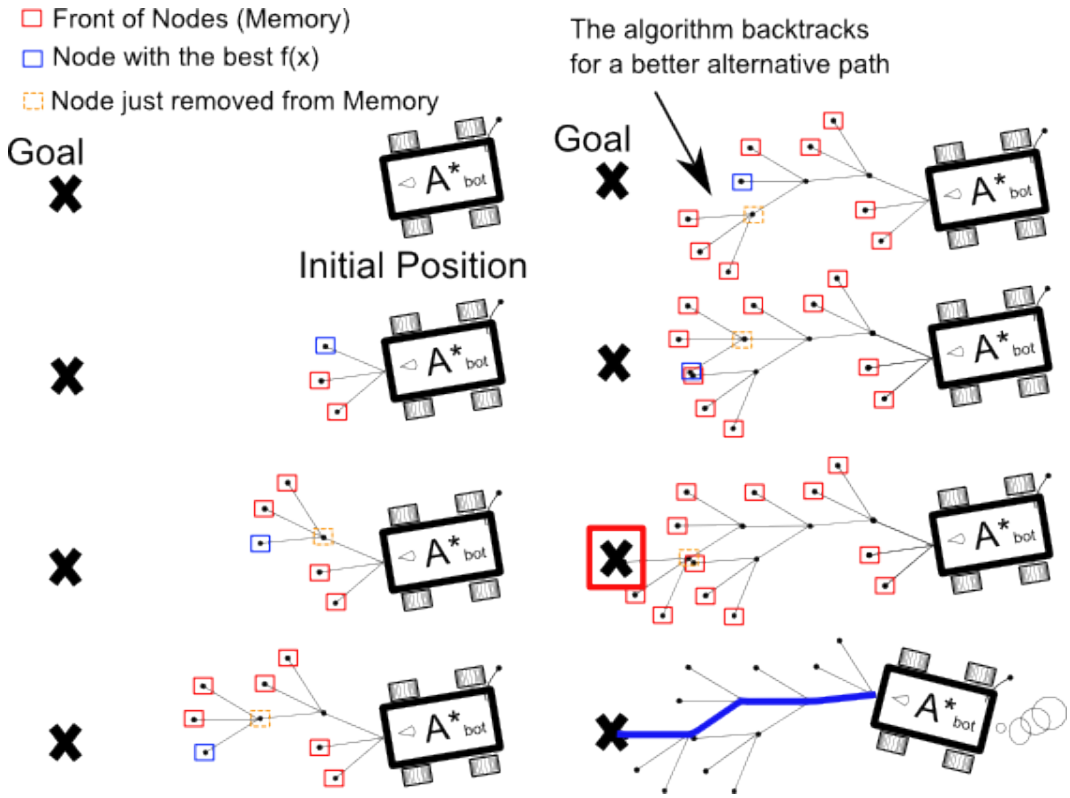

Figure 2.10: Diagram of a robot that uses A* for path planning.

Furthermore A is **complete**, meaning it will always find a solution if there exists one. The worst case for time and space complexity is $O(b^d)$ where b is branching factor or the maximum number of successors of any node and d is the depth of the goal in steps. Fig. 2.10 shows a robot using A* while trying to find the shortest path to some desired goal.*

2.2.3 Variants of Potential Fields

Another local method that avoids some of the local minimum problems is the vector field histogram (VFH) [Borenstein and Koren, 1991][Ulrich and Borenstein, 1998]

[Borenstein and Koren, 1990]. This approach constructs a *polar histogram* containing a value representing the *polar obstacle density* in every direction. As the method is bearing oriented as opposed to Cartesian position oriented, it can avoid some forms of local minimum. However, this can lead to dead-end situations and exhibit cyclic behaviors, where the robot moves from one obstacle to another.

2.2.4 Path Finding and Obstacle Avoidance

A classical approach to ground robot path planning is the construction of a *traversability map* [Singh et al., 2000] [Gennery, 1999] followed by a graph search method to find the best route to the desired goal.

A related method is the *arcs approach* [Lacroix et al., 2002] [Langer et al., 1994], where a series of arcs is generated and compared one to each other in order to select the one with the least cost. The robot then chooses steers along its way. The cost function is the sum of traversability values along its path. This type of search is called greedy as instead of searching a sequence of commands along the way, it selects the immediate best command. It is situated somewhere between control and planning.

A commonly used method is that of *behavior based algorithms* [Langer et al., 1994]

[Seraji and Howard, 2002] where every task in the path planner generates its own heading and speed setpoints. In the case of a conflict a behavior arbitrator fuses the different setpoints and finds the common decision.

The organization of path geometry around obstacles is also a quite important subject in robotics. A very common technique is the use of *Voronoi graphs* [Latombe, 1991]

[Barraquand et al., 1992] to find the routes between obstacles. Voronoi graphs are used to separate the neighbor areas from two or more obstacles at equal distances. In UAVs this has been applied in [Bortoff et al., 2000][Chandler et al., 2001]. This method permits to find a safe path among two or more obstacles. A very closely related method is the separation of the space with the Finite Element Method [Pimenta et al., 2005].

A method that searches a route among obstacles is the *Roadmap method* [Latombe, 1991] [Isler et al., 2005]. The method creates a graph consisting of lines connecting all edges of an object (or tangents leading to the object) including: the end goal, the obstacles positions and the robot position. It then proceeds at finding the optimal path at that graph. In [Isler et al., 2005] this is done in order to minimize time.

In [Rathinam and Sengupta, 2004] the authors use the Dijkstra's algorithm along with a *risk map* in order to navigate the drone through safe areas. Finally, a very good application of Park's lateral guidance law was proposed by Ducart *et al* in [Ducard et al., 2007]. The authors proposed the use of tangent lines starting from the drone and touching the security circle around the obstacle. The points of contact are then used as waypoints for the lateral guidance law and drive the UAV around the obstacle. This method can be very useful in the case of a drone that is moving and detecting obstacles in real time as it does not need beforehand trajectory calculation.

Another method that is lately gaining a lot of ground is that of *Rapidly-exploring Random Trees (RRTs)*. This method permits to perform a research in highly multidimensional

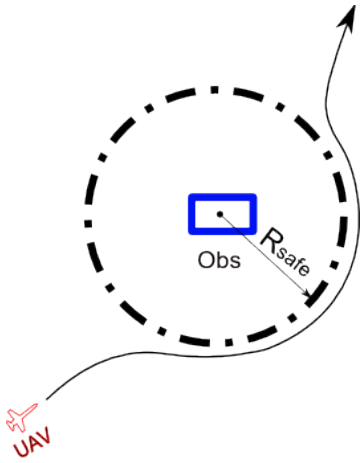

Figure 2.11: Trajectory of a robot that avoids an obstacles, while heading for some distant goal with the use the method proposed in [Ducard et al., 2007].

non convex areas and in drones is applied by exploring random arcs and then choosing the ones that are the closest to the end goal and not in contact with any obstacles. RRTs perform really well with path planning problems that involve obstacles and differential constraints. Work on RRTs can be found in [Tan et al., , LaValle, 1998, Bruce and Veloso, 2003, Kuffner Jr and LaValle, 2000].

2.2.5 Synthesis of planning methods

The basic tool presented in this section involve variants of potential field methods and graph searching tools. Robotics provides a set of tools that allow any type of robot to navigate from one point to another, while avoiding any obstacles that it may encounter. Overall, we can consider that the solutions are robust enough and with some slight changes they could be used for fixed-wing UAV path finding. However, one should not forget that fixed wing UAVs have two very important constraints: in contrast to ground robots and helicopter UAVs, they cannot stop. Furthermore, small UAVs may have computation limitations. The work that should be done in this domain would be to adapt somehow the above methods in order to produce a fixed-wing UAV path-finding algorithm that respects the above constraints.

2.3 Decision: game theory

This section deals with a basic analytic tool of decision theory. Decision theory is a framework that permits one to take the best decision under a set of rules and based on the information that the decision maker has at a given moment.

Game theory: introduction

Game theory [Fudenberg and Tirole, 1991, von Neumann and Morgenstern, 1953] is a branch of applied mathematics that tries to capture behaviors in strategic situations as the ones encountered in multi-agent environments. It deals with the decisions that an individual or a team has to take, when its success depends on the choices of others. It has applications that

span in various domains: social sciences, economics, biology, engineering and political science being some examples.

Example 2.3. ———————————————————————

Game theory: Quick Review

A game G is a mathematical object that describes the interaction of a group of players $i \in N$, where $N = 1, ..., n$ is the set of players. Each player can choose among a set A_i of actions $\sigma_i \in A_i$ (aka pure strategies) or among a set of mixed strategies $\Delta(A_i)$. Furthermore the player is considered to be rational and thus seeking to optimize a certain utility or payoff $u_i : A \to \Re$.

Games can be found in different categories depending on several parameters:

- *Cooperative vs non-cooperative games. In a cooperative game, players can form coalitions, as opposed to non-cooperative games, where each player is trying individually to optimize his own utility.*
- *Zero sum vs non-zero sum games. A zero sum game is a game where the sum of all payoffs of all players is equal to zero: $\sum_i^N u_i = 0$.*
- *Sequential vs simultaneous games. During simultaneous games, both players move simultaneously as opposed to sequential, where players alternate moves.*
- *Perfect information sequential games vs imperfect information sequential games. If all players know all actions performed by other players previously, then the game is said to be of perfect information.*
- *Complete information sequential games vs incomplete information sequential games. In incomplete information games, the players may not know exactly what their or their co - player's payoffs may be.*
- *Discrete vs continuous games. When players can choose actions only from a discrete set of actions, then the game is said to be discrete as opposed to continuous games, where players can choose a strategy from a continuous strategy set.*

———————————————————————

Games can be represented in the following forms:

- The **extensive form** (Fig. 2.12). Here the games are represented with a tree and each node (or vertex) represents an action (or decision) taken by a player. The payoffs are noted at the bottom of the tree. The extended form can represent both sequential and simultaneous games. For the later case, while one player is moving, there is uncertainty on what the other players are doing. To symbolize this, one has only to connect the nodes with a dotted line where this uncertainty arises – or circle them. The connected nodes represent the possible alternatives of that move and show what information is missing. This is known as the *information set*.

  To put it alternatively, an information set of a certain player P, is the set of all possible moves that could have taken place so far, based on the information that player P may have. For a perfect information game, every such set consists only from singletons: sets that have only one member.

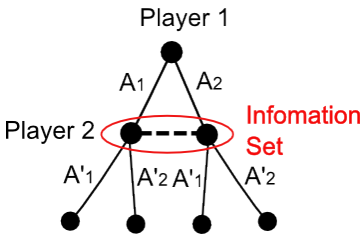

Figure 2.12: A game with imperfect information represented by its extensive form.

	Player 2 chooses "Silent"	Player 2 chooses "Betray"
Player 1 chooses "Silent"	-1,-1	-10,0
Player 1 chooses "Betray"	0,-10	-5,-5

Figure 2.13: Normal form of the Prisoner's Dilemma. Two friends are arrested by the police as suspects for committing a crime. However, the police does not have enough evidence against them and decide to separate them in order to offer them a deal: if one testifies ("betrays" his friend) for the prosecution against the other and the other remains silent, the betrayer goes free and the silent accomplice receives a full 10-year sentence. If the both remain silent, they both receive only six months. If each suspect betrays each other, they both receive five years each.

- The **normal or strategic form** (Fig. 2.13). Here the games are represented with a matrix and each cell depicts the outcomes of the moves for each player. Within the cell, one marks the payoffs of that combination. This representation is used only for simultaneous games or games of imperfect information.
- The **characteristic function form**. For games of cooperation, no individual payoffs are assigned but instead one can use a group payoff $v : 2^N \rightarrow \Re$ for every coalition S arising from a set N of players.

Dominance Strategies, Non-cooperative games and Individual Decisions

An event E is *mutual knowledge* if everybody knows E. E is *common knowledge* if everybody knows E, everybody knows that everybody knows that everybody knows E with this pattern continuing towards infinity.

A strategy σ_i^* strictly dominates σ_i ($\sigma_i^* \succ \sigma_i$) if:

$$\forall \sigma_{-i} \in \Delta(A_{-i}) : u_i(\sigma_i^*, \sigma_{-i}) \geq u_i(\sigma_i, \sigma_{-i})$$

In the case of the Prisoner's Dilemma (Fig. 2.14) the silent strategy is strictly dominated by the betrayal strategy. Regardless, what the other player may choose, a player will be better off if he decides to betray his co-player. However, this can mean that he will be trapped in a sub optimal outcome. This concept shows that rational, selfish decisions are not necessarily optimal for a player.

Let $\sigma_{-i} \in \Delta(A_{-i})$. α_i^* is a pure best reply to σ_{-i} if:

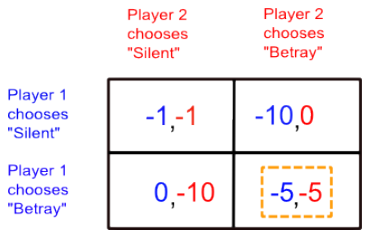

Figure 2.14: Strong Nash Equilibrium in Prisoner's Dilemma: Always betray your friend.

$$\forall \alpha_i \in A_i : u_i(\alpha_i^*, \sigma_{-i}) \geq u_i(\sigma_i, \sigma_{-i})$$

A strategy profile $\sigma^* = (\sigma_i^*, ..., \sigma_n^*) \in \Delta(A_1)$ X ... X $\Delta(A_n)$ is a *Nash equilibrium* [Nash, 1950, Nash, 1988] if for any $i \in N$:

$$\forall \sigma_i \in \Delta(A_i) : u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*)$$

Coalitions and Cooperative games

A cooperative game is a game where players form coalitions and they compete against other coalitions for some common utility. In a mathematical form every coalition S of a set N of players is assigned a payoff value $v : 2^N \rightarrow \Re$ describing how much value can a group of players earn by forming that coalition S.

There exist different methods for finding ways to fairly allocate the collective payoff to players of the coalition. Some of those solution concepts are known as the Core, the stable set (known as "von Neumann-Morgenstern solution") [von Neumann and Morgenstern, 1953], the Shapley value [Shapley, 1953] and many others. However, these methods are interesting when more than two players are involved, something that is not the case for this thesis where only two drones cooperate to guarantee visibility of a target.

However, there exists a concept in game theory that can be of interest for cooperation and that is *Pareto efficiency*. If one imagines that there is more than one way to distribute a common good or utility to a group of players, then a movement from one type of wealth allocation x to another type of allocation y that augments the wealth of one player, without making the situation worse for another player of the coalition, is called a Pareto improvement: $y \succ x$. Thus, a good allocation is called *Pareto optimal* or *Pareto efficient* [Fudenberg and Tirole, 1991], when no further Pareto improvements are possible within that set of players. The set of choices within a coalition that are Pareto efficient are known as the *Pareto frontier* (Fig. 2.15) or Pareto set. The Pareto frontier describes the maximum possible welfare for a given group of players. If one returns to the example of the Prisoner's Dilemma as it was seen earlier (Fig. 2.14), we can see that while the Betray-Betray strategy is a Nash Equilibrium, it is not Pareto efficient and that is why it seems so counter intuitive to us.

Differential games

Classical game theory representations like the matrix form and the extensive form seem to be excellent tools for formulating and proving the basic theorems of game theory but they seem to be unsatisfactory tools for solving real life problems.

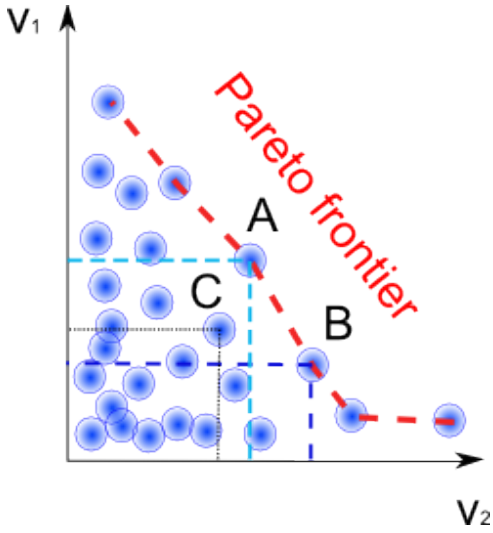

Figure 2.15: This graph depicts the Pareto frontier. Each axis is the welfare of a player, in an imaginary game where more utility is best. While decision C is better than decision B for player 1, it is a much worse decision for player 2 as it provides less wealth and thus it is dominated by point B. As a matter of fact, the only decisions that are not dominated by other decisions, are the ones that are on the Pareto frontier. In other words, the Pareto frontier depicts the maximum possible welfare for a given group of players.

Unless the problem is very simple, (*e.g.* very few players, with few discrete options for each player) the problem matrix becomes astronomically large, quite difficult to track and even more difficult to solve. This is why another method is needed to tackle the problems by analyzing the inner logical patterns of the games and producing some set of equations that one can solve.

Differential games [Isaacs, 1999] is a branch of game theory where players move sequentially instead of alternatively, and is mostly used in economic applications. It uses notions as barriers, zones and backward induction to deduce whether a pursuer can 'capture' an evader or not. In most cases the strategies are not discrete but continuous and in several cases, it can provide the control trajectory and use it to optimally guide a pursuer to a capture.

Classical problems for differential games are battles, pursuit-evasion games, dogfights, football, missiles intercepting aircrafts and many others.

One must bear in mind that in most cases differential games treat two-player zero sum games with perfect information. While there exist differential games that are discrete, for most cases differential games are considered to be continuous. This means, that players are making their decisions by adjusting the values of certain *control variables* that are the equivalent to strategies from classical game theory.

These decisions affect the values of certain quantities in the game that are known as *state variables*. These variables contain all necessary information for the game so that if one of the players had to stop and be replaced by a new player during the game, the new player would have all necessary information in order to continue the game from that point on – without an external viewer noticing the switch.

UAV applications for game theory

In [Sujit and Ghose, 2005, Sujit and Ghose, 2004] the authors present a game theoretic method for searching an area using $N \geq 2$ drones. As utility they use the uncertainty of the existence of a target in a given area or not.

In [Tomlin et al., 2000, Mitchell et al., 2005] the authors provide an algorithm that based on the above rationale permits to backwards calculate the initial configurations for collision avoidance. In [Reimann, 2007] the author provides three ideas related to differential games. For a multi UAV, multi target scenario he either proposes to decompose the problem in many mono UAV, mono target problems that he resolves using either differential games or the principle of maximum.

2.4 Competitive Approaches

When there are more than one agents acting in an environment and when each agent's success depends on the other agent's actions, then we are dealing with a *multi-agent environment*. The interactions that can arise in such an environment can be either cooperative, neutral or competitive.

Tracking a ground target is a *multi-agent environment* and in the quest for an *optimal decision*, the behavior of the target should be taken into account. Here we will consider the case where the target is trying to evade the drone and as such it enters in the context of *game theory*, the game being a *zero-sum game* [Fudenberg and Tirole, 1991]: if one agent wins then the other one loses. The notion of winning and losing is evaluated using an *evaluation function* which is tied individually to each problem *i.e.* a common class of problem is that of *pursuit-evasion games*, where one group tries to track down another group in a given environment. Even more both targets are considered to be *rational players*: they both seek to play optimally.

Pursuit evasion games

Any problem that involves a group trying to track down another group is called a *pursuit evasion game*. In UAV literature the solutions for this class of problems could be divided in two types: theoretic based and planning based approaches.

the most widely planning based method is a control method known as *Non Linear Model Predictive Control* (NMPC). It was used in robotics for the first time in 2000 as an application for an autonomous submarine that could follow the seabed at a fixed depth.

[Sutton and Bitmead, 2000]

NMPC is a method that was originally designed to be used for control of non-linear processes using a predictive iterative optimization. It consists of a two step loop:

1. First, a non-linear model projects the future state of the process using the optimal command found during the previous iteration,

2. An iterative optimization method based on Lagrange multipliers and steepest descent gradient optimizes a certain *cost function*, deriving the command that will be used during the next iteration.

The applications that have used this method include aircraft pursuit-evasion games

[Kim and Shim, 2003], helicopter obstacle avoidance, helicopter autonomous navigation [Shim et al., 2003][Lee et al., 2003][Shim et al., 2005] and path finding for a swarm of ground vehicles in the presence of obstacles [Xi and Baras, 2007].

The above iterative optimization step can be replaced and one can instead use an approach based on either Mixed Integer Linear Programming [Richards and How, 2003] or Convex Quadratic Programming [Singh et al., 2001] to derive the optimal command. These methods have been applied in spacecraft rendez-vous, automated air traffic control and autonomous ground robot navigation applications.

Autonomous pursuit is a class of applications where an autonomous drone tracks one or multiple targets in the presence of areas with threats, obstacles and restricted zones. One approach to this class of problems is the use of a state prediction in combination with a rule based method [Zengin and Dogan, 2007].

Studies that examine *visual pursuit* include the work presented in [Muppirala et al., 2005], where the pursuer and evader are considered to be connected with an imaginary fixed length bar. This way, the problem is transformed to a motion planning problem with sole goal to find a trajectory where the bar does not touch any obstacles.

However, in all the above cases it was considered that the UAVs have a complete FOV which often is not the case. There exist methods that show analytically a way to perform tracking using pursuit-evasion games [Bopardikar et al., 2007] for ground robots having only disk visibility. The theoretic case of search under *limited visibility* has been given an analytic solution in [Gerkey et al., 2006].

Adversarial searching

Adversarial search is a process that seeks to find the optimal decision for one player, while taking into account the actions of its opponent. This is done by trying to get the *Minimax value*, which is the best achievable payoff against the best possible play of the opponent. The algorithm that achieves that is the *Minimax algorithm* which is known to be complete, optimal (if the opponent is rational), it has a time complexity of $O(bm)$ and a space complexity of $O(b^m)$ for depth first exploration – with b being the branching factor and m the depth of the calculations.

Another well known method is $\alpha - \beta$ pruning. It ignores the nodes of the of the search tree that do not influence the final decision. This way it is possible to compute the correct minimax decision without having to process the whole search tree.

Example 2.4. ———————————————————————

Chess and adversarial searching

Few games have been studied as much as chess. In May 1997, news circled the globe that world champion Garry Kasparov had been defeated by a computer. The machine, named Deep Blue, apart from a massive parallel array of processors and custom made VLSI chips, used adversarial searching as its core algorithm. Deep Blue used an iterative deepening $\alpha - \beta$ pruning searching algorithm [Campbell et al., 2002] [Marsland et al., 1991] that permitted it to calculate an average of 126 million positions per move in a game that has a search branching factor of 35, leaving no place for resource allocation errors. Only the evaluation function, included 6000 to 8000 features. To enhance this effect, it searched the top levels of the chess tree, examining interesting positions and developing them beyond the standard processing horizon [Campbell et al., 2002].

Figure 2.16: During the last, famous sixth game, world champion Garry Kasparov loses against IBM Deep Blue computer.

Minimax *[Norvig, 1992] is an algorithm used very often in adversarial searching. Each configuration is given a numerical cost using an* ***evaluation function****. In the game of chess that would be the total value of pieces at their current position, plus their possible capability of taking an opponent's pawn. Then the algorithm predicts all possible movements and selects one based on its value. The basic, single depth step is known as a ply.*

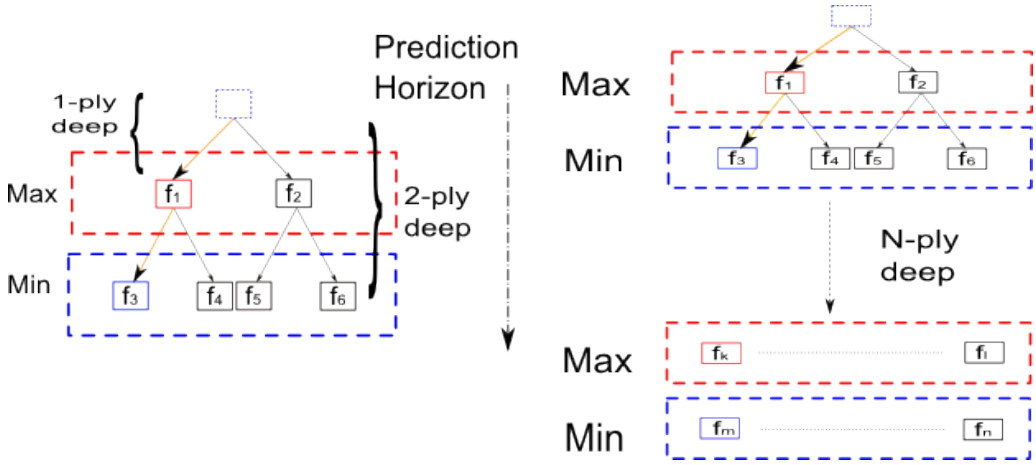

Figure 2.17: The basic round of any game, is the move of one of the two participants of the game and it is called a Ply. Here for reasons of simplicity the game is considered to have only two possible moves. In chess the average branching factor is about 35 and in the case of a robot path planning it can be all possible directions and speeds of movement. The **minimax algorithm** will choose either f_1 or f_2 (*evaluation function costs*), depending on whether it wants to maximize or minimize the outcome of the game at that ply.

Iterative deepening minimax *search is a minimax search of N-ply depth that is preceded by all N separate searches from* 1 *to* $N-1$*-ply depth. At each round, it either maximizes (if it is its own round) or it minimizes the evaluation function – if the round belongs to its adversary.*

However, what really won Kasparov was something even more quirky: a form of minimax algorithm designed to sacrifice short sighted wins in order to win bigger moves further in the horizon. This was what made Deep Blue ignore the bait of Garry Kasparov, when in a critical

move during the third game, he tried to sacrifice a pawn. A minimax algorithm very similar to this concept was published in [Gordon et al., 2006] and it is known as the Trappy minimax*: a trap is a move that looks good in the short term but has bad consequences in the long term. The only way to avoid these kind of traps is by performing full width minimax search to a depth greater than the final negative move. However, because of the branching factor, this is not easily achievable in every type of game, leaving us with a method that can win even the very best players in some games.*

Stealthy Navigation and Visibility Issues

Stealthy navigation or covert path planning is the process that permits a robot to navigate covertly in an environment using the route that offers the least visual exposure to adversaries.

In [Teng et al., 1993] the authors propose the use of a two grid cell representation of the terrain. They then use a point-to-region visibility analysis to calculate for every cell grid the map where it has visibility on the map. A point is visible to a given viewing point if and only if the line segment joining them in the three-dimensional space lies completely above the terrain. Then, each cell is compared to be checked if its elevation angle to the corresponding grid cell is greater or not than the angle of the line segment described earlier. In [Tews et al., 2004] the proposed idea is to use potential functions to attract agents in shadowed areas and repel them from areas where they are visible to adversaries. The construction of a visibility and accessibility map with the use of of harmonic function is proposed in [Ravela et al., 1994].

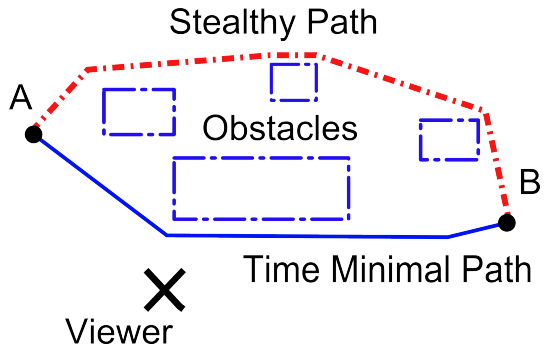

Figure 2.18: Stealthy Navigation. In this example, the agent knows the position of the viewer and decides to use the path that is more occluded, instead of the time minimal path, in order to get from point A to point B.

In [Marzouqi and Jarvis, 2003b] the authors use of an environment estimated visibility map with an estimated visibility cost value that represents the degree of exposure of that grid cell to the entire environment. This map is used together with an obstacle map to generate the visually optimal path for the robot. In [Marzouqi and Jarvis, 2003a] the same method is used but this time without recalculating grid cells that have already been treated.

In [Kim and Kim, 2008] a drone that moves in an urban environment tries to optimize the position of a circular trajectory in order to achieve a better visibility of the target. The visibility is examined based on the integral of visibility all around the target.

The authors of [LaValle and Hinrichsen, 2001] present an algorithm that permits to search

for unpredictable targets moving arbitrarily fast in a simply connected two dimensional curved environment.The algorithm is based on critical visibility events that occur because of inflections and tangents to piecewise smooth environments.

Target Trajectory Prediction

One of the most important aspects of adversarial target tracking is to be able to predict the future trajectory of the target. If one can successfully achieve that then target tracking becomes much more simpler. However, there is no reason that this information will come only under the assumption of an evasive target. Furthermore, in some cases this can be also a false presumption. There exist three other methods that may permit to predict the future trajectory of a target:

- **Prior target knowledge based prediction.** This can be made *e.g.* using a *Kalman filter* to predict future trajectories [Berg, 1983]. However, there are some serious drawbacks with Kalman filtering as in order to describe certain behaviors as human movement or vehicles arriving at crossroads, Kalman filters are not adequate.

  [Chong et al., 2000].

  However, in [Pannetier, 2006] there has been a serious work to achieve exactly that. The basic ideas behind this thesis was to use GIS information to transform the road network to a graph and then use that graph as a Markov chain in order to estimate the position of a target. The author proposes the use of *interacting multiple models* in order to estimate the position of a target on a road network.

- **Route Planning.**

  Route planning software is nowadays used in order to provide a way to get from point A to point B within the road network while minimizing some form of user defined metric [Edelkamp and Schrodl, 2003, Letchner et al., 2006]. They first model the road network as a graph, and then use graph search methods to find a route between two given points. This could be used in the cases where the target the UAV knows the end destination of the target.

- **Black-box based prediction**

  Black-box prediction is in essence *time series forecasting without* the use of a model. It consists of methods that use previous measurements to predict future trajectories of the process.

  Artificial neural networks (ANN) have been used in [Payeur et al., 1995] and

  [Helble and Cameron, 2007] as follows: the ground robot is considered to move in a 2D space. Each Cartesian axis is treated as an independent time series. In order to forecast future behavior the time series was then passed to a ANN that successfully forecasted its future behavior for a given time horizon.

  Grey predictors have been used in [Luo et al., 2001] to successfully predict the trajectory of a ground robot. The authors then used this method to make two robots chase each other.

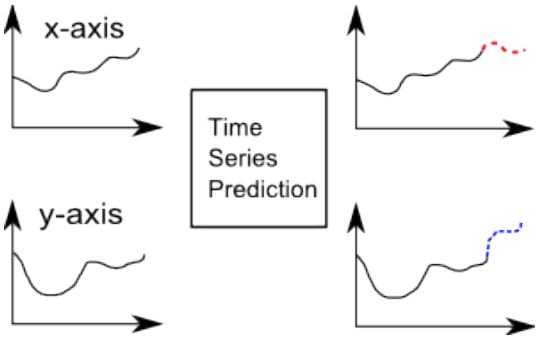

Figure 2.19: Target trajectory prediction. The trajectory of the ground target is decomposed in two time series, the Cartesian x and the Cartesian y. Those two time series are then fed to an ANN or a Grey Predictor that can give the future position without having a model or prior knowledge of the target.

Synthesis of Competitive Approaches

Target tracking has been studied by roboticists for quite some time, but mostly under the assumption that the target is not trying to evade. An evading target should be considered as an agent that will actively try to show intelligent behavior and attempt to maximize a certain utility.

Multi-agent robotic environments have been studied to some extent but the agents were known to cooperate. Robotics research is mostly fueled by the nature of the applications it is called to serve and for the most of the time, they were civilian: joint exploration, search and rescue, cooperative mapping are some examples of multi-agent robotic applications.

However, UAVs are slowly turning that page and introduce a novel application to robotics research: conflict. This is due to the fact that UAVs are most often used in some form of military, surveillance or strategic application where the other agent is assumed to be non cooperative. A case where AI has touched the study of conflict has been during the study of automated chess playing that brought results using methods closely related to brute force calculations: adversarial search is the most successful example.

As a result there is a lot of place for improvement in this domain. In target trajectory prediction there have been studies that can predict the behavior of a ground target using time series analysis. However, these algorithms do not take into account the presence of ground obstacles or of a road network. It would be interesting if one could take into account those factors and introduce them into the time series analysis in such a way that it could enhance the forecasting capability of those tools. The idea that can be borrowed from route planning is to transform the road network to a graph and use that to predict the future position of the target.

Furthermore, decision theory and game theory could be largely used to maximize the utility that the UAV tries to maximize, while taking into account the actions of its opponent. Finally, the tools developed for playing chess could be used in UAV pursuer-evasion games. This could be done by first predicting what is the best move for the target and then finding a best response to that move.

2.5 Cooperative Approaches

This section covers the work that has been done where multiple drones are trying to cooperate in order to maximize a certain utility. That utility changes from study to study and it is not confined to target tracking.

Multi-UAVs are mainly known for their formation flight applications. However, we do not present this type of applications because they are different from the interest of this thesis.

One first approach to collective target tracking/observation is given by *decentralized data fusion (DDF)*. The main idea behind this type of methods is to form some form of *distributed* Kalman filter or information filter in order to fuse the different information as they are acquired by every drone [Ridley et al., 2003, Ridley et al., 2002, Grocholsky et al., 2000, Parker, 2002]. Very close to this idea, we can find the work done in [Campbell and Ousingsawat, 2002]

[Sinha et al., 2005] where the UAVs change position in order to maximize the *collective information gain* at every run. When this position is orientated to a better estimation of the target, organized flocking behavior arises. This was shown in [Olfati-Saber, 2007] by means of simulations.

The authors of [Tang and Ozguner, 2005] examine ways to maximize *cooperative visibility*. They use a *steepest descent gradient based law* that attempts to *minimize the time between two consecutive observations* of the target by forcing the agents to pass next to the targets, when that is deemed necessary. One very important aspect of the proposed idea is that of *task decomposition* to assign the different drones to different tasks – actually traverses that maximize visibility. Once a drone has finished its traverse, it is assigned a new target.

One very important group of publications comes under the umbrella of multi UAV exploration. In [Schumacher et al., 2002] the authors propose a method for swarm searching, where the search allocation is formulated as linear programming problem. The authors of [Kitamura et al., 1993] propose a search scheme were the searchers search on a graph. In [Spires and Goldsmith,], space filling curves are used to cover the search space. Finally, in [Burgard et al., 2002] base the cooperation study on communication between the groups.

A *two level control* is seen in [Jung and Sukhatme, 2001]: A coarse deployment controller distributes robots across regions using *a topological map* and *density estimates*, and a target-following controller attempts to maximize the number of tracked targets within a region.

Another type of approaches to cooperative target tracking is provided by *game theory*. In [Ganapathy and Passino, 2003] the map is divided at a grid and the authors provide a study on how to distribute tasks when the drones must go to specified areas. Their metric is the time it takes to complete the mission and they show that game theoretic cooperative methods lead to much better times than non cooperative ones. In [Liu et al., 2003] the authors examine how to org anise a group of cooperative agents when they are facing a common adversary. The method is based on calculating the *minimax formula* for one to two steps look ahead evaluation functions. Finally, in [Li and Payandeh, 2003b, Li and Payandeh, 2003a] the authors show that manipulation of an object by several fingers can be seen as a game theoretic approach of cooperation. The methods make use of *Nash equilibrium* and *Pareto optimal agreement strategies*.

Another approach to UAV cooperation is the one proposed in [Beard and McLain, 2003, Beard et al., 2002, McLain et al., 2002]: first, a Voronoi graph is build around the obstacles and the threats. Then, it uses the K-best algorithm of [Eppstein, 1994] to find the best path for every drone. It ignores the paths that do not fulfill the constraints of non collision and communication distances. However, the authors have shown that the scheme that seems to

work the best for the group is to maintain some kind of hierarchy within the group: Every drone calculates the solution based on what the previous $N-1$ UAVs have done. This way the first drone calculates myopically only its own path while the last one does it by taking into consideration the missions of the whole group. In [Reimann, 2007] the author provides a differential games solution for a multi UAV, multi target scenario and proposes to decompose the problem in many mono UAV, mono target problems that he resolves using either differential games or the principle of maximum. He also proposes to transform the problem from a multi UAV to several in smaller minimum time decomposition problems.

In [Jin et al., 2003] the authors propose a simple cooperative approach, based on distributed assignment mediated through *centralized* mission status information. Using this information, each UAV assesses the task opportunities available to it, and makes commitments through a phased incremental process. The idea to use *supervisory control* has also been seen in [Girard et al., 2004, Maza and Ollero, 2004] where the area is divided to sub-areas where the UAVs are then assigned and act independently. Similarly, in [McLain et al., 2000] the algorithm proposed there divides the overall task to small sub optimal tasks and assigns them to each drone. Then each drone tries to find a path that is close to the global group mission. This is done using cost functions and treating each task as a minimization problem.

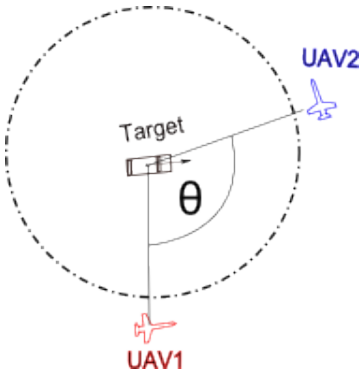

Figure 2.20: In this configuration the two drones remain in different clock angles in order to enhance visibility.

Finally, one very target tracking specific idea is to use *a clock angle representation* around the target and make sure that the drones remain on different clock angles around the target, maximizing thus visibility. In [Frew and Lawrence, 2005] they do exactly that using Lyapunov (Fig. 2.20).

Synthesis of Cooperative Approaches

Overall, the work of multi UAV target tracking seems to be gathered in two groups: decentralized data fusion and coordinated space search. The other work that has been done remains in the subject of game theory cooperation and hierarchic supervisory cooperation schemes. Finally, one should remember to mention the beautiful clock angle representation (Fig. 2.20).

One gets the impression, that as research on UAVs is relatively new, research on multi-UAVs is even newer in domains related to target tracking and we should expect to see much of this research to raise in the recent years to come.

2.6 Overall Synthesis of State of the Art

The research on UAVs provides a strong framework on Guidance, Navigation and Control. A UAV can perform very accurate trajectories if that is desired. Furthermore, there exist several solutions that can provide autonomous target tracking under simple assumptions (specific camera visibilities and ground target assumption). Planning methods can provide obstacle avoidance that can also be used in UAVs after they have been adapted to specific UAV dynamics. game theory offers a strong decision framework so that real autonomy can exist. Research on cooperative aerial robotics has provided some solutions for different UAV combined applications and pursuit evasion scenarios have been examined under the non cooperative multi agent interactions framework.

Chapter 3

Problem Statement and Formulation

The goal of this chapter is to present the hypotheses made to tackle the problem as well as the associated notations and vocabulary. The first section proceeds by explaining all the assumptions dealing with the UAV. The second section presents the target assumptions, and the third section presents the assumptions on the environment. The mission criteria are then presented.

3.1 UAV

3.1.1 Trajectory

The UAV is considered to be moving with a steady speed v_d, in a steady altitude h_d. The model that describes such an agent is the Dubin's aeroplane as presented earlier in the state of the art. The model we are using during this thesis is a simplified, kinematic version of the lateral dynamic of an aeroplane (Fig. 3.1):

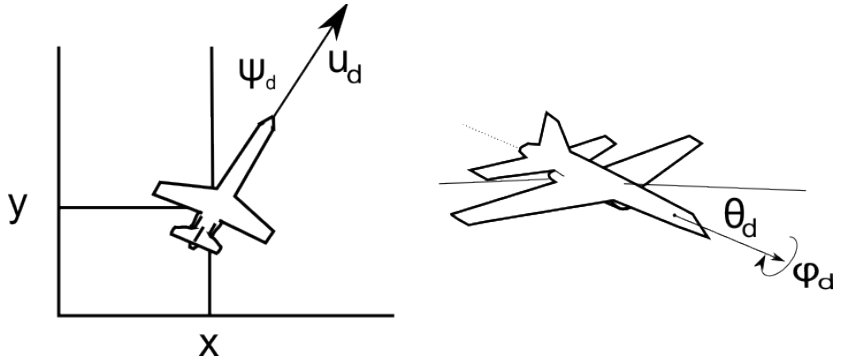

Figure 3.1: Figure depicting the different angles used in this model.

$$n = \frac{1}{\cos|\phi_d|}$$

$$R = \frac{v_d^2}{g\sqrt{n)^2 - 1}}$$

$$r = \frac{sign(\phi_d)v_d}{R)}$$

$$\psi_d = \psi_d - r$$

$$\dot{x} = v_d \cos\psi_d \cos\theta_d$$

$$\dot{y} = v_d \sin\psi_d \cos\theta_d$$

Where ψ_d is the drone heading, θ_d is the pitching angle, r is the yawing rotation speed, ϕ_d is the roll angle and v_d is the drone speed.

It should be noted that aeroplanes have a bounded turning radius.

$$\rho_{min} = \frac{v_d^2}{g \tan\phi_{max}}$$

An important consequence of this formula is that the minimum turning radius ρ_{min} increases increases with the airspeed v_d: as a result the aircraft needs more space to perform a maneuver.

3.1.2 Camera Types

There are many types of cameras that can be mounted on a UAV. Cameras have different visibilities, different fields-of-view and they can provide problems as to how far they can detect a target.

The basic types are the following:

- Pan-tilt camera. This type of camera can pivot around two axes: a vertical and horizontal axis and it can provide a wide FOV but not necessarily a full FOV.
- Roll-only camera. This type of camera can pivot only around the roll axis of the aircraft and thus it can provide a much more limited FOV.
- Static Cameras. These cameras are fixed and their FOV is defined by the lens focal length.

Not all UAVs have cameras with full FOVs. Depending of their capacity to see sideways or not the cameras can be separated in two broad categories: the ones that have a good sideways visibility and those ones that have a bad sideways visibility (Fig. 3.2).

A constraint that exists in all types of cameras is that of distance. After a certain distance no camera can detect the target. This aspect of vision is known as disk visibility as the UAV can see up to a certain distance and not more than that.

The active zone within which the target can see the target is known as the *capture zone* (Fig. 3.3).

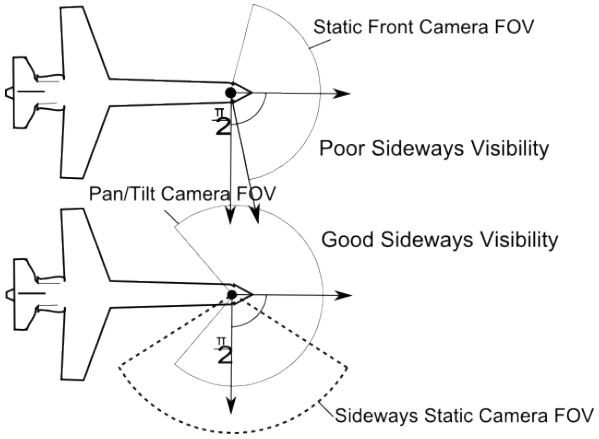

Figure 3.2: Different types of field of view and their sideways visibility

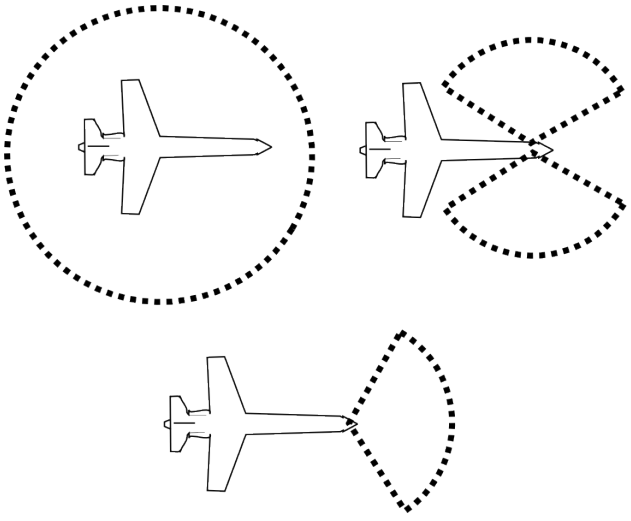

Figure 3.3: Different types of capture zones, depending on the camera visibility

3.2 Target

As a target we will consider vehicles and specifically two different types of cars:

1. **Off road vehicle.** A vehicle moving in a environment as a Dubins car:

$$\dot{x_t} = v_t \cos \phi_t$$

$$\dot{y_t} = v_t \sin \phi_t$$

$$|\phi_t| \leq \phi_t^{max}$$

$$v_t^{min} \geq v_t \leq v_t^{max}$$

2. **On road target**. Such a target moves linearly along roads, that are modeled as a graph (see below).

The target can be either neutral or evasive. An evasive target will attempt to maximize its distance with respect to the target and use any object in the environment to hide itself.

3.3 Environment

Drones are called to fulfill missions over different types of environment: urban, semi-urban or open environments. However, not all environmental features are of interest for us. This section attempts to model several of these features that influence the UAV as well as the target actions.

3.3.1 No Fly Zones

A No Fly Zone (NFZ) is a flying zone where the UAV must not enter. Under this broad title one can include various environmental features: while for a micro drone flying over a rural area, NFZs are the buildings that are high enough to reach its flying altitude, for a bigger UAV it can be the entire city. Depending from the mission, the capacities or the size of the UAV, the *granularity* of the No Fly Zones envelope changes (Fig. 3.4). Furthermore, there should always exist a security blanket around the NFZ to permit the UAV to recover from an unfortunate maneuver. Whatever the shape and size of those zones is, one thing remains certain: no autonomous drone should ever enter into such a zone and failure to do so indicates a problem of the algorithm. The reasons for that are obvious: safety for the inhabitants of that zone and safety for the flying vehicle itself.

For the rest of this thesis one will assume that despite the NFZs the ground is flat – unless otherwise is noted. When a NFZ is denoted in the map then the aircraft must avoid it using lateral maneuvers. Note that a realistic model of NFZs should contain their elevation, but since we assume the UAV flies at a steady altitude, we model NFZs in 2D, in the flying plane.

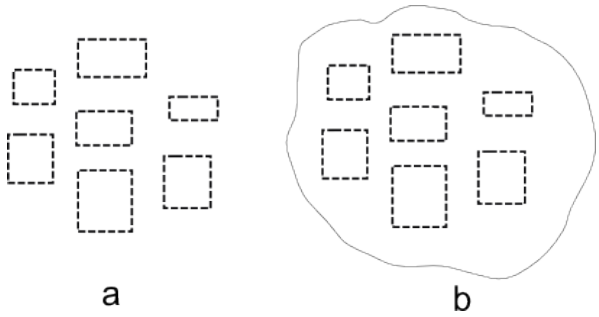

Figure 3.4: Two different ways to separate the No Fly Zones for the same area. A micro drone would only use the geographical information of the buildings (a), while a bigger UAV would have to avoid flying over the entire rural area altogether.

3.3.2 Visibility Obstructing Obstacles

Any obstacle that casts a shadow big enough to hide the target can be considered a Visibility Obstructing Obstacle or Visibility Obstacle in short (Fig. 3.5). Obstacles can have any shape or size but their existence is important as they can obstruct visibility from the air and thus influence the mission of a drone that is trying to track that specific target.

If one knows the exact shape and position of such an obstacle, one can calculate the position of the shadow using projective geometry. However, these kind of calculations go beyond the scope of this thesis and for this reason only very simple primitives will be used to describe the shadowed areas, their shadow being calculated only qualitatively and not quantitatively.

3.3.3 Road Network

Road network is important because it permits the UAV to predict how will the target move next. Besides, off road vehicles and humans, most ground vehicles need a road network in order to move. Under this assumption, one needs to have some form of representation of the network. For most advanced applications Geographical Information Systems or GIS can provide detailed maps that include every road on the area of interest.

Road networks are represented using graphs (Fig. 3.6). Nodes with more than two arcs leading to them signify crossroads and they are important as a UAV trying to anticipate the movement of the target can move to the wrong road before the target gets there, something that may lead to a loss of visibility.

3.4 Mission criteria

This thesis considers that the UAV will always try to keep the target within its camera field of view. In order to validate the algorithms, one must first set the criteria to use in order to achieve this. For our application, the most important criterion seems to be *what percentage of the overall trajectory does the drone succeed to keep the target within the capture zone.* And as the speed is considered to be steady throughout the whole trajectory: *what percentage of the overall time does the drone succeed to keep the target within the capture zone* (Fig. 3.7).

However, this image is quite optimistic as in reality the situation will be quite more

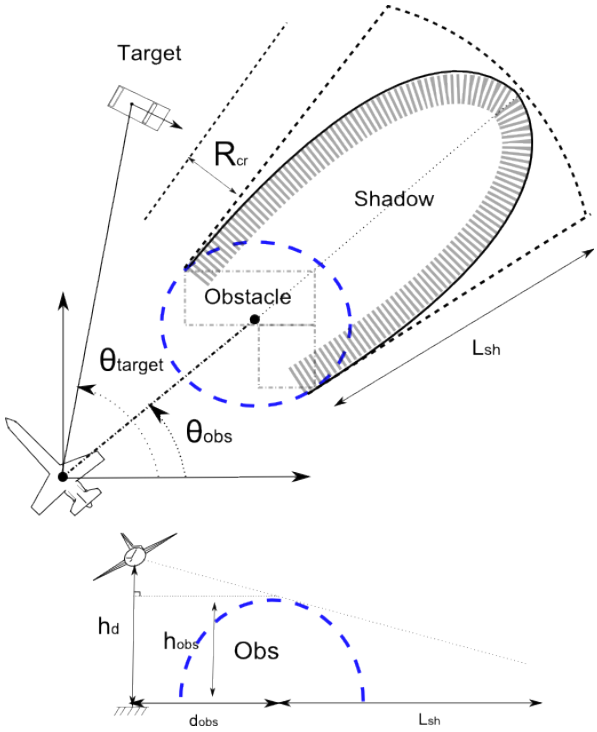

Figure 3.5: A visibility obstacle casts a shadow and this provides a good hiding area for the target.

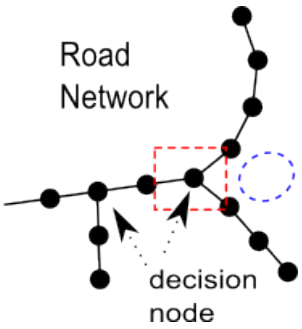

Figure 3.6: The road network as a graph.

complicated (Fig. 3.8): apart from an evading target, there will be numerous buildings, No Fly Zones and many, different obstacles that can providing hiding places for the target all over the landscape. This situation will work in favor of the evader and will make the work of the UAV quite difficult.

3.4.1 Metrics

In order to measure the trajectory ratio where visibility is maintained we are using the following formula that measures the mission's *visibility ratio* τ_{vis}:

$$\tau_{vis} = \frac{T_{vis}}{T_m}$$

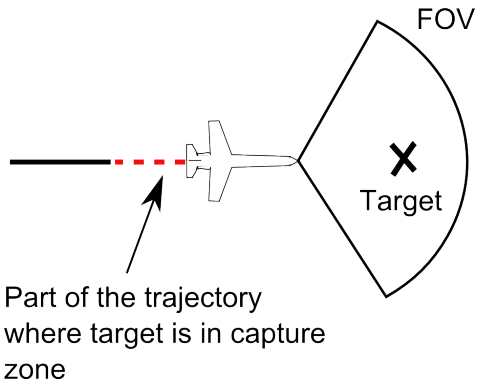

Figure 3.7: The basic mission is to maximize the portion of the UAV trajectory where the target is within the capture zone

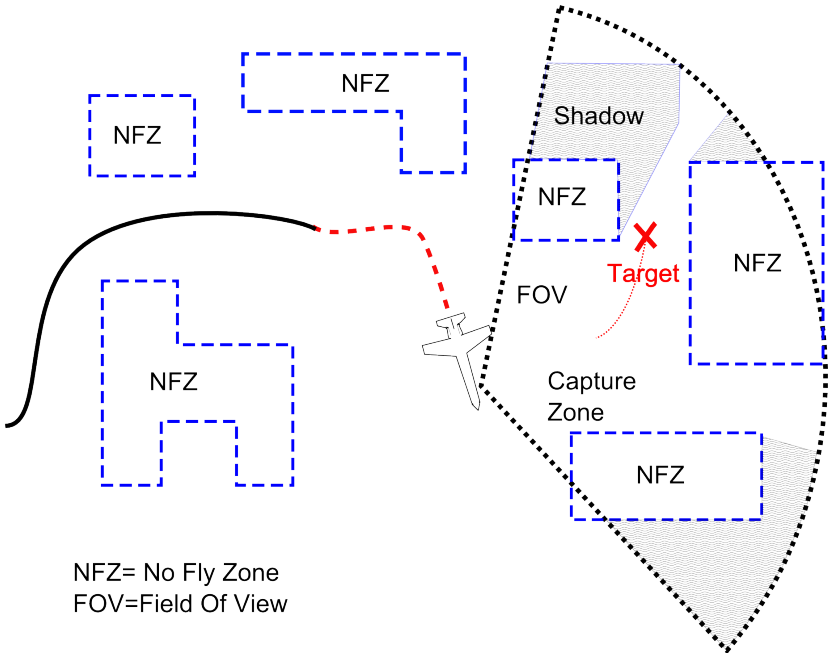

Figure 3.8: In reality the UAV has to face much more difficult situations where it has to juggle between avoiding obstacles (NFZs), remaining close to the target, having the camera at the right angle and making sure that no object is hiding the view of the target.

Where T_{vis} is the target visibility time duration and T_m the overall mission time duration. This metric gives us an idea for how well does the given drone algorithm manage to maintain visibility with the target.

3.5 Synthesis of the problem

Overall, the aim of this thesis is to endow a system composed of one or two drones with the ability to maximize the visibility ration of a ground target. The drone system must be able to adapt its decisions and its trajectory based on the following information:

- The type of cameras that it may be using, which in some cases may not provide a full field of view.
- The position prediction of a ground target that can change direction and speed.
- The position prediction of a ground target that can be evasive or neutral. An evasive target may not only try to maximize the distance between the drones and the target but it may use as well any hiding area in the ground to hide itself from the UAV(s).
- A map of all natural or human made ground obstacles, that can hinder visibility from a given angle in the air and therefore offer a hiding area for the drone.
- A detailed map of all non fly zones that have to be avoided.

Chapter 4

Tracking a target on a ground plane

As seen earlier, most of the UAV target tracking methods are based on Guidance, Navigation and Control (GNC). This chapter presents a GNC based approach to UAV target tracking under limited visibility. The following issues are addressed:

- Unification: Provide a method that works with all types of cameras and mountings.
- Good Visibility: Maximise the visibility of an autonomous UAV system.
- Compare existing methods.

4.1 Problem Statement

In this chapter, we consider the case of a UAV that flies at a steady altitude with a steady speed and attempts to track a neutral moving, ground target. "Neutral" means that the target is not aware it is being monitored, and so does not try to cooperate or to escape. It is moving on the ground with variable speed. There are no obstacles that hinder visibility and in most of the chapter it will be considered that the UAV has full capability of moving in space (no-NFZs). However, the case of NFZs is briefly covered in order to provide a method that works under all conditions. Fig. 4.1 introduces some variables that are important in our approach:

- the 2D distance $|\overrightarrow{L_1}|$ which is the vertical projection on the ground of the target line of sight, *i.e.* of the vector that joins the camera optical center to the target,
- the angle η between $\overrightarrow{L_1}$ and the speed vector $\overrightarrow{V_d}$,
- the camera orientation with respect to the plane frame, defined by the two angles (α, β),
- the lateral and vertical tracking errors x_f and y_f expressed in the image plane. Given the camera focal length f, the target angular coordinates with respect to the camera optical axis are $(\arctan \frac{x_f}{f}, \arctan \frac{y_f}{f})$.

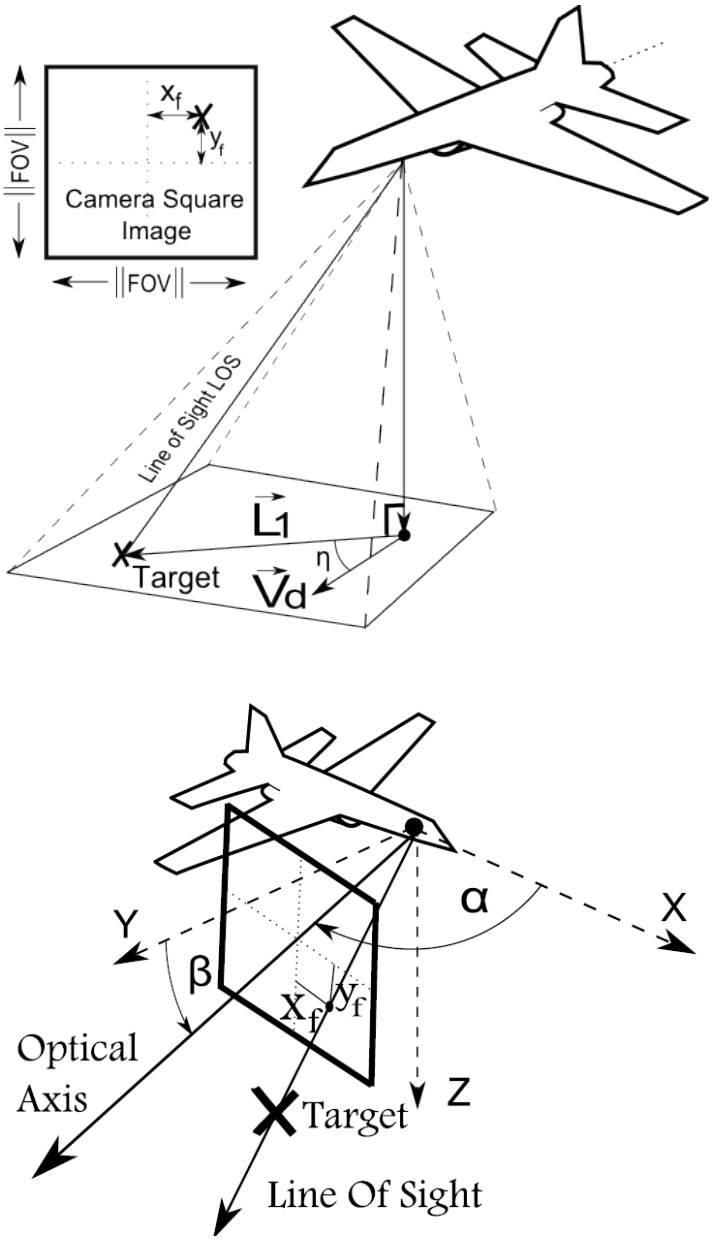

Figure 4.1: Aircraft/target geometry. Top: projection of the target line of sight on the ground, and definition of the angle η. Bottom: orientation angles (α, β) of the camera optical axis, and image coordinates (x_f, y_f) of the target.

4.1.1 Overview of our approach

Here we consider two cases:

- **Target out of sight:** If the target is out of the current field of view, the UAV uses a simple strategy that guides it where the target has lastly been seen. In the absence of information regarding the target location, various search strategies have been proposed in the literature (*e.g.* [Quigley et al., 2005a, Wong et al., 2005]).
- **Target in sight:** When the target is in the current field of view, our approach is to guide the UAV so as to optimize the target visibility. For that purpose, a two steps control strategy is defined:

  1. **An optimal angle $\eta^\star$ is computed.** "Optimality" means here that $\eta^\star$ is set in order to define a trade-off between the target distance, that must remain in the $[\rho_{min}, D_{max}]$ range and the lateral tracking error x_f of the target, which is to be minimized. D_{max} is the maximal detection radius and ρ_{min} is the minimum turning radius.
  2. **A Lateral Guidance Law helps the UAV maintain the desired heading $\eta^\star$.** For that purpose, the current η angle and the horizontal distance to the target L_1 are computed, and they are fed to a lateral guidance algorithm, which defines the roll command u_{cmd}.

Fig.4.2 shows the flow-chart of the approach. The two steps can be seen also as two loops: one internal and one external. The internal loop consists of a lateral guidance law and the external loop is a strategy that selects an optimal setpoint for the internal loop to follow. Note that while the inner loop of guidance loop is quite fast, the outer loop can be slightly slower – the sampling frequency depending of the camera acquisition capability.

We first present the lateral guidance law (step 2 - internal loop) and then we present the strategy (step 1 - external loop) that selects the optimal angle $\eta*$.

4.2 Lateral Guidance

When the target is in the field of view of the camera, the visual target tracking algorithm provides the image coordinates of the target (x_f, y_f). A lateral guidance law is used to control the heading of the aircraft (Fig. B.7). A significant advantage of this law is that it mainly depends on the heading error $\eta^\star - \eta$, which is straightforward to estimate, as it only depends on the relative position between the UAV and the target.

We use the variant of proportional navigation tuned for airplanes, introduced in [Park, 2004]. It is shown in [Park, 2004] that it outperforms the more classical methods of cross tracking [Niculescu, 2001] and proportional derivative guidance, in terms of tracking capability (error correction, speed of correction and trajectory following). By using this method we manage to follow the heading with greater accuracy than it would be with more classical means.

According to this law, the lateral acceleration needed to reach a target (Fig. 4.4) is given by:

$$u_{cmd} = 2\frac{V_d^2}{L_1}\sin(\eta^\star - \eta) \tag{4.1}$$

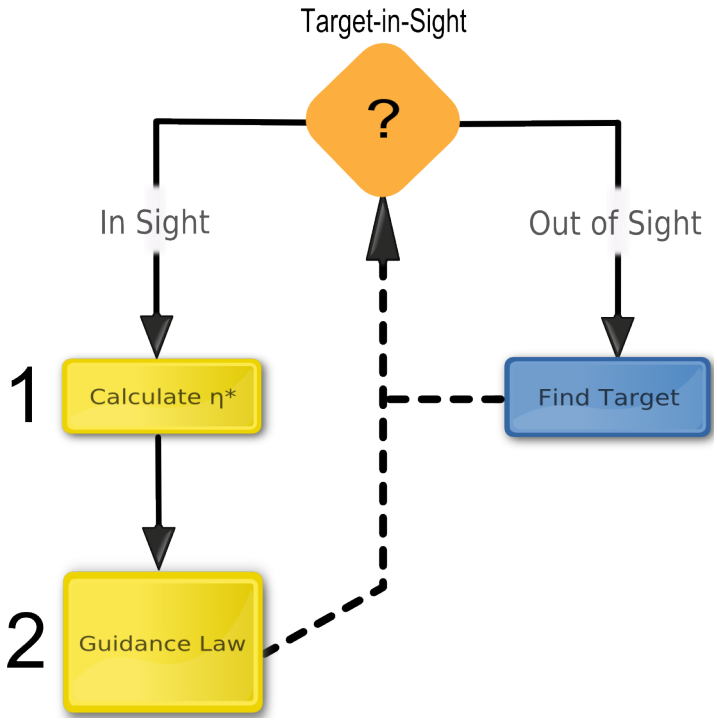

Figure 4.2: This is the flowchart of the approach. One can see that the aircraft repeats the steps 1 and 2 in a sequence.

Where u_{cmd} is the command of the lateral acceleration, V_d is the speed of the UAV and $L_1 = |\overrightarrow{L_1}|$ is the distance between the desired waypoint and the UAV. Considering that the desired waypoint is the projection of the target in the flying plane, L_1 and η are computed as follows:

- We can estimate the distance L_1 between the target and the UAV on the basis of the target image coordinates, the altitude h_0 and the two angles η and δ (Fig. 4.5).

$$L_1 = \frac{h_0}{\tan \delta} \tag{4.2}$$

Where

$$\delta = \arctan \frac{y_f}{f} + \beta + \phi \sin \eta \tag{4.3}$$

This relation stands if the UAV pitch angle p is equal to zero, a reasonable hypothesis since the altitude h_0 is a constant regulated value. Naturally, the closer the η angle is to $\frac{\pi}{2}$, the greater is the influence of the roll ϕ on L_1.

- η can be computed from the target image coordinate x_f, the camera focus length f and the angle *alpha* as (Fig. 4.4):

$$\eta = \alpha - \arctan \frac{x_f}{f} \tag{4.4}$$

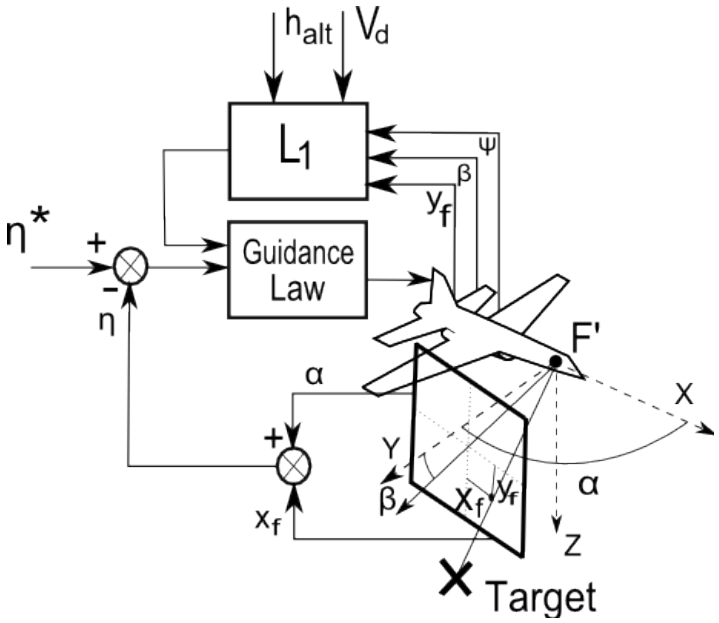

Figure 4.3: Lateral Servoing Loop.

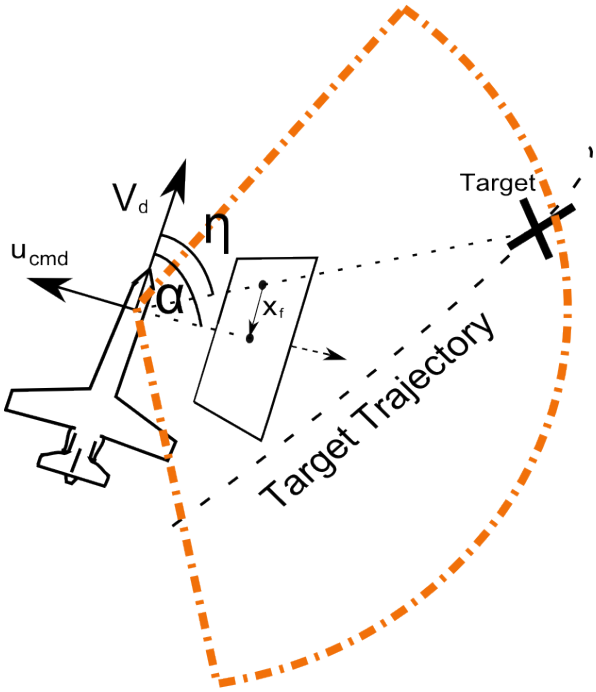

Figure 4.4: Lateral guidance Law configuration (top view).

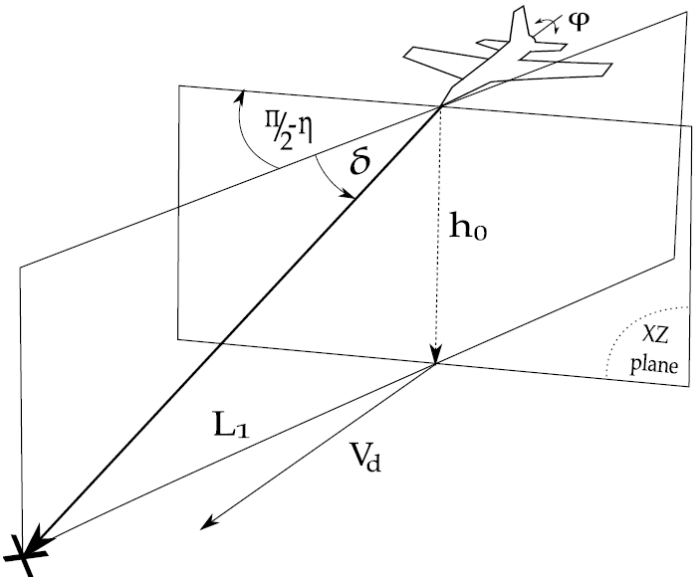

Figure 4.5: Calculus of the distance L_1.

Note that if the camera rolling angle is controllable, it can can be used to compensate the changes in the roll angle to minimize y_f. A simple proportional derivative controller can be used for that purpose.

4.3 Tracking Strategy

The setpoint angle $\eta^\star$ is defined in order to achieve two goals:

- Keep the lateral tracking error x_f as small as possible,
- Maintain the distance between the UAV and the target greater than the minimum flying radius ρ_{min}.

The first goal ensures a good target visibility, while the second one avoids situations in which, the target could enter areas that the UAV could not reach without complex maneuvers. The difficulty comes from the fact that depending on the camera configuration, the two goals may be contradicting one another.

4.3.1 Line of sight distance

To maintain a stable distance from the target, we must ensure that $\frac{d\overrightarrow{L_1}}{dt} = -|\overrightarrow{V}|\cos\eta$ stays as close to zero as possible, where $\overrightarrow{V} = \overrightarrow{V_d} - \overrightarrow{V_t}$ is the relative speed between the UAV speed and the target speed. For instance, if $\eta = \frac{\pi}{2}$, then $\frac{d\overrightarrow{L_1}}{dt} = 0$, which implies that the aircraft is circling around the target. The following criterion is therefore to be minimised:

$$J_1 = V^2 \cos^2 \eta^\star \tag{4.5}$$

4.3.2 The lateral target image error x_f

To maintain a good visibility, *i.e.* to keep the target as close as possible to the optical axis, one must minimize the lateral image error x_f. Equ. 4.4 gives:

$$x_f = f \tan(\alpha - \eta) \tag{4.6}$$

The following criterion must therefore be minimized:

$$J_2 = f^2 \tan^2(\alpha - \eta^\star) \tag{4.7}$$

.

Finally, to maintain the target within the field of view, we must satisfy the relation $|\alpha_{max} - \eta^\star| \leq \frac{||FOV||}{2}$ where $||FOV||$ is the camera lateral aperture. Hence the following constraint must be satisfied:

$$\eta^\star \in [\alpha_{max} - \frac{||FOV||}{2}, \alpha_{max} + \frac{||FOV||}{2}] \tag{4.8}$$

The resulting overall criterion to minimize under this constraint is:

$$J = \lambda V_d^2 \cos^2 \eta^\star + f^2 \tan^2(\alpha - \eta^\star) \tag{4.9}$$

where λ is a weight that permits us to give priority to one of the two criteria. Two interesting consequences arise from the definition of this criteria:

- If the system has a good sideways visibility, *i.e.* $\alpha = \frac{\pi}{2}$ and $\eta^\star = \frac{\pi}{2}$, then we have $J = 0$ and thus a trivial solution for our problem, that consists in performing circles around the target.
- If the system has poor sideways visibility, then J_1 and J_2 can not be satisfied concurrently. The proposed solution is to use the largest possible $\eta^\star$, which is bounded by $\alpha_{max} + \frac{||FOV||}{2}$. Since $\eta^\star \leq \frac{\pi}{2}$, the resulting trajectory is a spiral.

4.3.3 Tracking strategy

In the trivial solution where good sideways visibility is possible, then $\eta^\star = \frac{\pi}{2}$. Otherwise $\eta^\star = \alpha \pm \frac{||FOV||}{2}$, depending from the side that the target is approached. One must also ensure that the target distance remains below the maximal detection radius D_{max}, and that the minimum turning radius ρ_{min} constraint is satisfied. The following strategy is therefore applied (Fig. 4.6):

loop
  Calculate ρ_{min} according to V_d
  Calculate D_{max} according to h_0
  if $\rho \geq D_{max}$ **then**
    Move inwards towards the target ($\eta_{desired} = 0$)
  else if $\rho \in (\rho_{min}, D_{max})$ **then**
    Spiral inwards with an angle ($\eta_{desired} = \eta^\star$)
  else if $\rho \leq \rho_{min}$ **then**
    while $\rho \leq D_{max}$ **do**

Move outwards radially ($\eta_{desired} = \pi$)
end while
end if
end loop

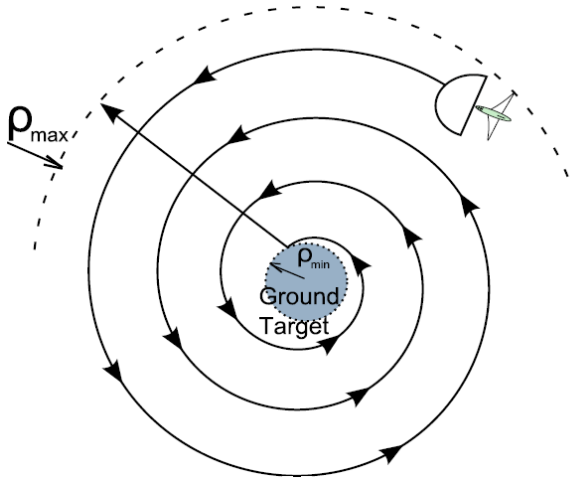

Figure 4.6: The spiral strategy.

Avoiding NFZs

In order to make the above method more robust, an obstacle avoidance algorithm is introduced into the loop. The algorithm looks for any NFZ that maybe close to the UAV and if there is more than one then it selects the one that is the closest and forces the UAV to go around the obstacle using bearing angles. There are two ways of achieving this: the first is presented in [Ducard et al., 2007] while the other one is a simple obstacle avoidance method presented here:

loop
if $d \leq R_{sec}$ **then**
$u_{cmd} = Kcos\theta/d$
else if $d \geq R_{sec}$ **then**
Resume previous algorithm
end if
end loop

Where d is the distance between the obstacle and the UAV, θ is the angle between the LOS of the UAV - obstacle and the $\overrightarrow{V_d}$ while finally K is a gain.

4.4 Results

4.4.1 Simulations

Simulations allow an easy evaluation of the visibility ratio $T_p = \frac{T_{Visibility}}{T_{Total}}$, where $T_{Visibility}$ is the accumulated time during which the target lies within the camera field of view, and T_{Total} the accumulated flight time.

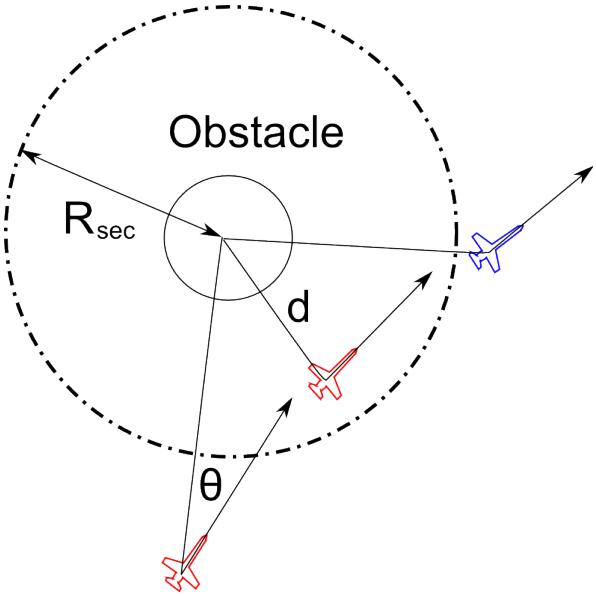

Figure 4.7: Graph depicting the obstacle avoidance setup.

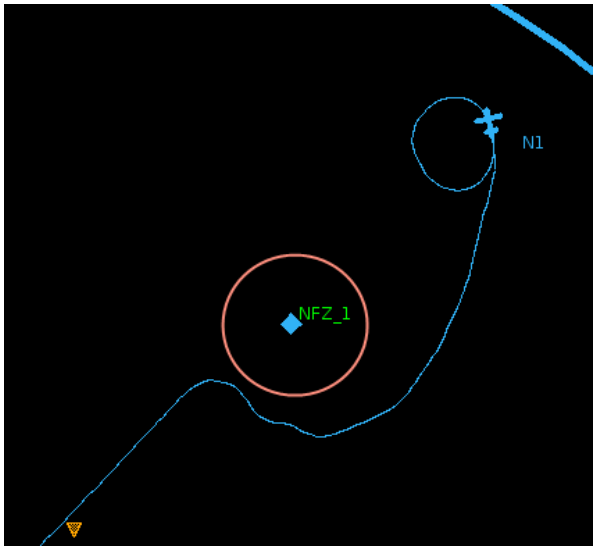

Figure 4.8: Simulation of a UAV avoiding an NFZ.

We compare the visibility ratio provided by our approach with other approaches proposed in the literature and presented earlier in the state of the art GNC (part 2.2.7):

[Stolle and Rysdyk, 2003] proposes a circular and a segmented arcs approach, [Lee et al., 2003] uses a double mode approach with the Sinusoidal and a loitering (Rose form) approach, [Rafi et al., 2006] proposes a simple circular form, [Quigley et al., 2005b] proposes circular trajectories with attraction fields and [Rysdyk, 2003] works on a clock angle frame. The comparisons happen for three different target speeds: $V_t = 0$ (static target), $V_t = 0.5V_d$ and $V_t = 0.9V_d$.

To establish these comparisons, we consider a camera mounted so as to exhibit a poor

	$V_t = 0$	$V_t = 0.5V_d$	$V_t = 0.9V_d$
Spiral and SpiralIn	97%	91%	82%
Two Arcs	78%	81%	97%
Circle	67%	69%	81%
Rose Loitering Mode	55%	60%	66%
Target Aim	51%	52%	55%

Table 4.1: Visibility ratio measured in simulations with different target speeds. The "target aim" scheme is defined by specifying $\eta^\star = 0$. The target was moving in a straight line with different speeds. SpiralI = [Rysdyk, 2003], Arcs =[Stolle and Rysdyk, 2003], Circle=[Stolle and Rysdyk, 2003][Quigley et al., 2005b][Rafi et al., 2006], Rose Loitering Mode = [Lee et al., 2003]

sideways visibility: $\alpha_{max} + \frac{||FOV||}{2} = \frac{\pi}{2}$. Table B.1 summarizes these comparisons: it shows that our approach exhibits a good visibility ratio and yields trajectories very similar to the ones presented in [Rysdyk, 2003], a more complex approach which relies on explicit trajectory definitions.

An interesting phenomenon arises when we see that for high speeds this strategy is outperformed by the segmented arc trajectory. This is because in high speeds (Fig. 4.9) the trajectory almost 'folds', demanding from the airplane to perform a turn with radius that is much smaller than its minimum turning radius. The aircraft cannot follow this turn, the command is saturated, the loop is broken and this leads to a loss of visibility.

Fig.4.9 shows the resulting trajectories generated by our control strategy for different target speeds, and Fig. 4.10 plots the evolution of the visibility ratio and the target distance ρ as a function of time.

4.4.2 Stability assessment for a noisy sensor

As the loop is based on a visual sensor, it is important to check how it responds to noisy measurements and up to what signal to noise ratio (SNR) can the algorithm continue to work.

In order to do this we add noise to our simulations, simulating a sensor that does not give good measures. Noisy and corrupt measurements are very common in robotics.

The definition of SNR is given by Eq. 4.10:

$$SNR = \left(\frac{A_{signal}}{A_{noise}}\right)^2 \tag{4.10}$$

Where A_{signal} is the amplitude of the signal, A_{noise} is the amplitude of the noise. This measure was achieved by taking the average of the signal and noise over a time interval. The noise was a randomly produced white noise with variable amplitude that was added on the estimation of η.

Table 4.2 shows that as the SNR diminishes so does the visibility of the system as well. We see that when the SNR falls under 10 the behavior of the pursuit deteriorates. When it goes under 1 the algorithm becomes unstable and the UAV clearly behaves as if it has no clue for where the target is.

One could interpret such a SNR as the percentage of the overall time that the sensor gives a correct reading. When the sensor corrupts more than 50% of the measures (SNR=1) the

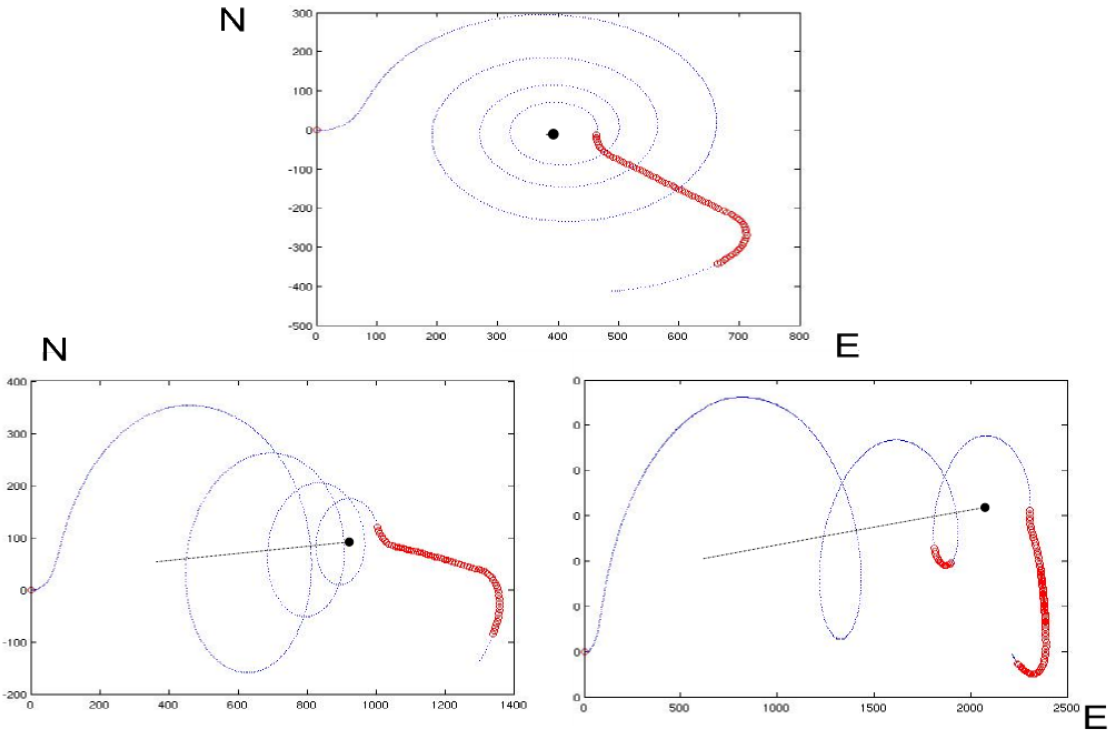

Figure 4.9: Trajectory generated by our approach, for three different target speeds. The trajectory is overlapped by big red points when the target is not in sight. One should notice that in the high speed version the aircraft cannot follow the command given by the trajectory strategy as the desired turning radius is smaller than the minimum turning radius.

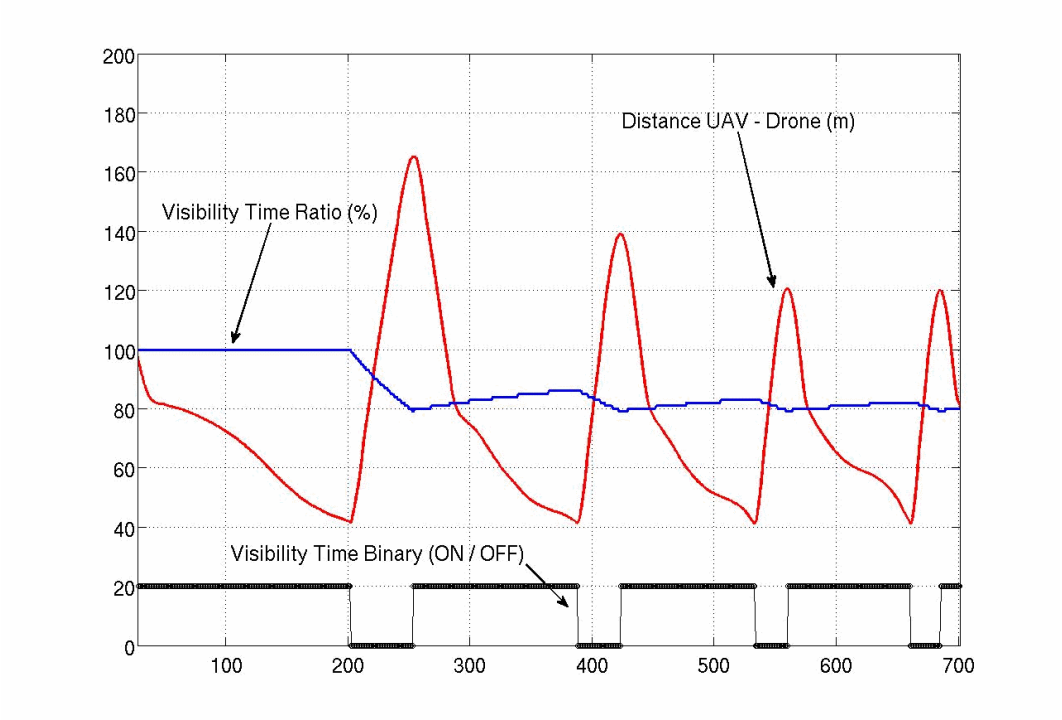

Figure 4.10: Evolution of the visibility ratio and the target distance ρ as a function of time. The target is static.

algorithm becomes unstable.

4.4.3 Flight Tests

We evaluated our approach with a hobby aircraft and a simulated target[1]. According to the position of the simulated target and the measured position of the UAV, the software could log the target visibility and distance measures. The motivation to develop such a hybrid real aircraft / simulated target experiment is to validate whether our strategy can cope with realistic conditions.

Our hobby aircraft has a $1m$ wingspan and a $0.80m$ length. We use the Paparazzi au-

[1]Integration of vision on board our aircraft in under way in the laboratory.

SNR	Visibility (%)	Comments
No Noise	69%	Good
25.8	63%	Good
0.7	51%	Not Good Enough
0.4	41%	Unstable

Table 4.2: Visibility ratio versus the signal to noise ratio of the sensor signal. One can see that as the noise increases the quality of the pursuit deteriorates. When the SNR drops under 1 the system starts to behave bad.

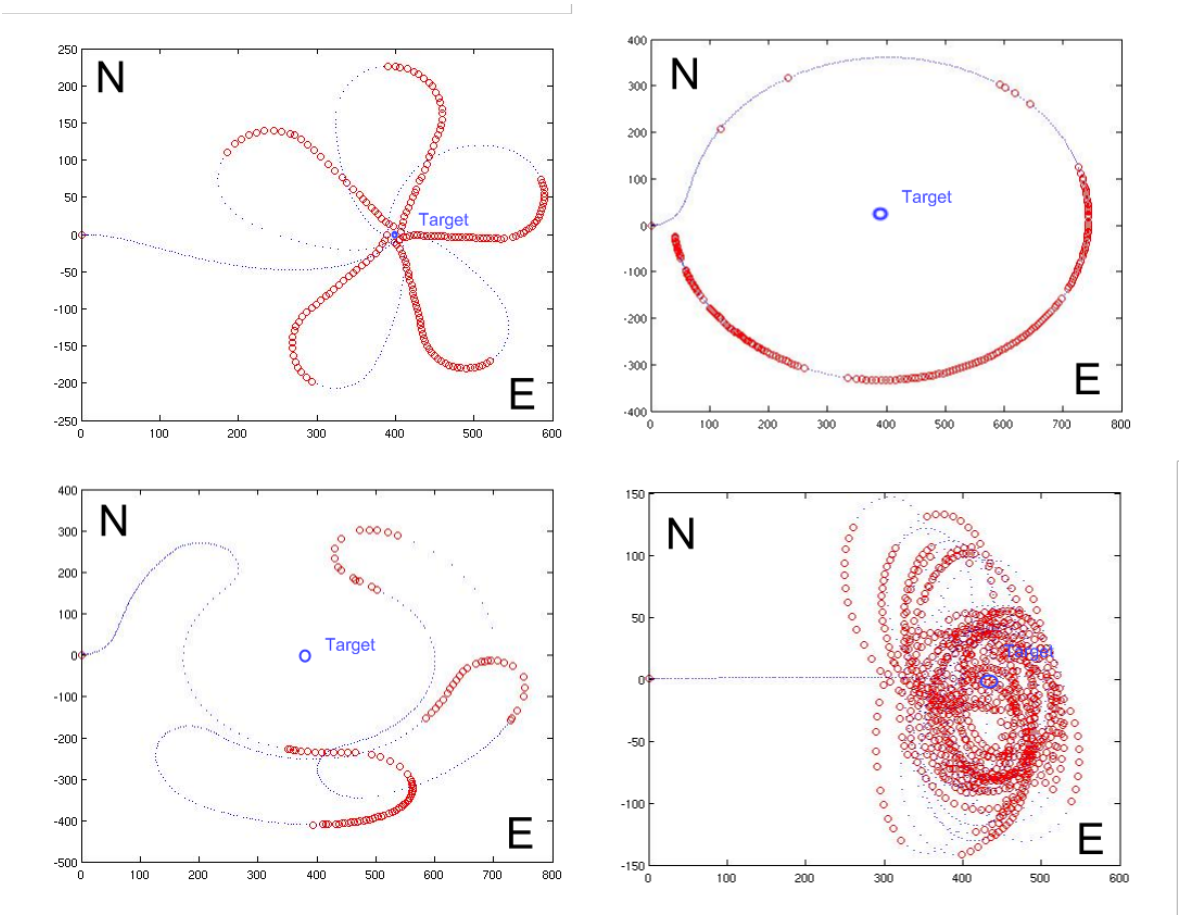

Figure 4.11: Different Strategies proposed in the literature, with a static target. The Rose approach (up left) was proposed in [Lee et al., 2003] and the Two Arcs approach (down left) was proposed in [Stolle and Rysdyk, 2003]. The top right trajectory is the application of circular trajectory scheme, with small wind perturbations: it exhibits less than 50% visibility in these conditions. The bottom right trajectory is the application of our lateral control law with $\eta^{\star} = 0$. The target was static and located at 400,0. The FOV of the aircraft was such that offered visibility only when the target was situated at the front 180 degrees of the aircrafts.

topilot, that regulates the speed, altitude and heading variables (see appendix A). Besides flight control, our software is run on a ground laptop, linked to the UAV autopilot by a serial modem.

Once airborne, the UAV is put under the autopilot guidance, and set to fly at a $h_0 = 80m$ altitude, at a cruising speed of $10m/s$. Figures 4.12 and 4.13 show the resulting trajectories resulting from the tracking of a fixed target. Even with quite important wind speeds (up to $7m/s$) , the UAV performs well, achieving a visibility ratio of up to 70%.

Fig. 4.14 plots the logged visibility ratio and UAV/target distance. One can notice that in the first 500 seconds the overall visibility ratio is sensitive to whether the drone is approaching or not the target. This effect is due to the shape of the UAV's FOV. The specific FOV can see an object only if it lies in front of the UAV's path. However, as time passes by, these effects do not influence the overall average. This is because of the nature of the slow approaching - fast moving out nature of the algorithm.

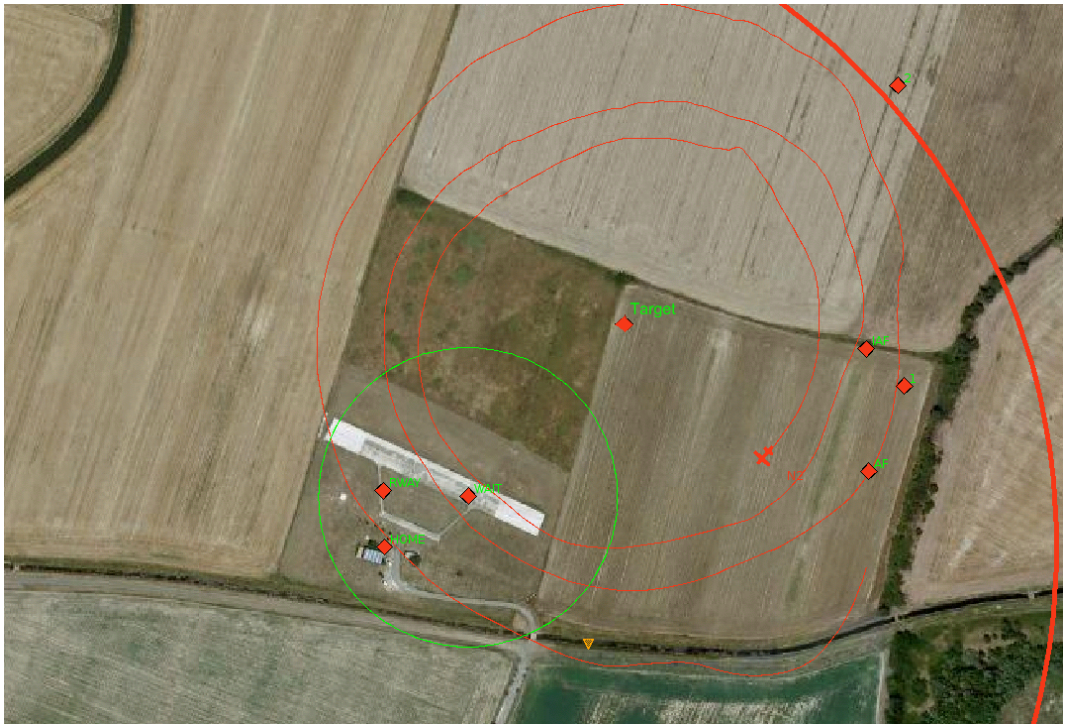

Figure 4.12: A static target tracking trajectory over our test field. The target is static sitting in the middle of the field. The red dots at the right part are rendez vous point that help initiate the experiment.

4.5 Conclusions

We presented a control strategy that allows a UAV to maintain good visibility of a ground target [Theodorakopoulos and Lacroix, 2008]. Stating the problem with simple geometric equations, our approach defines an optimal heading which is tracked using a lateral control law. The approach yields a good visibility ratio in poor sideways visibility conditions. Simulations compare the result with previous contributions, and actual flight tests showed that it

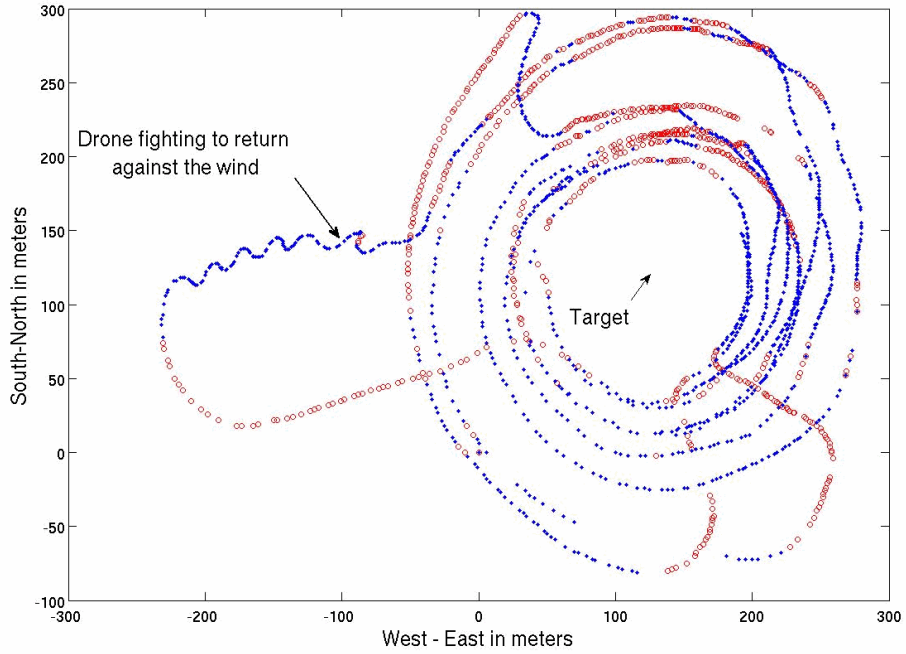

Figure 4.13: An other static target tracking trajectory, in more windy conditions (wind speed attained up to $7m/s$). The trajectory is plotted in blue when the visibility is ensured, with red dots otherwise.

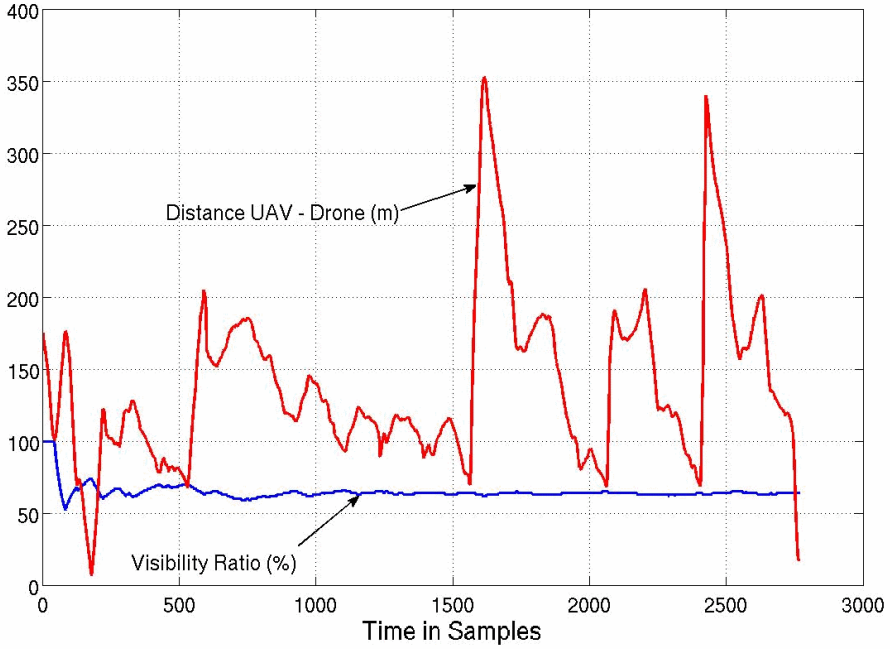

Figure 4.14: Visibility-ρ during a spiral real flight test

works even in the presence of unmodeled phenomena.

This work relies on the assumption that the target is not trying to evade the UAV. We will discard this assumption, and aim at tackling the problem within a pursuer / evader framework.

Chapter 5

An adversarial iterative predictive approach to target tracking

The work in this chapter is inspired by the different predictive and iterative solutions presented earlier in sections 2.3 - 2.5. This chapter adapts and slightly changes those ideas so that they can be used for UAV / evasive ground target tracking applications. However, the aforementioned approaches often demand an analytical preparation using Lagrange multipliers to find the optimal solution. Our conviction is that when one has to deal with hide and seek games, this analytical step may not be necessary. One can instead use classical iterative search methods to find a near optimum solution that can perform successful hide and seek games in real time and then apply this type of approach to UAV ground target tracking.

While classical pursuit-evasion approaches are based on an agent approaching very close to another agent in order to achieve a capture, the case examined here is slightly different: the goal is to place the pursuer UAV in a good visibility position. Inversely, the evader is trying to hide itself in areas that are hidden by occluding obstacles. Such a decision depends on the position of those obstacles and the shape of the UAV's FOV: a UAV that has a camera with only a limited - side FOV view will approach the target very differently from a UAV that has a full FOV. While the case of a limited FOV has been studied in some theoretical based approaches [Gerkey et al., 2006, Bopardikar et al., 2007], it has not been used in predictive approaches.

Furthermore, a predictive method can also be used to attempt and guess the future trajectory of the target and thus anticipate its position, avoiding thus configurations that could cause a loss of target visibility to the pursuer. Lastly, one can expand this type of methods to include a second UAV into the loop, something that can enhance the target's visibility ratio.

5.1 Problem Statement

In this chapter we will consider a UAV that flies at a steady altitude with a steady speed and attempts to track an evasive ground target. Naturally, the target will attempt to use any ground obstacles to hide from the UAV. All ground obstacles are considered to be domes of a fixed radius. Furthermore, the UAV will have to respect the presence of certain NFZs, which it will have to avoid at all costs. The UAVs are considered to have a sphere visibility FOV (full FOV up to a certain maximal distance).

The target is considered under two different behaviors: either moving freely in an open

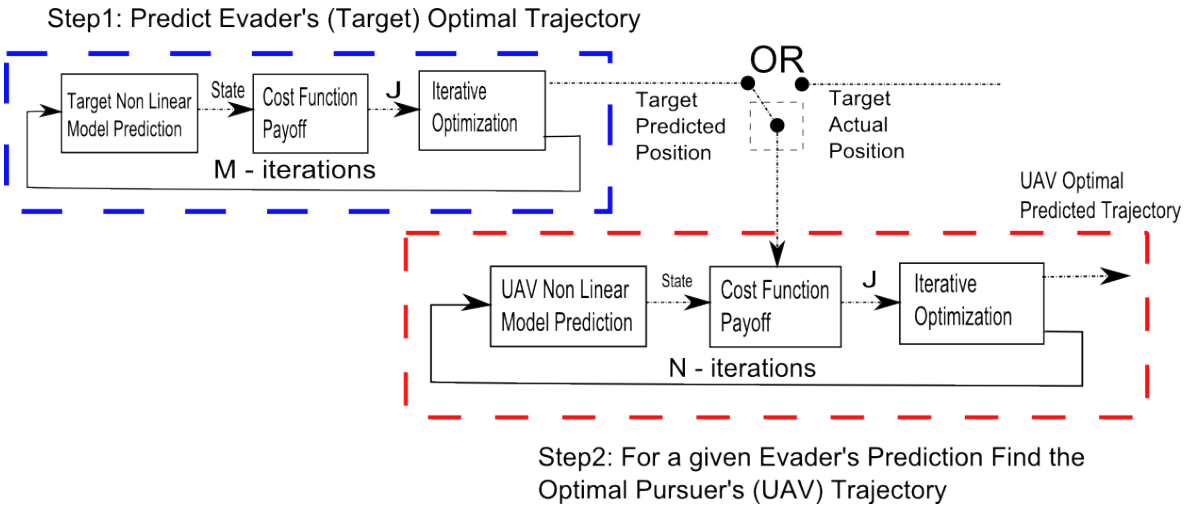

Figure 5.1: The algorithm is a two step process. In the first step a prediction of how the target should behave is calculated and according to that information the second step is initiated. During the second step the UAV chooses the optimum path according to the predicted target trajectory and according to a series of criteria that define the cost function. Both steps make use of an iterative optimization scheme where different paths are compared in terms of payoff value (cost function).

area or moving in an urban along a local road network. This information is important to predict the position of the target as well as commanding the target when designing a *hide and seek* simulation.

Finally, during the second part of the chapter, the case of two cooperating UAVs is considered.

5.2 Overview of the approach

The basic approach consists of a two step loop (Fig. B.8):

- *Target prediction:* The UAV tries to predict what will the target's path be in the near future (*the prediction horizon*), or it uses its actual position (estimated or well known),
- *UAV prediction:* During the prediction horizon, the UAV must choose a future path that maximizes the target capture, respecting the presence of restricted no-fly zones (*NFZs*), and optimizing an observation distance.

This is achieved by assigning a *cost value* J_p at each candidate path, that is produced by integrating the costs $J_i(x_i^{UAV}, x_i^{target})$ of each position along the path computed on the basis of the relative UAV/target positions (*a configuration*): $J_p = \sum_N J_i$, where N denotes the number of configurations that constitute a path – the prediction horizon. Each configuration cost naturally takes into account the target visibility and the obstacles for the UAV. This is done with the use of specially designed *potential functions* that encode all mission related constraints and criteria – distance, visibility and obstacle avoidance.

However, it is numerically daunting to calculate all possible paths and costs in real time for any useful horizon. Hence, the optimization method must include a way that will permit the UAV to calculate and select a good path without having to calculate every single path.

We propose two methods here for deciding that path:

- *Greedy method*: This approach projects different paths that use the same bank angle ϕ all along that path. The approach is named greedy because the same command ϕ is used throughout the entire path. After all paths have been examined, the process returns the path with the least cost J.
- *Informed Search method*: This approach considers all possible paths as if they belong to an imaginary graph. Each vertex of that graph is a possible position of the UAV in space and it has a cost value J. A path on that graph has a total path value equal to the sum of all the cost values of all vertexes on that path. Instead of using the same command ϕ for the entire path, we use a different command at every node. The method then returns the value with the best cost value J.

5.3 Evaluation functions

In this section we are using the term agent to describe either the UAV or the target. Their position and pose are known as the state x_k of the agent at moment k. Depending on how the surrounding objects and agents are positioned in relationship to this agent, one can assign a numerical value at that configuration. This configuration cost is known as $J(x_k^{UAV}, x_k^{Target})$.

Now, depending on the way that one chooses to assign values to configurations, one can guide the agents to pursuit certain configurations, while avoiding others. This type of encoding can promote any kind of behavior simply by directly rewarding the desired configurations and by penalizing the opposite ones. A smaller value will result to a better visual tracking.

5.3.1 UAV Evaluation Function

In this subsection, we will present the different elements one may want to control during the flight of a target tracking UAV. The list of UAV evaluation functions is:

- NFZ avoidance evaluation function.
- UAV / Target range evaluation function that maintains the UAV at certain distances from the target.
- Shadowed areas avoidance evaluation function. This function helps at maintaining the UAV away from zones that the target cannot be seen.
- Correct FOV orientation. Function that is useful if the UAV does not have a full FOV.

No-Fly Zones

We saw earlier that NFZs are modeled as a collection of cylindrical obstacles. In order to guarantee that the UAV has enough space to reverse direction in front of any obstacle configuration, we use a *Minimum Critical Distance* R_{cr}. The value of this distance is defined by the UAV dynamics, and acts as a safety blanket surrounding the obstacles (Fig. 5.2).

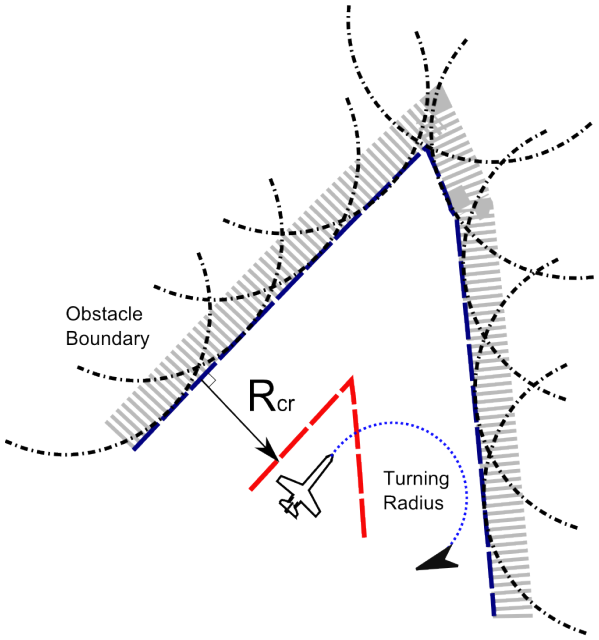

Figure 5.2: A set of circular obstacles is used to represent a dead-end for the UAV. The distance R_{cr} is set so that the UAV can perform a 180°-turn when facing any obstacle configuration.

Avoiding NFZs is achieved by forming a repulsive field around each object (Fig. 5.3). The equation that describes such a repulsive field is the following:

$$J_o = \sum_i ((max(R_{cr} + R_i^{obs} - d_i, 0))^2) \tag{5.1}$$

Where d_i is the distance between the UAV and the obstacle i, and R_i^{obs} is the radius of obstacle i.

UAV / Target range

The relative distance between target and the UAV is quite important and should be kept within certain limits: a) If the UAV flies too far away from the target, then visibility degrades seriously. This distance is known as R_{max}. b) If a limited visibility UAV flies right above the target position there is a risk that it will be outmaneuvered by the target. This is due to the greater maneuvering capability of the target that can alter directions easier than the UAV. A ground vehicle can stop and change directions at any moment. As a result when the target is right under the UAV all it takes to lose visual contact is a sudden target deceleration and the UAV will fly over it, completely losing it from sight (Fig. 5.4). The minimum distance is known as R_{min}. The cost function that describes this can be seen here:

$$\kappa = max(d - R_{max} + \epsilon, 0) \tag{5.2a}$$
$$\lambda = max(R_{min} - d, 0) \tag{5.2b}$$
$$J_e = max(\kappa, \lambda) \tag{5.2c}$$

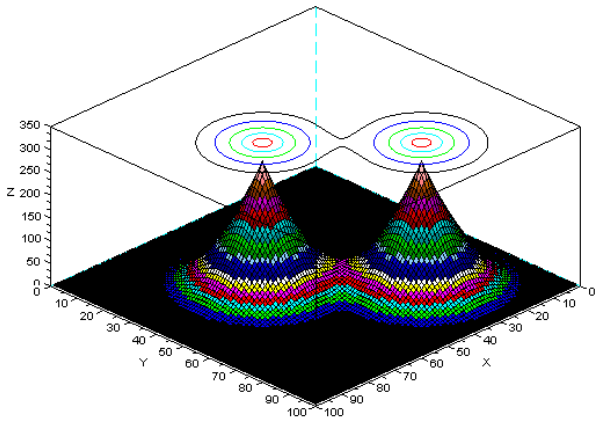

Figure 5.3: The result of using an evaluation function in the presence of a double obstacle.

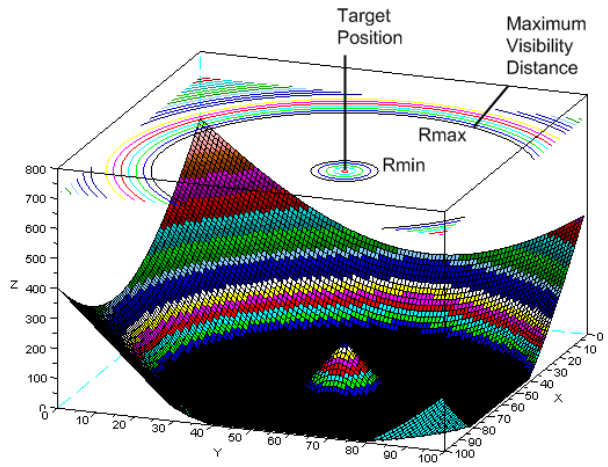

Figure 5.4: Graph of the evaluation function element that keeps the UAV within a certain area around the target but not over it.

Where d is the distance between target and UAV and ϵ is a small constant that when used with R_{max} it makes sure that the UAV changes position before it arrives at the limit distance R_{max}.

Avoiding zones that do not permit a direct LOS with the target

The ground surface is rarely flat and obstacles are common. Hence, there is a need to know when a certain configuration puts the target in a zone that cannot been seen from some areas in the sky, because of some ground obstacle. This is achieved by knowing the exact positions of the agents, the obstacles, the size of the obstacles. By using a simple approximation one can calculate the position of the center of the shadow and thus deduce whether or not the

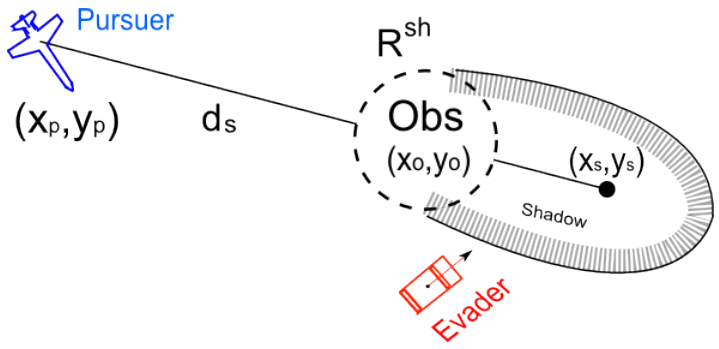

Figure 5.5: This diagram shows a simple way to approximate the center position of the shadow.

target is within that zone.

The exact shape of the obstacles is not important as we only want to quantitatively calculate the center and size of the shadow. Shadowing or occluding obstacles are represented on the ground (Fig. 5.5) as domes of given radius. The idea behind this method is that it is not necessary to know the 'exact' position and geometry of the shadow, just a simple approximation will be adequate to reproduce the behavior of a shadow-seeking agent.

The position of the center of the shadow can be calculated by considering that a point is between two other points. If one then considers that the point in between is closer to one of the two points, one can arrive at the following formulas:

$$\frac{c_1 x_s + x_p}{c_1 + 1} = x_o \tag{5.3a}$$

$$\frac{c_1 y_s + y_p}{c_1 + 1} = y_o \tag{5.3b}$$

Where (x_s, y_s),(x_p, y_p),(x_o, y_o) are the coordinates of the shadow center, the UAV's and that of the obstacle's respectively.

The above set of equations can easily be transformed to:

$$x_s = \frac{c_1 + 1}{c_1} x_o - \frac{x_p}{c_1} \tag{5.4a}$$

$$y_s = \frac{c_1 + 1}{c_1} y_o - \frac{y_p}{c_1} \tag{5.4b}$$

Where c_1 is used to fine tune the simulated distance between the obstacle and the center of the shadow. By trial and error we found that a value like $c_1 = 3$ is satisfactory. Adding the point $\{x_s, y_s\}$ as a repellant to the potential field makes sure that the agent avoids areas from which the target cannot be seen.

$$J_{sh}^d = \sum_i max(R_i^{sh} - d_i^{sh}, 0)^2 \tag{5.5}$$

Where d_i^{sh} is the distance between the ith shadow center point $\{x_s, y_s\}$ and the target, while R_i^{sh} is the radius of the ith shadow that is typically connected to the obstacle size and here is considered as a constant. It should be noted here that the position of the shadow

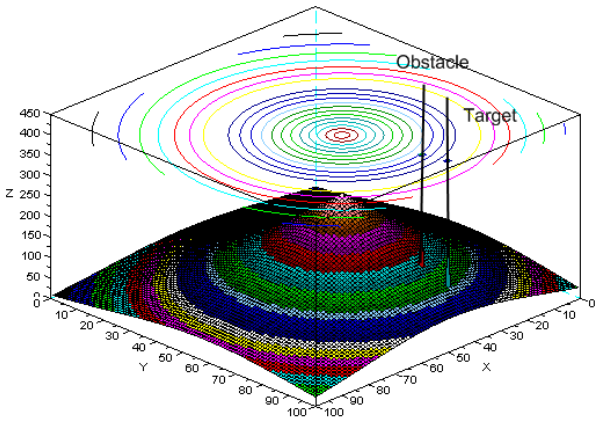

Figure 5.6: The area to be avoided by the UAV is the area most probable to get obscured by the obstacle. Notice that we have simplified the calculations of the shadow by creating a circular potential field around the center point of the shadow. The alternative would be to construct a more complicated potential field that would calculate the exact shape of the shadow. This is not necessary as the agents still move towards the directed area.

depends from the positions of the obstacle and the UAV. As a result the position of the shadow changes as the UAV changes positions. By measuring the distance d_i^{sh} we know how far is the target from the shadow center.

Correct FOV visibility orientation

Not all UAVs have the capability of a full FOV. If that is the case then some special care must be taken to ensure that the UAV will have a correct orientation. The basic idea is to ensure that the target remains as close as possible to the center vertical line of the camera image. If for example the aircraft has a static camera that is positioned sideways, then the cost function will always penalize the aircraft poses that do not have a target at the side of the aircraft. The cost function that describes this configuration is:

$$J_f = \cos^2 \theta_f \tag{5.6}$$

Where θ_f is the horizontal angle difference between the camera axis and the target bearing. As a side note, it should be noted that we consider a camera that has at least the capability to pivot in the aircraft roll axis, and thus offsetting the rolling angle of the UAV.

5.3.2 Overall path cost

UAV path cost integration

At every iteration a cost function is calculated. The overall cost value for that specific path is the sum of all the cost values of every point in that path as it is given by the following

equations:

$$J_i^d = w_e J_e + w_o J_o + w_f J_f + w_{sh} J_{sh}^d \tag{5.7}$$

$$J_p = \sum_{i=1}^{N} J_i^d \tag{5.8}$$

Where N is the number of nodes in that path p, w_e, w_o, w_{sh} are weights that are selected in order to put higher priorities in some mission criteria than others i.e. NFZ avoidance. J_e is the evaluation function that permits to guard the distance between the UAV and target within a given range $R_{max} - R_{min}$, J_o is the evaluation function that keeps the drone in a safe distance from NFZs, J_f is the distance that forces to the UAV to have a correct coordination towards the target and finally J_{sh}^d is the evaluation function that assures that the UAV will stay away from configurations where the target is behind of some ground obstacle.

Target path cost integration

The same process takes place for the case of the off-road target. Here are the equations:

$$J_i^t = w_{in} J_{in} + w_{go} J_{go} + w_{sh} J_{sh}^t \tag{5.9}$$

$$J_p = \sum_{N} J_i^t \tag{5.10}$$

History Horizon Considerations

Finally, we have been experimenting with multiplying each cost J_i with different weights according to the depth of the prediction. This means that a prediction that is closer to the present moment will have greater influence to the overall cost function than another prediction that has been made more far in the future.

$$J_p = \sum_{N} w_i J_i \tag{5.11}$$

Where N is the length of the overall path in steps.

$$w_i = \alpha * w_{i-1} \tag{5.12}$$

and $\alpha \in [0, 1]$

Such a scheme can make the behavior of the UAV more elastic, something that gives higher value to obstacles that are closer to the agent.

5.4 Prediction and Iterative Optimization

5.4.1 UAV prediction

The goal of this step is to find a series of bank-angle commands that minimize the cost function J_p for a given prediction horizon. Once that path is found the first step of that path is passed as a command to the UAV and the whole process is repeated. For an on board UAV

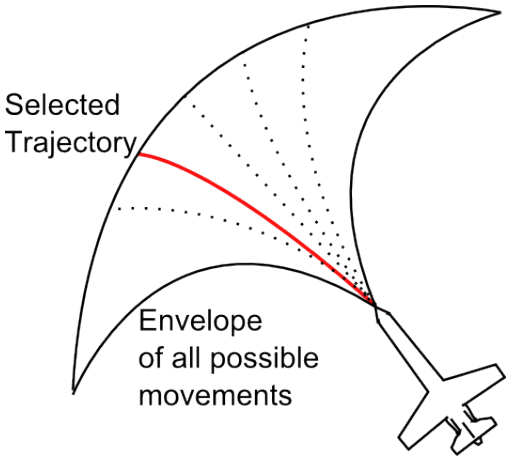

Figure 5.7: The Greedy Method: Different trajectories are projected in the future and the one with the least cost is selected

computer the task of finding that optimal path may be costly in terms of calculations. If the decision-action cycle is not fast enough, the UAV may start to oscillate or even worse, it may enter some NFZ. For this reason all calculations must remain as simple as possible.

At every moment the UAV discretizes the command configuration space $[\phi_{min}, \phi_{max}]$ in D discrete commands and it does that for a horizon of N steps. To calculate all the possible configurations becomes a problem of $O(D^{N+1})$ complexity.

The method includes two steps: a) First, we use a Greedy method to predict the position of the target for a horizon of M steps from the present moment and b) We use that predicted position to find where the UAV should go and we do that using one of the two proposed methods.

Greedy approach

According to this method, we discretize the roll command $[\phi_{min}, \phi_{max}]$ in D different intervals. We then predict the cost function along the path of these D commands. This is done for N steps of a fixed time interval δT. In other words, the prediction horizon is N X δT. Each such path is examined and its cost value J is noted. We repeat this process until all D different values of roll have been examined. The algorithm then returns the path with the least cost J. We use thus D different values of ϕ. These values remain the same throughout each of those paths. This method is the simplest and fastest in terms of calculations but it has a limitation in terms that the same command is applied all along the path, lacking thus some flexibility. (Fig. 5.7)

Informed search approach

A more rich path finding scenario would be to use a classical path finding algorithm like A* (Sec 2.2.2) . It should be reminded that there is no fixed end node. Instead, all that is known is that the prediction will move to a depth of N iterations. The solution proposed here is to set a imaginary end node that is connected to all end points with a *zero cost function path* which permits the algorithm to run Fig.5.8.

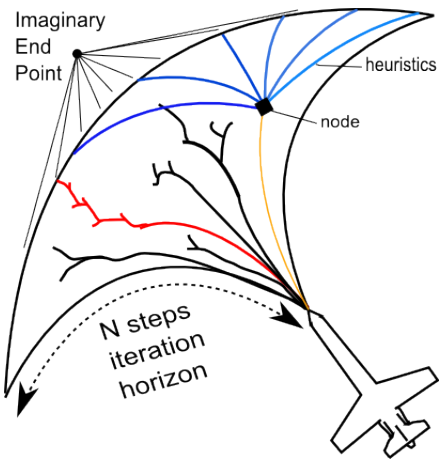

Figure 5.8: The A* Method applied to UAV path planning.

As a heuristic, we use the Greedy algorithm with some of the criteria of the cost function relaxed. This is to ensure optimality. According to [Russell et al., 1995] in order for A* to be optimal the heuristic must be more optimistic than the actual cost function of the path. In this case, we decided to form a cost function with very few criteria: obstacle avoidance, correct FOV orientation and shadow visibility.

Let's imagine that A* is currently at a specific prediction depth and wants to predict up to a given future prediction horizon. In order to do that it will keep moving along a path while examine nodes on that path (Fig.5.8). Every node it meets it evaluates the value of the cost to get to that node plus the heuristic of what it will cost to get to the end of the prediction horizon.

5.4.2 Target Prediction

The ground target model is used as a prediction for the future trajectory of the target. There are two main models that are in use here: a) A target model that moves within the road network, following strictly the geometry of the road and b) A model that evolves on an open flat surface and has thus much more freedom of movement than the previous model. The first model evolves only forward with different speeds and always within the road geometry, while the second model moves on a flat surface and closely resembles that of a Dubins car. For our simulations the second model (*Off-Road*) uses the same greedy iteration mechanism as the UAV to choose its path.

This part is based on the assumption of *game-theoretically rational players*, which means that the target will always try to move in such a way that it will avoid capture. This is related to the *minimax algorithm*, where the UAV selects the action that yields the best results in terms of cost function optimization, given that the target will do whatever is possible to avoid capture.

Overall, we study three types of targets:

- A target where no knowledge exists apart from its current (estimated) position.
- A target that moves on a Road Network (Urban and Semi-Urban environment).

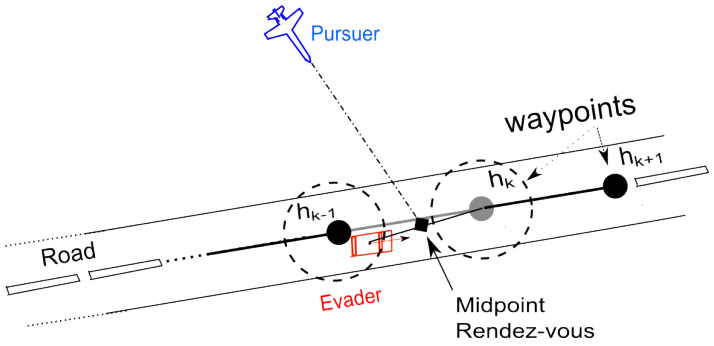

Figure 5.9: *Midpoint rendez-vous*: In order to be able to anticipate the movement of a moving target on a road, the UAV aims at the point between the future waypoint (h_k) and the actual position of the target. When the target, gets closer than a certain margin distance from a waypoint, that waypoint is considered as the current one until a new waypoint is approached. This way, the future position of the target can be predicted without having to estimate the target speed.

- A target that moves on an open space (Off-road vehicles) in the presence of ground obstacles.

I) A target where no knowledge exists apart from its current (estimated) position.

This is the simplest case, where no prediction is made and the UAV simply reacts to the decisions of the target without any power of anticipating its movements.

II)) A target that moves within a Road Network (Urban and Semi-Urban environment).

This type of target, constantly follows the road and the only decision it has to take is which road it will follow when it arrives at a crossroad. In order to achieve this, it is already assumed that the road network is known a priori and it can be discretized in a series of waypoints. The whole structure is actually a graph that has two types of waypoints:

a) *Simple-Waypoints*, which are connected to only two other waypoints, one before and one after.

In order to be able to anticipate the position of the target, we use as coordinates the method of *midpoint rendez-vous*: at any given moment, the UAV aims at the midpoint between the target's actual position and that of the target's following waypoint. This way we do not need to estimate the speed of the target: If the target decides to stop, then the midpoint will also stop, while the UAV will continue to close the distance, relieving us from the burden of estimating the speed of the target. When the UAV finds that the target has approached close enough to a waypoint, the next waypoint is fetched from the list and the process starts all over again. This idea is related to missile *Deviated Pursuit Guidance* as it was seen in 2.2.2 (Fig. 2.2(b)), where the missile aims at a point in advance of the target current position, as well as to the missile *Three Point Guidance*(Fig. 2.2(c)) , where the missile aims for a point between the launcher and the target.

b) *Decision-Waypoints* (Fig. 5.10), which are connected to three or more other waypoints.

The way a decision is made is by looking which path is most distant from the UAV and uses that knowledge to understand which side of the obstacle the UAV resides. As there will be only one side of the obstacle that is not visible by the current UAV position the target then chooses that side. The question of whether the target will get to the shadow before the

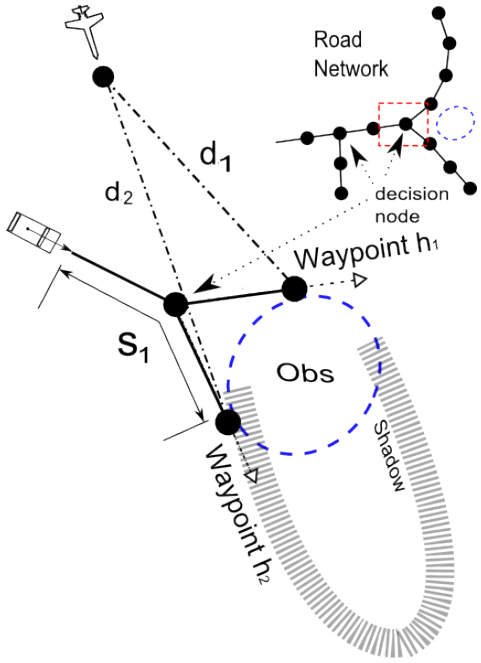

Figure 5.10: *Decision-Waypoints*: The target is trying to decide which path to follow (h_1 or h_2). Its decision is based on which side is more possible to provide a hiding area. The way it does that is by finding out which of the two paths is the most distant for the UAV to reach.

shadow moves depends only from the difference of distances d_2 and s_1 and the speed of the two agents.

If no obstacle is close to the crossroad then the target will still select the path that is the furthest from the target, hoping that it can evade capture by adding more distance between the target and the UAV.

$$h^i_{dec} = max(d^i_2, d^i_2) \tag{5.13}$$

Where i is the ith node-waypoint where a decision needs to be taken. This type of navigation is known as *Stealthy Navigation* (Fig. 5.11).

It is clear that the above approach does not assure success. How can one know how exactly a human driven car will behave under such circumstances ? No user studies on this matter exist, so we can only make assumptions. One can hope that the UAV can be positioned in such a way that visibility can be assured whatever the choice of the target is. This is one of the approaches used in the next chapter. However, the case that was examined in this chapter is that if the UAV makes the opposite decision from the target and take the other route than the one chosen by the target, a visibility loss will occur. Under this assumption the only model that remains is that of a rational player that always tries to hide behind an obstacle whenever it has the chance.

III) A target that moves in a open space (Off-road vehicles).

For this kind of target, the same greedy algorithm that decides the path of the UAV is used. The target evaluation function has been presented in an earlier section. Here the target is considered to move with a steady speed and in some simulations would either move or stop.

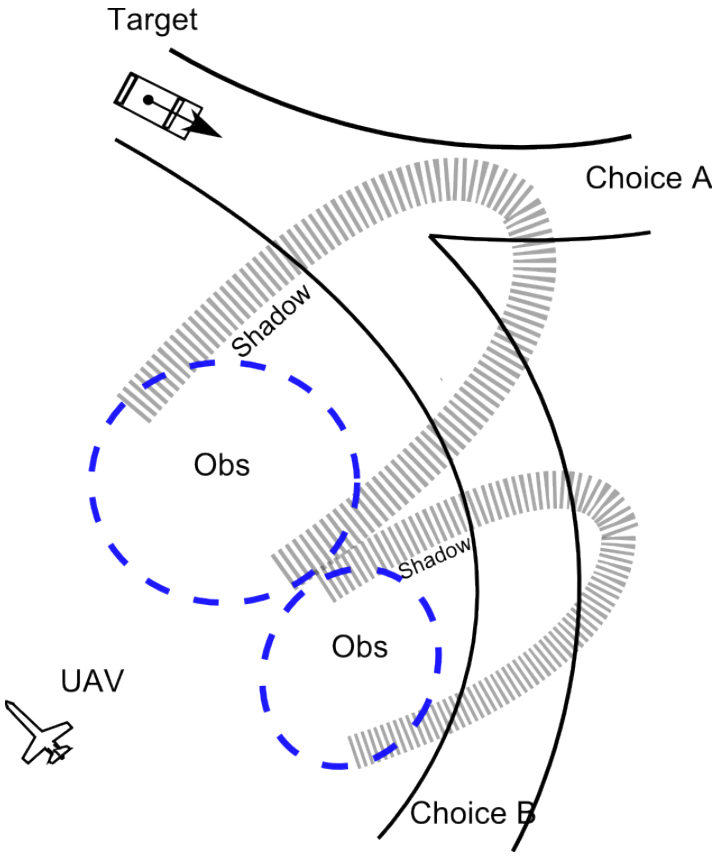

Figure 5.11: An example of *Stealthy Navigation*: A target moving along a road network arrives at a crossroad. There it has to choose between route A and route B. The latter being more 'stealthy' will be considered to be its final choice and this will help the UAV to anticipate it's movement and make sure it has visibility before the target hides behind the two obstacles.

5.4.3 Off-road target evaluation function

When the target is moving in an open terrain, its position can be predicted using the same approach as in the case of the UAV. Hence, the target evaluation function shares many common elements with the UAV. However it should be noted that the use of a target evaluation function is *only for the case of a target that doesn't use a road network* as the case of a target that moves on a road network can best be described using graphs as it will be shown later. This it is because a target that moves in a road network does not have the option to exit from the road network. Once on the road network it will behave in very predictable manners e.g. Drive straight in straight highways, turn in crossroads etc. The list of target evaluation functions is:

- Ground obstacle avoidance evaluation function.
- Shadowed areas attraction evaluation function. This function forces the target to move into areas that are hidden from the UAV point of view.
- Direction inertia. This maintains the movement of the target towards the same direction it was moving earlier.

Ground obstacle avoidance

Avoiding ground obstacles is done also using the same type of formula used in UAVs for NFZs by forming a repulsive field around each ground object:

$$J_{go} = \sum_i max(R_{cr} + R_i^{obs} - d_i, 0)^2 \tag{5.14}$$

Where d_i is the distance between the target and the ith obstacle while R_i^{obs} is the horizontal radius of ith obstacle.

Attraction towards zones that do not permit a direct LOS with the UAV

Here, the target uses the different shadow coordinates and selects the ones that are the closest to the moment of the decision. It then moves towards that shadowed area. The equation that attracts the target to the shadow is:

$$J_{sh}^t = max(d_i - R_i^{sh}, 0)^2 \tag{5.15}$$

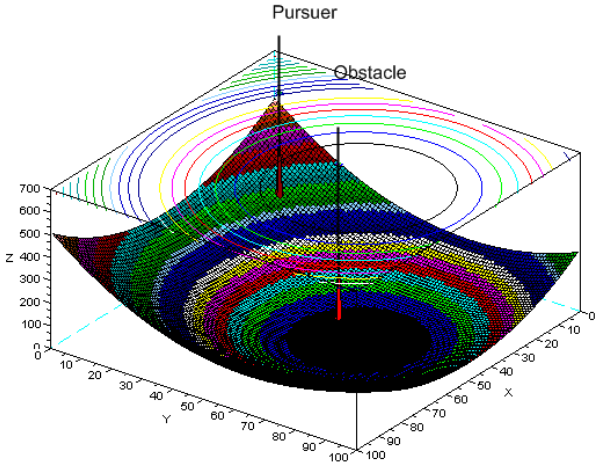

Figure 5.12: The shadowed area is the area that the target is attracted the most.

where d_i is the distance between the closest shadow zone $\{x_s, y_s\}$ and the target, while R^{sh} is the radius of that shadow that is typically connected to the obstacle size and here is considered as a constant.

Direction inertia

This element maintains the speed towards the direction that the target was moving already. In other words, it encodes a kinematic constraint and this is done to ensure that the target prediction will maintain the same direction as previously.

$$J_{in} = |(\theta_t(k) - \theta_t(k-1))|^2 \tag{5.16}$$

Where θ_t is the target direction.

5.5 Results

For our application, the most important criterion seems to be *what percentage of the overall time T_{total} does the UAV keep the target within the capture zone.* Therefore, we use an additive criterion that increases when there is a visibility loss and diminishes when visibility has re-established: The smaller the value of the cost, the better was the algorithm in terms of visibility and obstacle avoidance.

This kind of payoff is seen only in *games of degree* where the game ends with a score, as opposed to *games of a kind* where the payoff is Boolean (capture or not capture). Furthermore, special penalties are established, for the moments where the UAV approaches too close to a NFZ.

5.5.1 Evaluation of optimization methods

In order to evaluate every algorithm, every scenario was left to unfold for a fixed time T_{total} and then the overall score V_{is} of that run would be noted down.

This can be calculated either using a length ratio or a time ratio:

$$V_{is} = 100\frac{T^{No}_{Capture}}{T_{total}} + \sum Flags$$

Where $Flags$ is a penalization for every time the UAV comes too close to an obstacle, $T^{No}_{Capture}$ is a time interval where the UAV had lost visibility with the target and T_{total} is the total duration of the test run. The lower the ratio $V_{is} \in [0, 1]$ is, the better the algorithm is.

Preliminary Simulations

We conducted a first simulation to see qualitatively how the algorithm behaves in a very simple environment where the target and UAV are contained within a small area. In a first series of simulations, the target was ignoring the presence of the UAV and it was moving with a steady speed in a large circle. The UAV had to maintain visual contact with the target by being within the radius of a hundred meters, while avoiding obstacles and making sure not to fall within the shadowed region of a fourth, visibility occluding obstacle.

For 90 min of simulated flight, the A* based algorithm scored 142 units which was far better than the Greedy algorithm that scored 245 units. As a side note, however, we should say that the A* really pushed the capabilities of our machine and the decision branches of every node had to be cropped to 6, when in the Greedy case were around 40. As a result, the movement of the aircraft was more abrupt and aggressive in terms of maneuvering. If we tried to use more than 6 child nodes, the code would become slow and the trajectory would oscillate as delays would appear in the decision process.

It is understood that the maximum prediction horizon for such a system can be calculated as follows: in order for the system to remain stable it must be that $T_{calc} < T_{cr}$, where T_{calc} is the calculation time and T_{cr} is the system critical period that guarantees stability for the aircraft. The worst case scenario for A* is that it will open all nodes in its path. This produces a time complexity of $O(b^d)$ where b is the branching factor and d is the depth of the search - the horizon depth in simulation steps. So for the worst case scenario if it takes T_e to examine a node then it will need $b^d T_e$ seconds to examine a certain depth. Hence, from the above formula one can infer that:

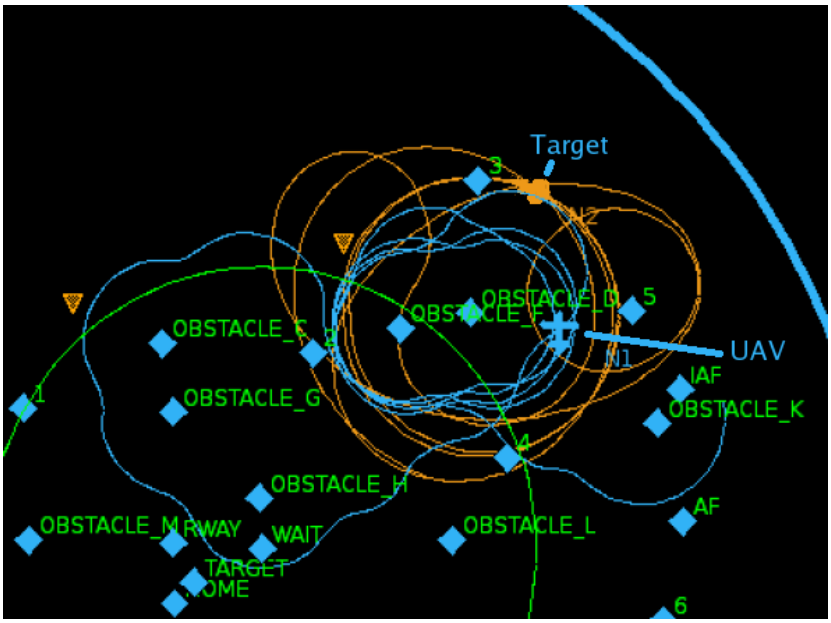

Figure 5.13: The behavior that systematically emerged in the off-road target scenario, was a hide-and-seek game around the shadow-casting-obstacle (*Here, obstacle D*) that at some extreme cases would stabilize in a circle, with the target hunting the shadow and with the UAV hunting the target. Both UAV and target where obliged to stay within a virtual area of about 300 meters radius. The pre-conditions that can trigger such a behavior can be either algorithmic or speed/distance related. The algorithmic conditions are related to whether the target is seeking a local solution or a global one. A global solution would make the target switch obstacle, something that would brake the pattern. On the other hand the pattern could brake if the UAV was close and fast enough to capture the target.

$$d_{max} = \log_b(\frac{T_{cr}}{T_e})$$

Where d_{max} is the maximum number of iterations that the graph search can search.

Simulations

In a second series of simulations, the target actively tries to avoid the capture whenever this is possible. The following cases are examined:

- *Greedy VS Informed Search Approach*
- *Target Prediction VS No Target Prediction.* In the first case the drone attempts to predict the position of the target while in the second it would use the actual target position.
- *Off Road Target VS Target on Road Network*
- Different Road Complexities (number of bifurcations)

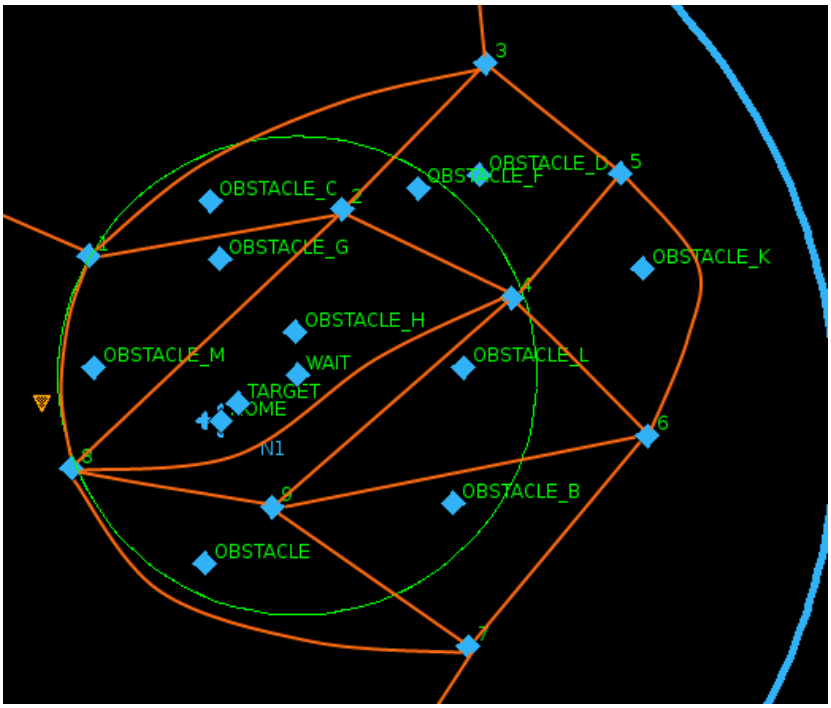

Figure 5.14: Graph showing the complex road network. The target was moving on that network changing directions without notifying the UAV, which was trying to anticipate the target's next move at every moment.

When, the target moves in a road network it often comes to crossroads. When this happens it has to face two or more possible routes. Most of the times an obstacle will be between the two routes. If the target moves on path A while the UAV moves on path B then a visibility loss will occur as the obstacle will be between them. As a result its score will deteriorate.

In order to observe the behavior of the different algorithms in various road complexities 5.14), we use three types of road networks with different complexities (Tables 5.4). By complex, we mean that the bifurcations and obstacles vary, obliging the UAV to take more decisions as to where the target may go next. More decisions mean a greater possibility for errors for the UAV and thus making it more difficult to maintain visibility throughout the experiment.

As can be seen (Tables: (B.1),(5.2),(5.3)), in a general manner the A* algorithm dominates the Greedy algorithm. Similarly, the algorithms that use a target prediction step behave better than the algorithms that simply use the actual position of the UAV as input, if the target is moving on a road network. However this comes with a cost, as 'better' algorithms tend to use more computer resources.

An interesting feature is that, as the road complexity increases so does the difference between the score of the algorithm that uses a target prediction step to its counterpart that does not use a target prediction step. The reason for this is the following:

Every time the UAV fails to follow the target at a crossroad, especially in the presence of an obstacle that casts a shadow, it suffers a temporary loss of visibility. An algorithm that does not predict how the target will move next, is much more probable to take wrong decisions of that nature. Now, if that happens during a long simulation, errors and visibility

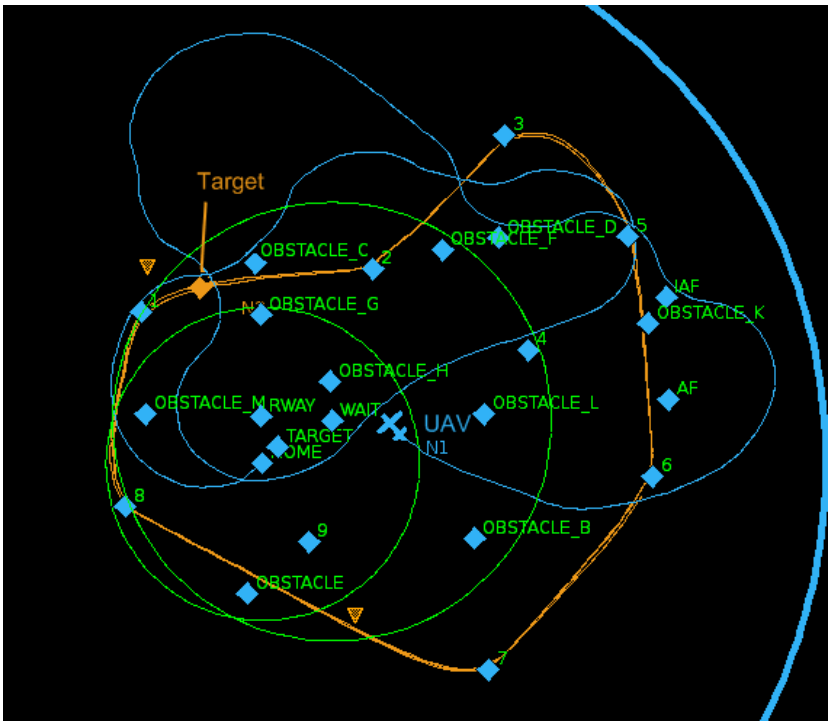

Figure 5.15: The UAV pursuing the target on a road, using the greedy method.

losses tend to accumulate, giving poorer results than its target predicting counterparts.

One can also see that when the system predicts the target behavior it gets much better results than when it simply inputs the current target position. This should not come as a surprise as in the first case the UAV has more time to maneuver into position and thus can track the target for longer intervals.

However, it becomes clear that for the case of the off road moving target the behavior shows the opposite signs: no-prediction works better than prediction. This happens because predicting the trajectory of an off road target is much more difficult than the case of the road network moving target. As a result, moves in the wrong direction quite often and this costs visibility as it takes time to get back in visibility distance.

5.6 Extension to the multiple UAV target tracking case

There is no doubt that however effective mono-UAV strategies may be, there will always be some configurations where visibility is lost. It is also very straightforward that the introduction of one or more UAVs will increase the target visibility. This section propose an extension to our approach to the case of several UAVs (Fig. 5.16).

5.6.1 Two UAV visibility

Obstacles cast shadows that diminish the total visibility area for every UAV. As a result, each UAV has a visibility area that depends on its own position. It is clear , that the presence of a second UAV can enhance the overall group visibility if the UAVs do not share the same

	Greedy (G)	A*
Road + Target Prediction	133.2	118.2
Road + No Target Prediction	167.2	145.7
Off-Road + Target Prediction	194.7	139.2
Off-Road + No Target Prediction	183.0	120.0

Table 5.1: Algorithm scores in a very Simple Complexity road network. Each simulation lasted about one hour and during that time, the UAV met about 60 bifurcations and 200 obstacles.

	Greedy (G)	A*
Road + Target Prediction	155.2	143.2
Road + No Target Prediction	205.7	206.1

Table 5.2: Different algorithm scores in a Medium Complexity road network: 120 bifurcations and 200 obstacles.

	Greedy (G)	A*
Road + Target Prediction	310.1	150.2
Road + No Target Prediction	324.3	221.5

Table 5.3: Different algorithm scores in an even more Complex road network: 120 bifurcations and 300 obstacles.

	Greedy (G) G	A*
Simple Road Network	34.0	27.5
Medium Road Network	50.5	64.9
Complex Road Network	14.2	71.3

Table 5.4: Greedy - A* Score differences for different Road Complexities

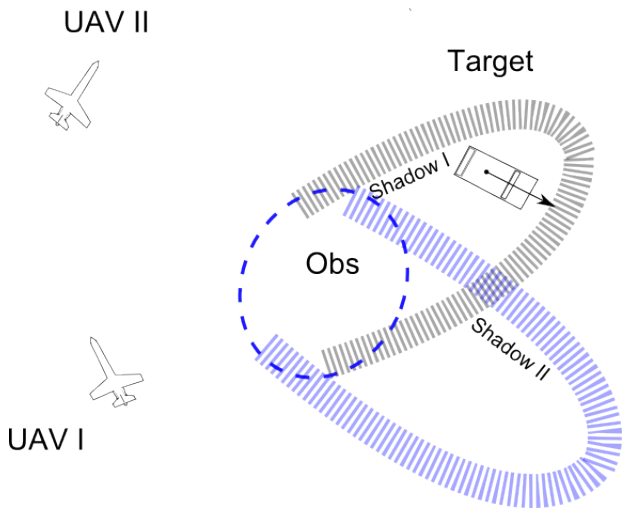

Figure 5.16: The two UAVs hierarchical scheme. Whenever UAV I is about to lose visibility with the target, UAV II positions itself in such away that will make sure it has visibility while the first one is in the shadow.

position. In the case of two or more UAVs the total visibility of the system is the Boolean OR of all individual visibility areas:

$$V_{is}^{total} = \bigcup_{i} V_{is}^{i} \tag{5.17}$$

One very basic principle of multi-UAV target tracking is that in order to maximize the overall visibility one has to make sure that the UAVs have different positions. This no-overlap principle ensures that all UAVs have a different viewing angle to the target.

5.6.2 Two UAV strategies

The approach we take is hierarchical:

- The first UAV ignores the presence of the second UAV and continues to perform as it was described earlier
- It is the task of the second UAV to anticipate the movement of both the moving target and that of the first UAV and to react accordingly. Thus the second UAVs uses the position of the first UAV and the position of target as an input, when deciding its future position. If the first UAV looses visual contact with the target for any reason, the second UAV gets in a position to continue viewing the target while the first UAV maneuvers out of that low visibility position. In a sense, the second UAV acts as a visibility safeguard, as it always tries to compensate for the loss of visibility of the first UAV.

The reason we decided to use this type of approach is because we believe that it is a good compromise between the communication load and the calculation load. The first UAV has only to find a configuration that is effective for itself while the second UAV has only to know the position of the first UAV and has to calculate its position based on the first UAV. An

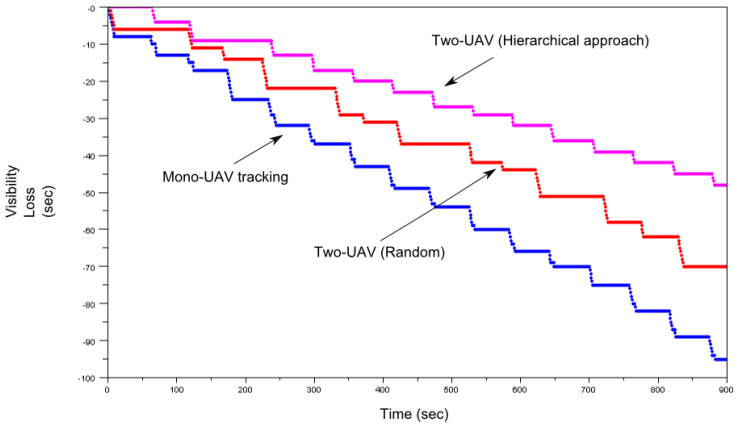

Figure 5.17: This graph shows the visibility loss of each approach vs the time of the experiment. The target was moving in a simulated area with ground obstacles. The blue graph shows what happens when there is only one UAV tracking the target and as expected, that is when the most visibility loss occurs. The red graph shows, what happens when there is one UAV actively tracking the target and a second UAV randomly moving in the area but with active target detection. As expected, the visibility loss time diminishes. Finally, one can see that when both UAVs actively track the target, the least time of visibility loss occurs.

		1 UAV	2 UAVs (UAV2: Random)	2 UAVs (UAV2: Hierarchical)
	Vis Loss (sec)	93	69	48

Table 5.5: Visibility loss time for different UAV tracking combinations: One UAV, One active tracking UAV and a randomly maneuvering UAV and finally, two hierarchically moving UAVs.

alternative would be that the first UAV has to solve a maximization problem using an optimization method and then assign the future trajectories for all UAVs. This kind of approach has been used mainly in formation flight and it has been observed [Hattenberger, 2008] that in large formations the stability of the trajectory becomes problematic. This would be because of the calculation load as each UAV should take into account the position of all other UAVs.

5.6.3 Results of the two UAV approach

A series of simulations was conducted to find out what are the effects of adding a second UAV into the loop. Two scenarios were used: During the first one a UAV follows a random trajectory in the airspace, while during the second scenario a second UAV uses the hierarchical approach described above to enhance the overall system visibility (Fig. 5.17). The results show that by adding a second UAV there is a significant improvement on the overall visibility even when the UAV is moving randomly. However, the real improvement appears when the second UAV uses an hierarchical predictive approach. The improvement is of the order of 50% for the cooperating UAVs and of 34% for the randomly moving two UAVs.

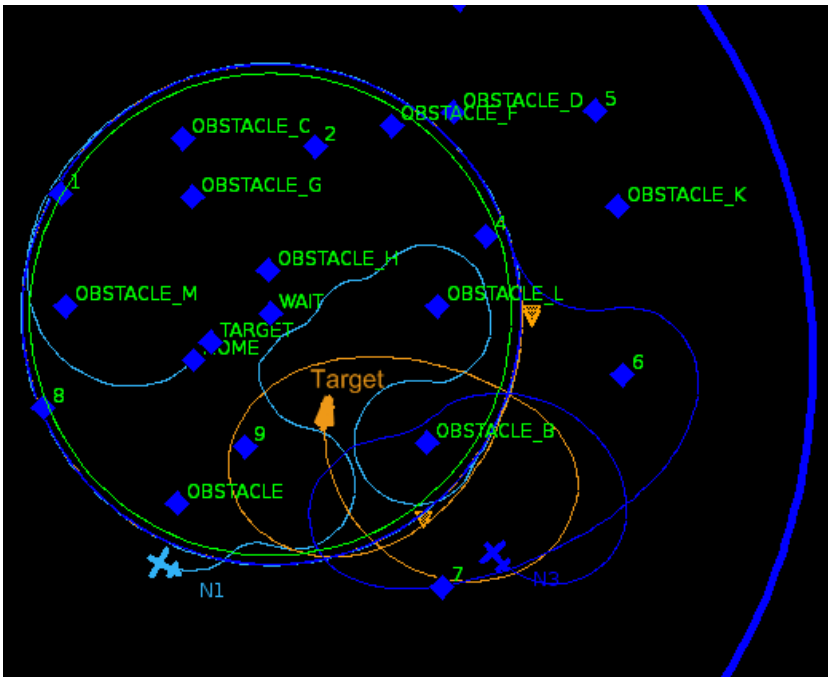

Figure 5.18: Here, two UAVs are chasing a target that is maneuvering, trying to get out of visibility.

5.7 Conclusions

This chapter presented an iterative predictive method that permits a UAV to track an evading ground target [Theodorakopoulos and Lacroix, 2009]. It uses a model of the target to predict into the future and two different methods to find the best suited route by optimizing mission related evaluation functions.

The method behaves quite well in simulations. Since the simulating tools are quite realistic (they are part of the actual flight system used to fly our real model aircrafts, see appendix A), we are confident that even if there will be an overall score quite deteriorated because of the presence of noise and delays in a real setup, one would obtain a similar behavior in real flight testing as in simulated flights.

The method has been extended to the case of two UAVs tracking a ground target. The chapter then proceeds to show how the different agents behave, using simulations. The cooperating approach is not by any means a perfect one. However, it is simple and efficient, and does not call for a centralized coordination. As a result one could add several UAVs that would be all independent and thus modular. Finally, it is clear that real flight testing would show exactly how things work. Such experiments are under way and the results should show how the method behaves in realistic conditions.

Chapter 6

A discrete game theoretic approach to target tracking

6.1 Introduction

As shown during the state of the art, game theory provides a strong framework for decision making. This chapter uses the tools provided by game theory to produce a novel approach to UAV target tracking. Before one engages in such a type of study, one should examine whether game theory is appropriate for this type of missions.

Many scholars and psychologists have argued that humans do not always behave accordingly to game theory models *i.e.* when a lighter version of prisoner's dilemma was put to the test, only 60% of the participants behaved accordingly to the model [Tversky, 2004]. However, when humans try to take decisions in risky situations, they tend to approach much more the game theoretic model [Kahneman and Tversky, 2004], with business, politics, and biology being the best examples[1]. This means that a ground vehicle driven by humans can use different trajectories than an unmanned ground robot. However, for most cases both are expected to behave (or play) rationally and *i.e.* attempt to use a road that is covered by vegetation instead of a road that is on the open and thus easily tracked from the air.

Furthermore, for many situations game theory can provide decisions that are dominant no matter what the other opponent may choose. It can analyze and decide whether a kind of play is optimal for the pursuer even if the target is not actively trying to evade. This type of decision making can put the pursuer in a good configuration without having to predict the opponent's play. This chapter attempts to provide algorithms that will permit the evader to escape, when followed by a pursuer. This combination of a pursuing UAV and an evading target will form a dual that is very frequent in pursuer-evader games.

One cannot know how much information will the evader have: does it have full information or not ? As this fact will probably remain unknown, one will assume that he knows best the areas that surround him. If for example the target has to make the choice between two areas that can provide stealthy navigation, it will prefer to use the closest one.

[1]The entire cold war era, with its nuclear deterrence plans and arms race, is based exactly on that model: a single game known as "Chicken". By both USSR and USA having the capability of a second devastating nuclear strike to each other, the Nash equilibrium of that game was not to attack and for over thirty years proved to be true.

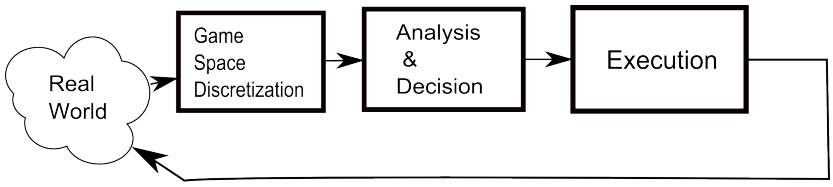

Figure 6.1: The three step process of this approach.

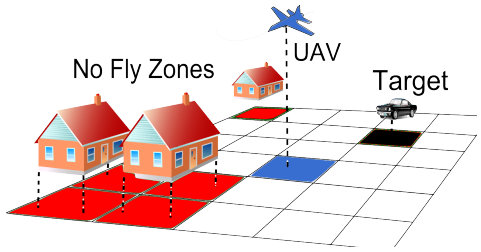

Figure 6.2: The board of the game.

6.2 Overview of the Approach

This section presents the overview of the proposed method. The method applied for each player consists of a loop of three distinct steps (Fig. 6.1):

1. *Game space discretization:* This steps consists of creating a virtual discrete world based on the real world configuration in order that a game theoretic analysis may take place. This step takes as input the positions of pursuers, evaders, obstacles, NFZs and outputs a virtual discrete map that will be known as the *board* of the game (Fig. 6.2). This map is local and updated as the drone is moving.

2. *Analysis and decision:* This step analyses the board, compares the ratio of forces and decides an optimal move for each agent *i.e.* if the architecture is used for one drone solely then an optimal move will be calculated for that drone only.

3. *Execution of the discrete command:* This block outputs a roll command for the drone (or a turn command if the algorithm is used for the ground evader) that will bring it to the desired area.

6.3 Game space discretization

The first challenge of a classical game theoretic approach is the way one would attempt to formalize the problem. While classical game theory tools are discrete, in reality such type of aero-terrestrial games are highly continuous and very demanding in terms of control frequency: the aircraft(s) and the evader are moving constantly and their position, speed and heading are only known as continuous figures. Furthermore, those commands need to be refreshed periodically in order to keep the agents on the right trajectories.

This continuous reality must somehow be transformed into a discrete one and this should be done in such a way that one can tell with a certain, quantitative way, when is the game in favor of the pursuer team and when it is in favor of the evader. Furthermore, this type

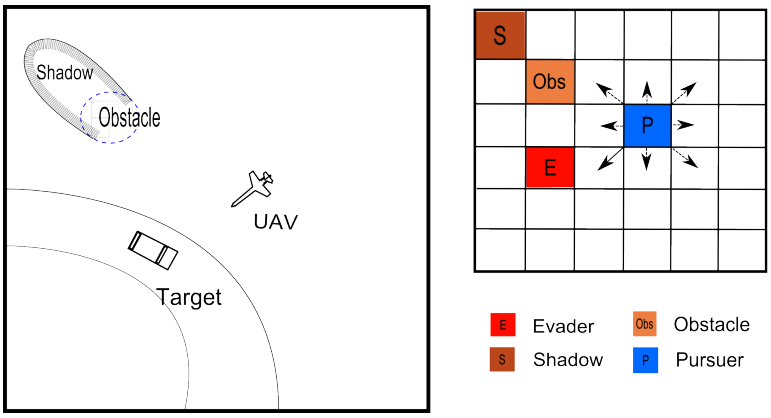

Figure 6.3: Discretization of the game space using a grid. Each cell is assigned information regarding the real game space situation.

of mapping should also be able to be reversed both ways, from discrete to continuous space and vice-versa with little or no error. In a more formal way, one must achieve a mapping $f : \mathcal{E} \rightarrow \mathcal{D}$, where $\mathcal{E}$ is the continuous configuration space that includes all the states of both pursuers and evaders plus the variables that they control (roll command etc.) and $\mathcal{D}$ is the discrete configuration space for the same type of variables and players.

The most natural idea is to decompose the area where the game evolves in many cells of equal size and thus create a two dimensional grid. To each cell $C_d \in \mathcal{D}$ one can assign an area of Cartesian coordinates $(x_d, y_d) \in \mathcal{E}$. Such a discretization of space assigns to cells all necessary information to make sure that the game is played in the same way that it would be played in the continuous world. Areas that include NFZs are be marked impassable, areas that provide shadows are marked as shadows etc. (Fig. 6.3).

6.4 Analysis

6.4.1 Evaluating Game Configurations

The first step for any game theoretic approach is to examine the outcome of every decision combination: the utilities of each player. While some configurations may yield a positive outcome for one player or team they may yield a negative outcome for the other team – or player, we will use these terms alternatively. One notion should be made clear: one does not try to evaluate individual positions but instead one attempts to evaluate entire configuration areas of the game space from the perspective of a specific player or even that of the whole team. Therefore, there may be configurations that may be costly for one team while they are completely indifferent for the other team *i.e.* a pursuer occupying the same cell with a NFZ may be costly for a pursuer, but this configuration will not influence the evader.

At this point, we define the *board* B_t of the game, which we will use whenever we want to address the two dimensional discrete space game configuration. The game configuration is the position of all obstacles, players and shadows at a given instant.

As such each board configuration will have a certain value (or payoff) $P_i(B_t) \in \Re$ for every player (or team) $i \in \{1, ..., N\}$ where N is the number of players (or teams). Therefore, each team i can evaluate any board configuration B_t and calculate the value P_i of that configuration

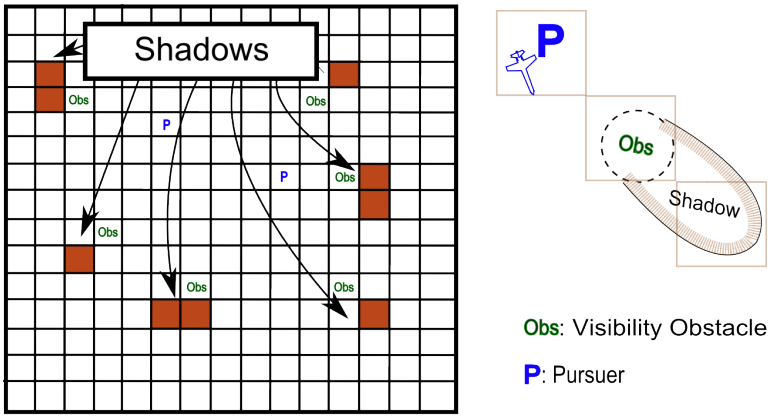

Figure 6.4: The Shadow map is a map that depicts all the shadowed areas on the board.

from the perspective of that team.

There exist two types of evaluations functions:

1. Individual agent's evaluation functions.
2. Entire team evaluation functions.

Individual pursuer evaluation function

The evaluation function, uses several matrices in order to calculate the value of a given board configuration from the perspective of a given agent: here it is from the perspective of a pursuer. It achieves that by calculating the positions of all shadows on the map based on the position of each pursuer and the position of each visibility obstacle. (Fig. 6.4). This map will be known as the *shadows map* where

$$C_j^{sh} = \begin{cases} u_{shadow} & \text{if there is a shadow in that cell} \\ 0 & \text{if there is no shadow in that cell} \end{cases} \tag{6.1}$$

Where C is a cell of a board and u_{shadow} is the utility (payoff) of the shadow.

This type of shadow map assumes that the drone can see up to infinity and with a full field of view. However, this is rarely the case. In order to encode more details of the camera capabilities, the shadow map should be manipulated as shown Fig. 6.5.

Similarly to the way the Shadows map is formed, one can form the rest of the evaluation function matrices as well: the pursuers map C^{sh} depicts the areas where there are pursuers, the evader map C^{ev} depicts the areas where there are evaders and the NFZ map C^{nfz} can show which areas are allowed to fly over and which not. Each cell of the map contains a positive value assigned to it only if in the same real area there is the matrix associated instance. In all other cases the matrix cell is equal to zero.

An exception to the rule is a special type of matrix known as the *distance Matrix*. The distance matrix forms a kind of potential field around a given matrix and this allows to reward certain areas that are closer or further from a given type of agent or instance. An example is given for the *pursuer's distance Matrix* in (Fig. 6.6). Other common types of distance matrices are the *Center distance Matrix* C_{center}^{dist} that gives greater values in position close to

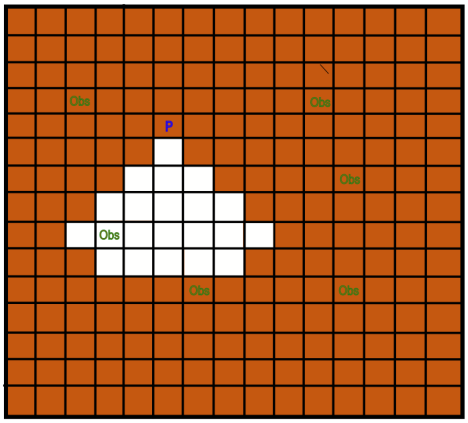

Figure 6.5: The camera field of view information can be encoded within the game by manipulating the shadow zones, within the shadow map. The brown cells, depict areas where the pursuer does not have visibility.

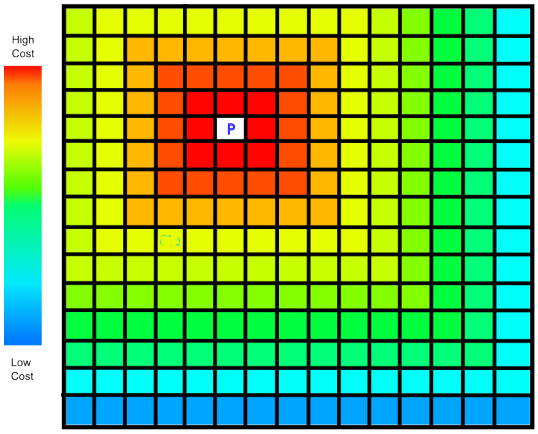

Figure 6.6: The pursuer distance Matrix rewards the configurations where the evader is closer to the pursuer.

the game board center and the *evader's distance Matrix* C_{ev}^{dist} that rewards positions close to the evader.

Then these matrices are used to calculate the value of the board from the perspective of the given player by using the equations 6.2, 6.3, 6.4 and 6.5. In these formulas we use weights for the different types of instances - an obstacle will not have the same weight as a shadow for example. We then sum up all the cells in a given map to find a given payoff. For the visibility payoff for example, we will sum up all the cells that are occupied by both an evader and a shadow. If such a sum is zero, then it means that no evader occupies the same space with a shadow.

Calculation of the visibility value of the board:

$$P_P^{vis} = \sum_j w_{shadow} C_j^{shadow} w_E C_j^E \tag{6.2}$$

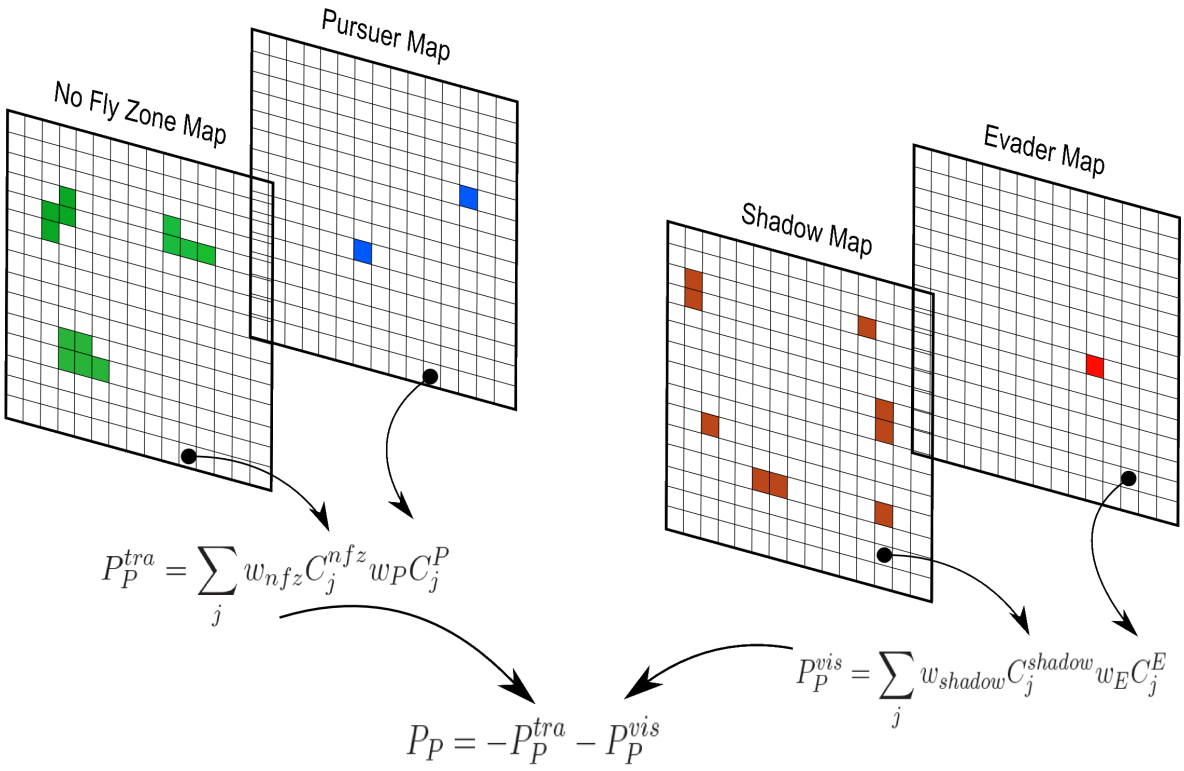

Figure 6.7: Formation of the pursuer's team value or payoff - simple version. Two pair of matrices are multiplied cell by cell and then added in order to form scalars that form the value. If a pursuer map's cell and its analog on the NFZ's map have both a non zero value then the result will be a non zero scalar.

Calculation of the traversability value:

$$P_P^{tra} = \sum_j w_{nfz} C_j^{nfz} w_P C_j^P \tag{6.3}$$

Calculation of the agents proximity to each other value of the board:

$$P_{P_E}^{prox} = \sum_j w_{prox} (C_E^{dist}{}_j + C_j^E) C_j^P \tag{6.4}$$

Calculation of the agent's proximity to the board center value:

$$P_{P_c}^{prox} = \sum_j C_{center}^{dist}{}_j C_j^P \tag{6.5}$$

The sum of those values give us the total value of a given board for the team of pursuers (Fig. 6.7):

$$P_P = -P_P^{tra} - P_P^{vis} + P_{P_E}^{prox} + P_{P_c}^{prox} \tag{6.6}$$

The weights set the priorities, *i.e.* whether the drone should move into a NFZ or let the evader slid into a shadowed zone. In order to make sure that the drone will prefer the latter, one should make sure that $w_{nfz} w_P \geq w_{sh} w_E$.

Team pursuer evaluation function

This type of evaluation function assigns a value based on the team's outcome and not only based on the outcome of an individual pursuer. The payoff function is calculated as in the individual case, using cell multiplications and then normalizing the different parts using proper weight selection.

The overall team payoff is given by the following formulas:

$$V_{team}^{pursuer} = -P_p^{tra} - P_P^{vis} + P_{P_E}^{prox} + P_{P_c}^{prox} \tag{6.7}$$

Where P_p^{tra} is the OR traversability of the team configuration:

$$P_p^{tra} = \sum_j (C_j^{P_1} + C_j^{P_2}) w_{nfz} C_j^{nfz} \tag{6.8}$$

Where P_P^{vis} is the AND visibility of both pursuers positions. This means that in order for the visibility to be lost, both pursuers must be in zones that lack evader visibility:

$$P_P^{vis} = \sum_j C_{P_1}^{shadow}{}_j C_{P_2}^{shadow}{}_j C_j^E \tag{6.9}$$

$P_{P_E}^{prox}$ offers an element that rewards positions that bring the evader close to any of the drones.

$$P_{P_E}^{prox} = \sum_j (C_j^{P_1} + C_j^{P_2}) w_{prox} C_j^E \tag{6.10}$$

Lastly, the use of P_{pc}^{prox} rewards positions close to the center of the board. This was done because during our simulations, we had to keep the game contained within an area and decided to do so using this formula.

$$P_{P_C}^{prox} = \sum_j (C_j^{P_1} + C_j^{P_2}) C_{center\,j}^{dist} \tag{6.11}$$

Evader's evaluation function

The evaluation function is quite similar for the evader with some slight differences.

Calculation of the visibility value of the board:

$$P_E^{vis} = \sum_j w_{shadow} C_j^{shadow} w_E C_j^E \tag{6.12}$$

Calculation of the traversability value of the board:

$$P_E^{tra} = \sum_j w_{ground} C_j^{ground} w_E C_j^E \tag{6.13}$$

Calculation of the agents proximity to each other value of the board:

$$P_{P_E}^{prox} = \sum_j w_{prox} (C_E^{dist}{}_j + C_j^E) C_j^P \tag{6.14}$$

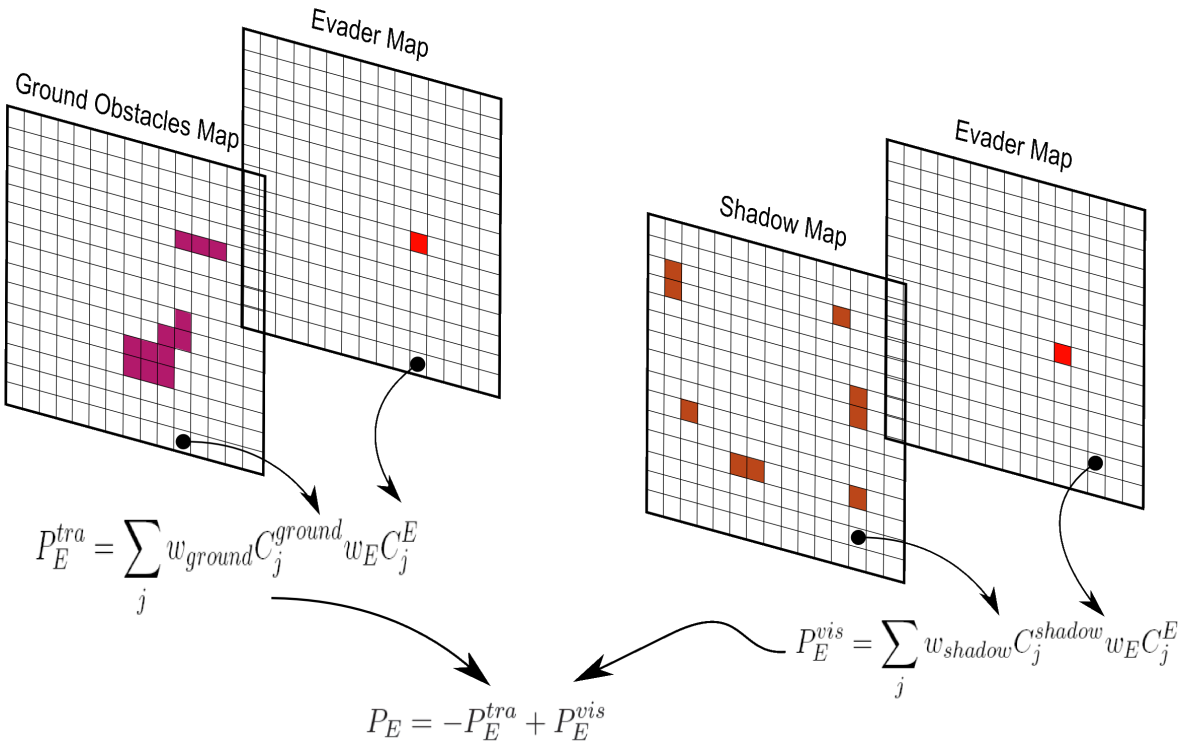

Figure 6.8: Formation of the evader's payoff.

Calculation of the agent's proximity to the board center value:

$$P_{E_c}^{prox} = \sum_j C_{center\,j}^{dist} C_j^E \tag{6.15}$$

The total board value for the evader is the following:

$$P_E = -P_E^{tra} + P_E^{vis} - P_{P_E}^{prox} + P_{P_c}^{prox} \tag{6.16}$$

Notice that the sign of the visibility and the sight of the intra-agent proximity values are inverted here showing that the same configurations that maximize the pursuer's utilities are the same that minimize the evader utilities and vice versa. (Fig. 6.8).

6.5 Mono-pursuer decision making

This section presents the solutions for one pursuer tracking one ground evader. Therefore the words team and player have here the exact same meaning. The objective of this section is to use different game theoretic tools to take decisions for each team.

6.5.1 Utility Maximization

This is the simplest method: the pursuer examines all possible alternatives (cells on the board) and decides to use the one that offers the highest payoff $argmax(P_P)$ using its evaluation function based on the *current* evader position.

6.5.2 Adversarial Searching

The most promising method in adversarial searching is the minimax algorithm as it was presented in the state of the art chapter. Using this method one can either solve the tree using local methods *i.e.* examining as probable destinations only the cells next to the cell under question, or considering all the cells as options. The minimax solution is known to be a mixed strategy and for zero sum games is equivalent to the Nash equilibrium solution.

Local Adversarial Searching

During this method each leaf node expands only to its closest neighbors. The maximum branching factor of such a search tree is 9, including the option of remaining on the same cell. Therefore, the maximum time complexity of the minimax algorithm is $O(bm)$ where b is the branching factor and m the depth and the space complexity is $O(b^m)$. However, the algorithm will examine the neighbor cells and if an obstacle is occupying one of those cells, that cell will not be added to the initial node's children.

Global Adversarial Searching

This method is more exhaustive than the local version and it uses a branching factor equal to the number of cells of a specific game area (or even the entire board) minus the cells that are occupied by obstacles. However, if the optimal decision for a player is more than a cell away then it means that it will take some time to get there and in the meantime the ratio of forces may change. Furthermore, there is no guarantee that the opponent can see in the space horizon that the algorithm is predicting. In the latter case the pursuer may find itself in a far away cell, anticipating a movement that his opponent will never make because he simply cannot predict that far.

On the other hand, this method allows us to take more strategic decisions using less computer resources i.g. for a solution that lies 5 cells away from the current agent position, the local method may need to examine up to 9^5 nodes before it finds it as it has a space complexity of $O(b^m)$, while the global method will see the solution with much less nodes opened. A board can have as much cells as one wishes to examine, depending on the horizon around the drone one wishes to examine plus the number of cells that one wants to include.

In this game the branching factor is equal to the number of possible alternatives and this is the number of free cells on the game board. One can easily understand that a search tree of this type cannot be explored in a greater depth: Already at b=100 and one has to evaluate 10K combinations before a conclusion is reached.

Furthermore, predicting further than one decision step for each player would not offer as much as one would think as it would mean that the evader would not select the most easily accessible area but some other more elaborated solution. One can simply not expect that the evader will think as far as that decision horizon.

However, as the moves are considered to be somewhat simultaneous when it has to do with one movement ahead, it is not important who plays first. Hence, if one keeps the decision adversarial tree at one step ahead for each player, it does not matter who plays first.

It suffices if first the evader decides for its optimum movement and then based on that decision to find the best response for the pursuer (Fig. 6.9). This form of minimax algorithm has only O(2b) space complexity - b is always the branching factor of the decision tree .

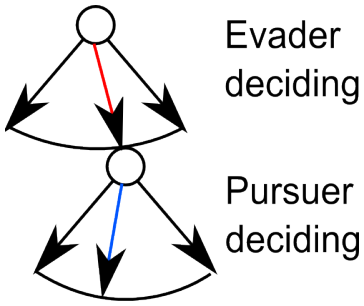

Figure 6.9: The minimax algorithm used in this type of game: First, one examines the alternatives of the evader given the current pursuer position and a decision is made. Based on that speculative decision, pursuer is making his own choice.

6.5.3 Nash equilibria and dominance strategies

This approach attempts to find the Nash equilibria for a non cooperative game. The game matrix is constructed as follows: we consider two players P and E and each cell of the matrix denotes the payoffs for both players $\{P_P^{\kappa\lambda}, P_E^{\kappa\lambda}\}$. Each player has to choose between m strategies, where each strategy is a cell $C_d \in B_t$ belonging to the board. The number of strategies each player has at its disposition is equal to the number of cells on the board. Player's P strategy is κ and player's E strategy is λ.

A pair of strategies $\{\kappa^*, \lambda^*\}$ is called a Nash equilibrium if
$\forall\kappa = 1, ..., C_m$ and $\forall\lambda = 1, ..., C_m$:

$$P_P^{\kappa^*\lambda^*} \geq P_P^{\kappa\lambda^*} \tag{6.17}$$

and

$$P_E^{\kappa^*\lambda^*} \geq P_E^{\kappa^*\lambda} \tag{6.18}$$

Finding the Nash equilibrium is considered to be NP hard even if it has not been proved yet [Porter et al., 2008]. The algorithm used in this thesis can be found in [Sureka and Wurman, 2005] – and has been sketched in section 2.2.2. The algorithm is a tabu, gradient based search algorithm on the game's matrix. Therefore, if there are multiple Nash equilibria, the found equilibrium will depend on the position from which the search was initiated. It is better to initiate the search from the actual position of the agents, as distances in the matrices have a direct relationship with real world distances.

The reason we chose this algorithm is because this game;s matrix is quite big: 10000 elements for a board of 100 cells and only this algorithm provides a fast searching method to localize a pure Nash equilibrium, without having to compute all the payoffs of the matrix - their number being around the figure of 20K. Other popular algorithms can be found in [Lemke and Howson, 1964, Porter et al., 2008].

6.6 Multi-pursuer decision making

During this section, we are going to examine the methods that permit to two pursuer to collaborate optimally using game theoretic tools in order to maximize evader visibility.

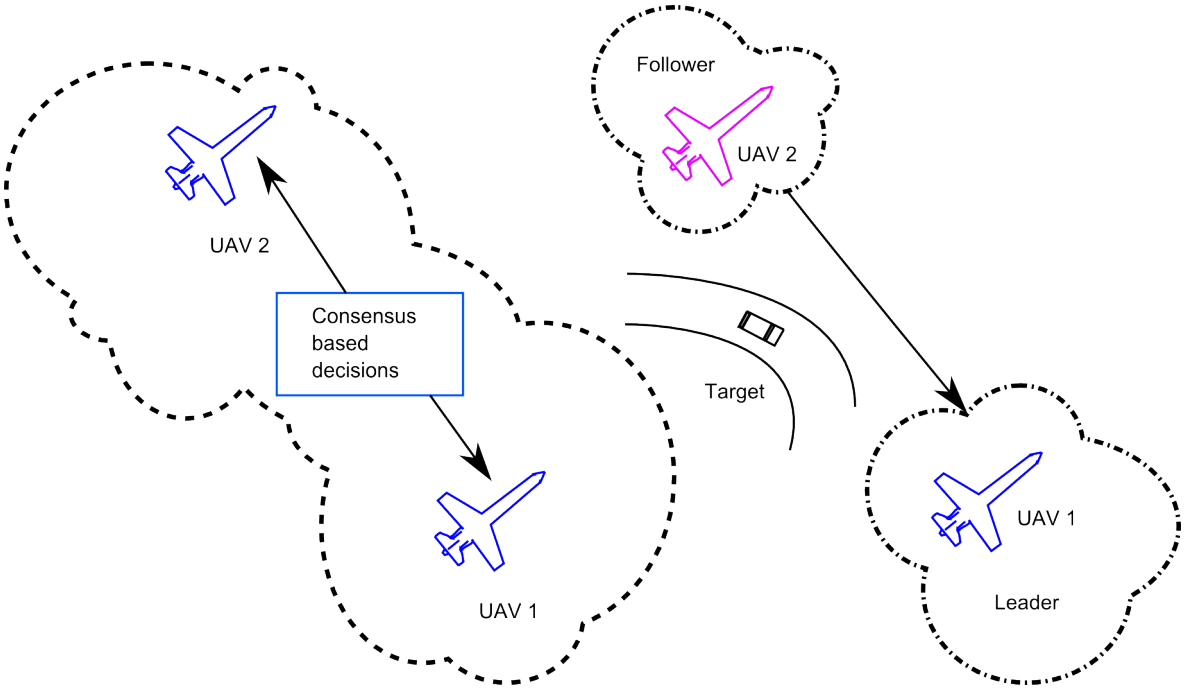

Figure 6.10: Two approaches are proposed for the collaboration of two pursuers. According to the consensus based approach both drones take each others presence for granted and attempt to find a common solution that maximizes group utility. On the hierarchical approach the leader drone ignores the presence of the second and continues selfishly to maximize its own utility, while the follower drone attempts to position itself in such a way that maximizes group utility and not its own. This way it kind of acts like a visibility safeguard.

6.6.1 Hierarchical VS Consensus based approaches

When dealing with two drones, one no longer evaluates the position of a single player but the entire team's configuration. The first question to answer the role of each pursuer. There are indeed two ways of tackling this problem:

- Either consider both pursuers as equals and thus make sure that both pursuers have to change positions in order to maximize their team utility (both players arrive at a common consensus). The behavior then will arise through local interactions,
- Or use an hierarchical scheme and thus while the first pursuer will behave as it is by its own, the second will do all the extra calculations in order to maximize the team's position even if the first pursuer is blind to its actions.

Consensus based methods need not to be centralized: by running the same analysis and decision algorithms in both pursuers, it is almost sure that the same decisions will be taken – if the same information is available to both agents. However, as this is not always the case, one can also profit by letting the two agents communicating either the information or by running the process in one of the two agents and deciding for both. If consensus methods are used, then the same type of multi-pursuer decision algorithm is guiding both drones.

Hierarchical scheme is a scheme that can add as many drones as needed without having to change the decision process of the drones already in the mission: if one adds a N-th pursuer to the loop, then all N-1 previous players are not aware of its efforts but the N-th pursuer has to take into account all their positions and to make sure that the team utility is maximized. If hierarchical methods are used then one drone will continue to use mono-pursuer decision methods, while the second or any other drone above that layer will use multi-decision techniques to draw its decisions. In one sense the drone that uses mono-decision techniques is considered a leader and all other drones followers.

drones. There are two main alternatives: a) Use a leader that decides all the future positions of every drone in the group, by solving a global optimisation problem or b) Let the behavior arise through local interactions. The first method is based on the communications between the different agents. The leader has to establish a communication channel with every other drone of the group and send through it, the future flight plans.

In both cases we believe the proposed hierarchical approach is a good compromise that minimizes the communication burden without risking the stability of the group formation.

6.6.2 Utility Maximization

This is the same method as it was presented for the mono-pursuer case but this time applied to the entire pursuer team. The pursuer examines all possible alternatives (cells on the board) and decides to use the one that offers the highest payoff using its team evaluation function based on the *current* evader position.

$$C_{dec} = argmax P_P^{team} \tag{6.19}$$

6.6.3 Cooperative Nash equilibria

This approach attempts to find the Nash equilibria for a cooperative game, this means that both pursuers were commanded but without a coordinated command. The game matrix is constructed as follows: we consider two players P_1 and P_2 and each cell of the matrix denotes the payoffs for both players $\{P_{P_1}^{\kappa_1\kappa_2}, P_{P_2}^{\kappa_1\kappa_2}\}$. Each player has to choose between m strategies, where each strategy is a cell $C_d \in B_t$ belonging to the board. The number of strategies each player has at its disposition is equal to the number of cells in the board. Player's P strategy is κ_1 and player's E strategy is κ_2.

A pair of strategies $\{\kappa_1^*, \kappa_2^*\}$ is called a Nash equilibrium if
$\forall \kappa_1 = 1, ..., C_m$ and $\forall \kappa_2 = 1, ..., C_m$:

$$P_{P_1}^{\kappa_1^*\kappa_2^*} \geq P_{P_1}^{\kappa_1\kappa_2^*} \tag{6.20}$$

and

$$P_{P_2}^{\kappa_1^*\kappa_2^*} \geq P_{P_2}^{\kappa_1^*\kappa_2} \tag{6.21}$$

In order to find the Nash equilibrium we use the same method used in the mono-pursuer decision case. In this approach the two agents are two drones that try to cooperate while in the previous Nash use one agent was a pursuer and the other was the ground evader.

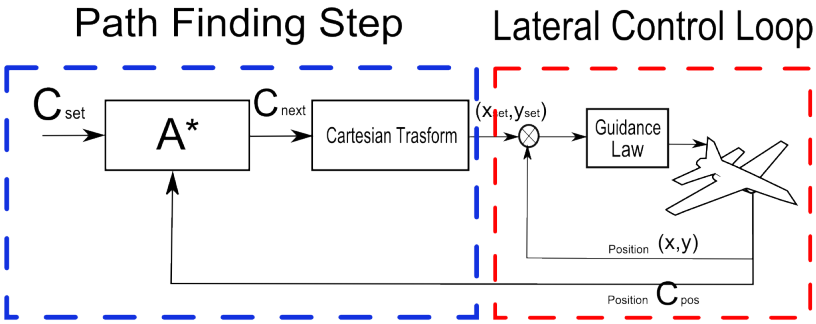

Figure 6.11: Making sure that the decision is properly executed is a task of the Realization block. During this step the set point cell C_{set} is passed along with the current drone cell C_{pos} to the A* algorithm who finds a clear, shortest route from $C_{pos} \rightarrow C_{set}$. C_{next}, the first step of that route, is passed to a block that returns its Cartesian coordinates. Those coordinates are then passed on to a lateral control loop, that using a guidance law and a position feedback brings the aircraft there.

6.6.4 Pareto optimality method

The Pareto optimality method, as presented in section 2.3, examines the individual payoffs U_{P_1}, U_{P_2} of every player and then tries to maximize the collective payoff:

$$\text{Select } \{\kappa_1^*, \kappa_2^*\} \text{ s.t. } argmax\{U_{P_1}^{\kappa_1^*} + U_{P_2}^{\kappa_2^*}\}$$

Once again one has to avoid to calculate all possible payoffs from every possible combination so some search method must be used. We decided to use a tabu, search method in order to find the optimal points.

6.7 Realization

As during the discretization step there exists a discontinuity between the abstract, game theoretic world and the continuous real one. The output of the analysis and decision step is a cell. However, if the agent is not on that cell, as in most cases it will not be, some procedure must make sure to bring it there. The Realization step achieves that using two steps (Fig. 6.11).

1. *Path Finding:* It first finds the shortest route from the current cell C_{pos} until the desired cell C_{set}, while avoiding any obstacles or NFZs on the way. It achieves that by using an A* search algorithm. It then takes the first cell C_{next} of that optimal route and passes it on to the next step.

2. *Lateral Control Loop:* This step takes as input the current position of the aircraft and a desired cell. It transforms the cell position in Cartesian coordinates (x_{set}, y_{set}) and uses those coordinates along with some guidance algorithm to navigate the agent there.

6.8 Extension of the evaluation functions: visibility maps

In this section we present an extension of the method that permit to guess how the evader will move next without having to take into account the present position of the pursuer or the evader. It also provides a way for the evader to avoid capture without having exact information for where the pursuer is at every given moment.

The method consists of calculating a priori the areas on a map that visibility can be hindered and then using this map as a guess for how the evader will move next.

Under certain circumstances, one can safely consider that the evader will actively try to select positions in the environment that cannot be seen easily from the air. This subject arises several questions:

- Is there any way to measure how visible is a point from the sky ?
- Which factors affect the visibility of a point from the sky ?

In this section we examine these elements and we provide a simple way to examine all points on the map and determine, how well they can be seen from the sky. This produces a map that can be superimposed to a real one and offer a visibility measure for all points on that map. If this map is known beforehand, one can assume that a well informed evader will always try to find a cover close to those areas. This may prove useful for the pursuer as it describes the possible safe destinations. It should also be noted that this map is independent of the position of the pursuer and this can be useful for the evader that does not have the means to know constantly at every moment the position of the drone.

6.8.1 Indicator of Stealthiness

The first criterion that helps us understand how well an area can hide a evader, is the overall percentage of the FOV from which the evader can be visible. An area that is surrounded by 4 buildings is far better hiding area than an a field with only one building. The second criterion is the proximity of obstacles to the area under examination: the more obstacles are around a certain area the more probable it is for the evader to find an area to hide itself there.

To understand this, lets visualize the following example (Fig. 6.12): on the one hand, we have a castle with a huge garden and on the other hand, a forest with a small road passing through. While, both configurations probably conceal the evader from almost any point in the horizon, things change drastically when the observer (pursuer) approaches and starts to circle around the evader. In the case of the castle, almost 100% visibility can be achieved if the garden of the castle is big enough to allow the drone to circle within or close to the boundaries of the external walls. However, this would not be the case for the forest road. Here, the opening is so small that an airplane circling the area would not manage to see the evader all of the time. In the same sense it is better to be closer to a wall than further away.

In order to put this into a formula that will permit us to evaluate a point on the map as to what extent it can hide the evader, we use the approach that follows.

6.8.2 Measuring Stealthiness from the air

The size of a shadow depends from the height of the object, the distance of the observer from the area in question and from the altitude of the observer – in this case the pursuer.

However, one of the simplifications we did during the problem statement of this thesis was that the altitude of the pursuer would remain constant during the whole mission and that we will consider the height of all obstacles identical.

$$L_{sh} = f(d_{obs}^{eye}, h_{obs}, h_{eye}) \tag{6.22}$$

The above formula states that the size of a shadowed area that an obstacle of fixed size casts depends from the height of the obstacle, the altitude of the eye (pursuer) and the distance between the two of them.

Throughout, this chapter we constantly measured the positions of the different agents and tried to calculate the position of the shadow. This approach does something different: it assigns visibility values to the different cells without knowing where the drone may be - the distance d_{obs}^{eye} which is presumed fixed and within the average camera FOV distance.

The algorithm then takes that value and divides the map into cells of size L_{sh}.

It then proceeds, cell by cell, to test how many of the neighbor cells are occupied with obstacles. It then assigns a value from 0 to 9 to that cell, with 0 being a complete open area and 9 being an area that is shadowed from all directions. The algorithm steps can be seen in Fig.6.14.

loop
  Select L_{sh}
  Divide the map into cells of equal size L_{sh}
  for (each cell C_j in B_t) **do**
    if (within the cell area there is an obstacle) **then**
      $C_j = 1$
    else
      $C_j = 0$
    end if
  end for
  for (each cell C_j) **do**
    $N_j = 0$
    for $i = 0$ to $i = 9$ **do**
      add the values of the surrounding cells
      $N_j = N_j + C_j$
    end for
  end for
end loop

The map that is then produced can be used in the evaders evaluation function as it was seen earlier and offer an alternative as to where the evader may go next. The basic idea is that the evader can use the closest hiding area to the evader as a hideout and thus the pursuer must assume a position that will find the evader.

The positive point is that the above calculation can be done offline before the start of the mission and thus it will not provide an extra burden to a real time calculation.

	Nash	Minimax	Utility Max	Test Case (Random Movement)
τ_{vis}	96%	93%	92%	83%
Vis Loss (sec)	15	30	32	66

Table 6.1: Visibility ratios τ_{vis} for a low obstacle density map

6.9 Results

6.9.1 Evaluating the different approaches

In order to simulate an evasive evader, we presumed that the evader uses a decision method that maximizes its utility.

All algorithms are coded in C ANSI and are interfaced with the open source flight platform known as Paparazzi. Paparazzi uses quite realistic models and permits us to use the exact same code in simulation and real flight testing. Finally, all graphs were passed on to Scilab, an open source program developed by INRIA.

The number of obstacles can vary and it is assumed that an approach that behaves relatively well when there are not many obstacles and thus shadows in an area, it may find a hard time to produce results when there are many obstacles in the area. In order to avoid the necessity of building huge maps, we decided to close an area and force the agents to move in that closed arena. This means that the agents will be forced to meet the same obstacles over and over again. In that sense, what is important is not the number of obstacles an area has but their relative density. For our testing, we used three type of maps of light, medium and heavy density. Finally, each test lasted approximately 7.5 minutes. All these elements are exactly the same conditions we encounter when we fly real aircrafts, were we have a limited airspace to work with, different battery limitations and random air conditions. This was done to ensure similarities between simulations and real flight testing.

6.9.2 Mono-pursuer decision making

During the first series of tests (Fig. 6.15), we used a medium obstacle map. There the approach that behaved the best was the minimax approach that lost the evader only for 9 seconds in a test of duration of 450 seconds. However it should be said that those figures are simply demonstrative as the size and type of real shadows will differ greatly from these simulations. The ratio of such a method is 98%.

The method that behaves a little less well was the Nash equilibrium with a ratio of 96% and third comes the utility max with a ratio of 92%. In order to give an image of how the method calculates the payoffs functions one can see figure 6.16 a snapshot of calculating the payoff of all the cells of the map. Each cell represents an area of about 5m by 5m.

During the second series of simulations (Fig. 6.17), we use also a test approach to compare the methods with. During this test approach the drone is commanded to perform pseudo-random arcs around the area. In these simulations the density of obstacles is very low. In low densities the probability that a random moving aircraft will capture the evader is about 83% as it was proved by this run. However other test took the number down even to 70%.

The classification of the three methods plus the test case can be seen table 6.1.

In a third series of simulations (Fig. 6.18) the agents found themselves in a high obstacle density environment and the results can be seen in table 6.2.

	Nash	Minimax	Utility Max	Test Case (Random Movement)
τ_{vis}	98%	97%	95%	93%
Vis Loss (sec)	5	12	19	23

Table 6.2: Visibility ratios τ_{vis} for a high obstacle density map

6.9.3 Multi-pursuer decision making

In the multi-pursuer scenario, two drones cooperated together to find out how the two drones can augment the team visibility. Just by adding a second drone, the number of shadowed areas diminishes significantly as now there are far less areas that cannot be seen from both pursuers. However, as it is shown there are approaches that behave much better than others.

In the following test, three approaches were tested in a high obstacle environment (Fig. 6.19), and found out that they behave much better than the test method as it was presented earlier. Specifically, the method of cooperative Nash approach behaved equally well with the hierarchical Utility max method by losing the evader from sight of view for only 7 seconds. This is a significant improvement from the mono-pursuer approach of the same map that lost the evader for about 22 seconds. These figures of course are simply for comparison reason as in reality, the shadow formations are very different and the lengths of flights are longer, so errors do accumulate. Third, the Pareto approach scored less because of the tendency of the method to bring some times the two drones in close positions and falling in visibility traps. Something logical if someone considers the way that the Pareto method chooses its positions: by summing up the individual interests of two agents.

6.10 Conclusions

This chapter presents a game theoretic approach to UAV ground target tracking. The method works well as long as the evader behaves evasively. It consists of creating a discrete grid map of the real world and then analyzing the situation using game theoretic tools.

The approach uses evaluation functions to form payoffs according to each agent's perspective. The tools used for the mono pursuer case are:

- a maximization of the utilities of each player,
- a Nash equilibrium for two competitive players: pursuer and evader,
- a global adversarial search.

A second UAV is added and the approach is expanded to the two pursuers scenario. The two pursuers attempt to maximize the team utility. This is achieved by using the following tools:

- a maximization of the utilities of each team,
- a Nash equilibrium for two cooperative players: pursuer1 and pursuer2,
- a global adversarial search.

The approach has been tested using simulations. The software platform used for simulating is the same platform used for real flight testing and the models are very similar to real ones. Hence, one should expect that the behavior of the UAVs during real flights would be similar as in simulation but maybe of different quality.

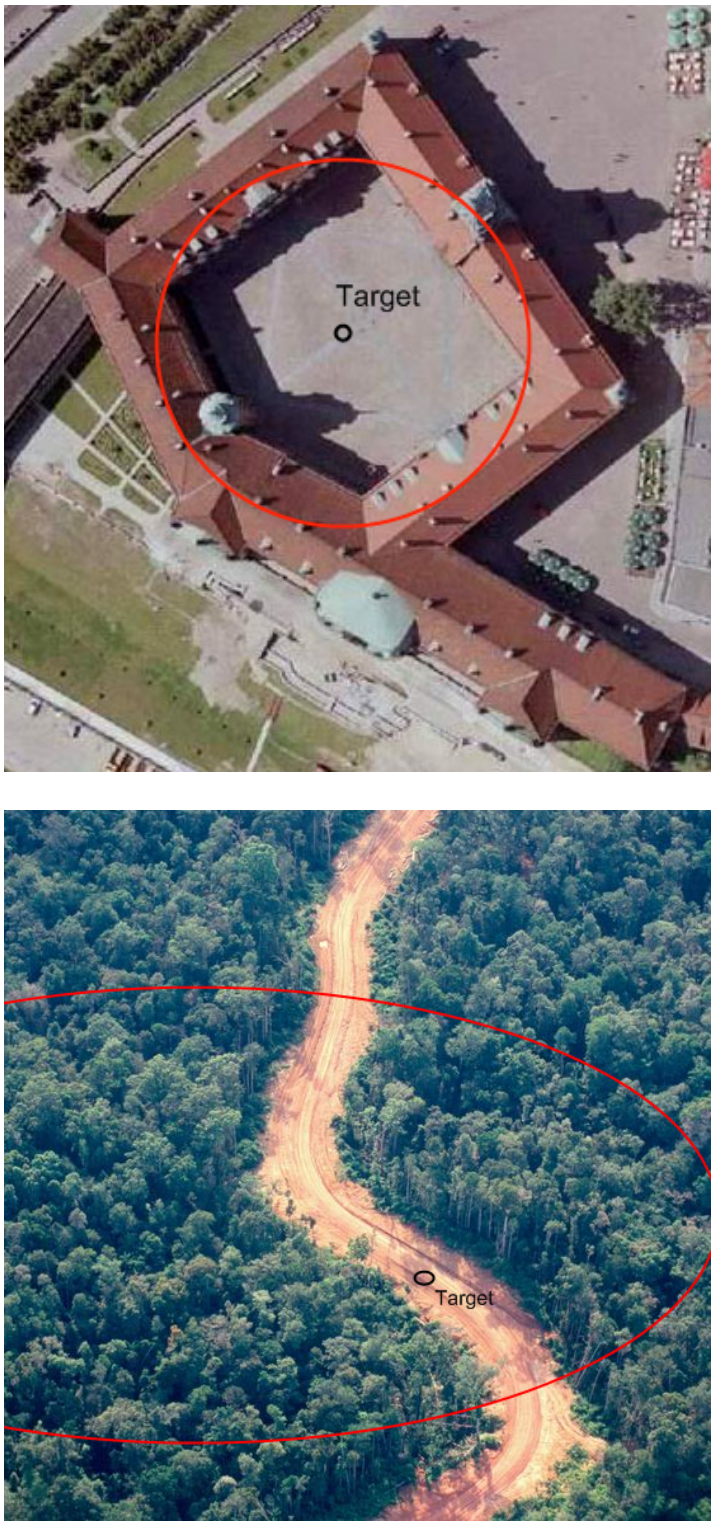

Figure 6.12: Aircraft/evader geometry. Top: aircraft trajectory (red) around an area with poor Indicator of Internal Stealthiness (Here a castle) something that would permit an aircraft to track a evader (black circle in the middle) by circling over it and within the wall limits. Bottom: obstacle structure with good Indicator of Internal Stealthiness (Here a forest road). An airplane circling over the evader would not manage to hold visibility with it 100% of the time.

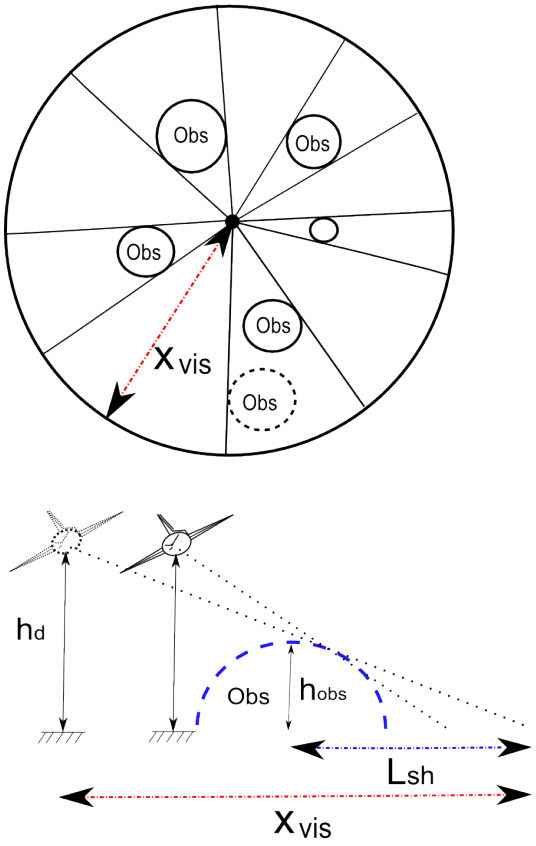

Figure 6.13: The indicator of stealthiness of a given point depends on different parameters as what percentage of the total FOV around it is open or not. In order to calculate that it is needed to know how far the shadow of an obstacle can reach.

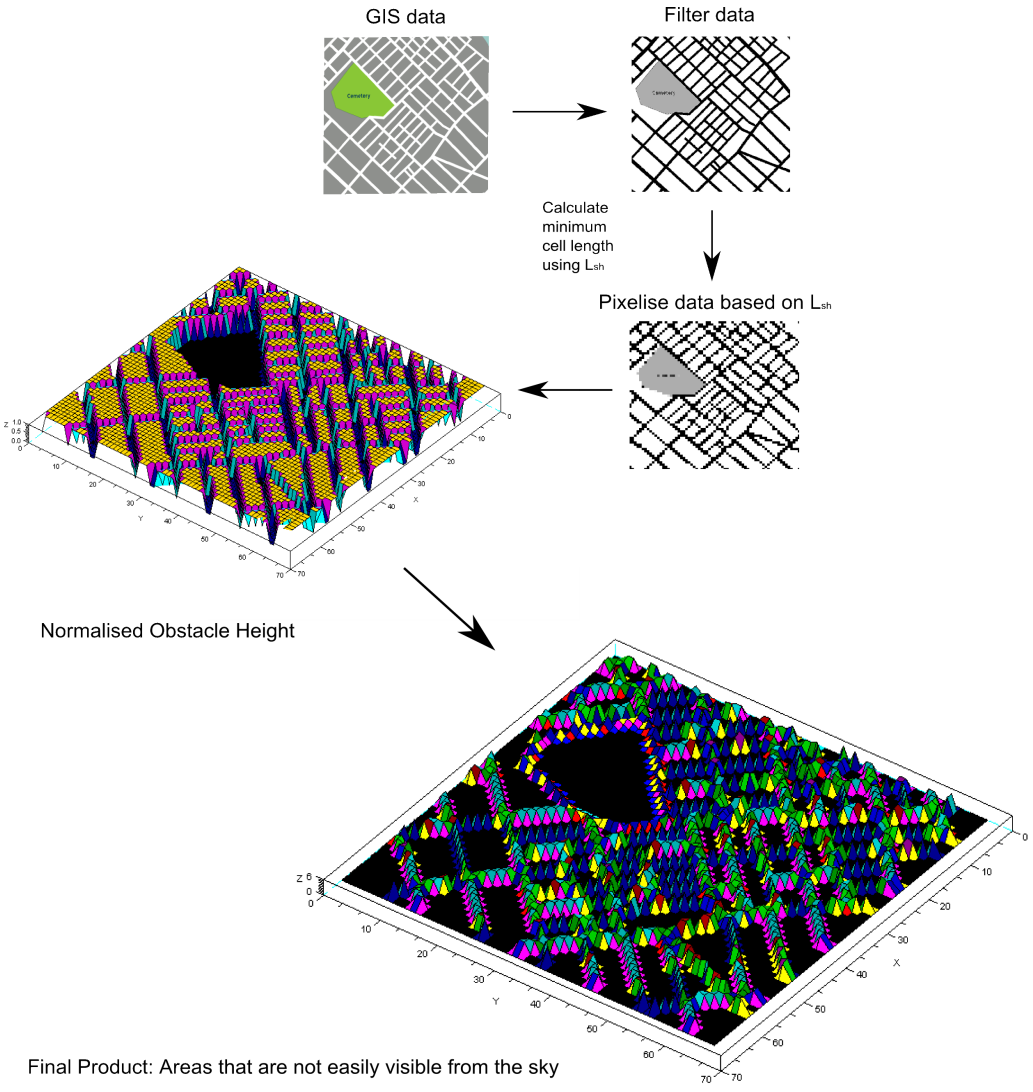

Figure 6.14: The algorithm that detects places on the map that have a good stealthiness indicator.

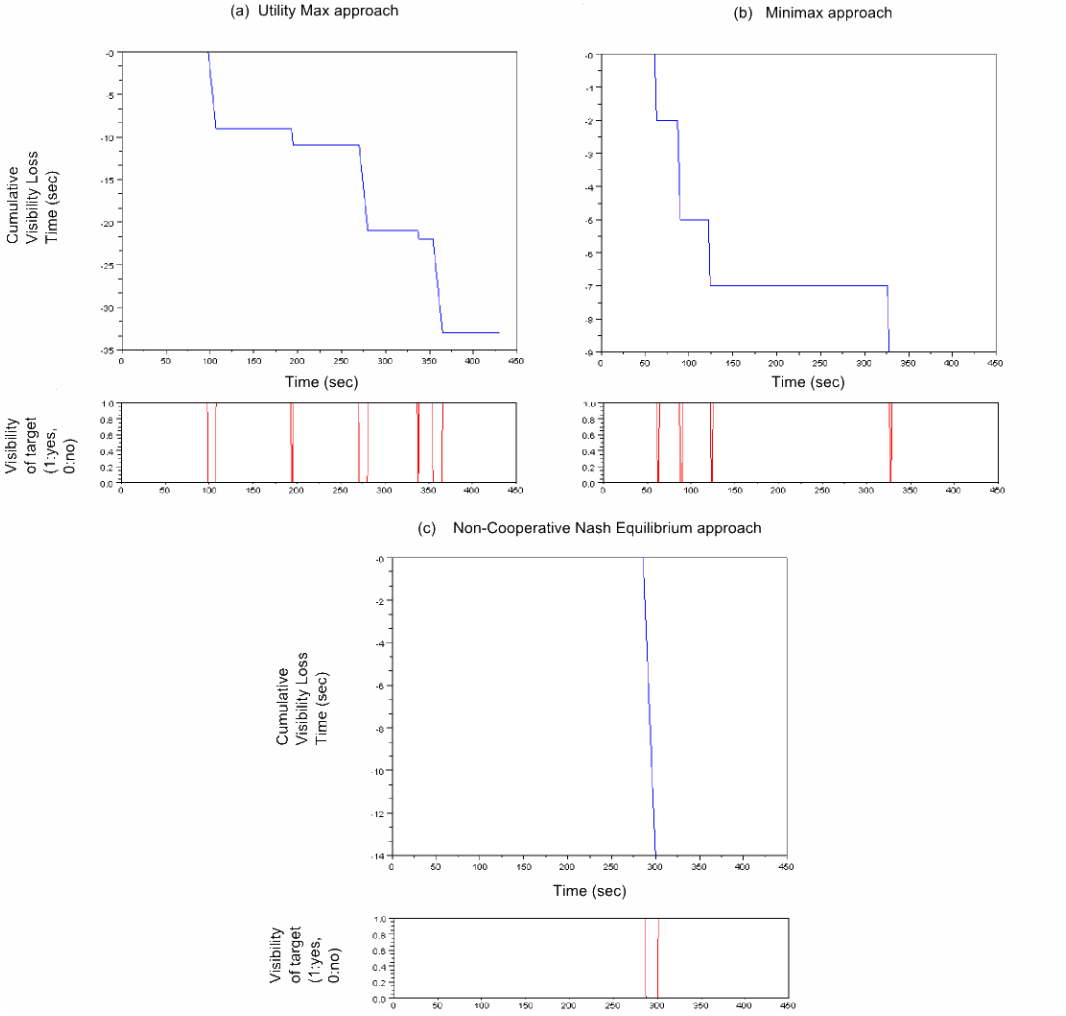

Figure 6.15: This is a simulation at a medium obstacle map. The three approaches (Nash, Utility Max and Minimax) are tested one against another: the result is measured by the amount of time that the pursuer looses sight of the evader during a 450 second duration test. The smaller graphs (red) show the visibility of the evader in a boolean manner: if the evader is seen, the value is equal to one.

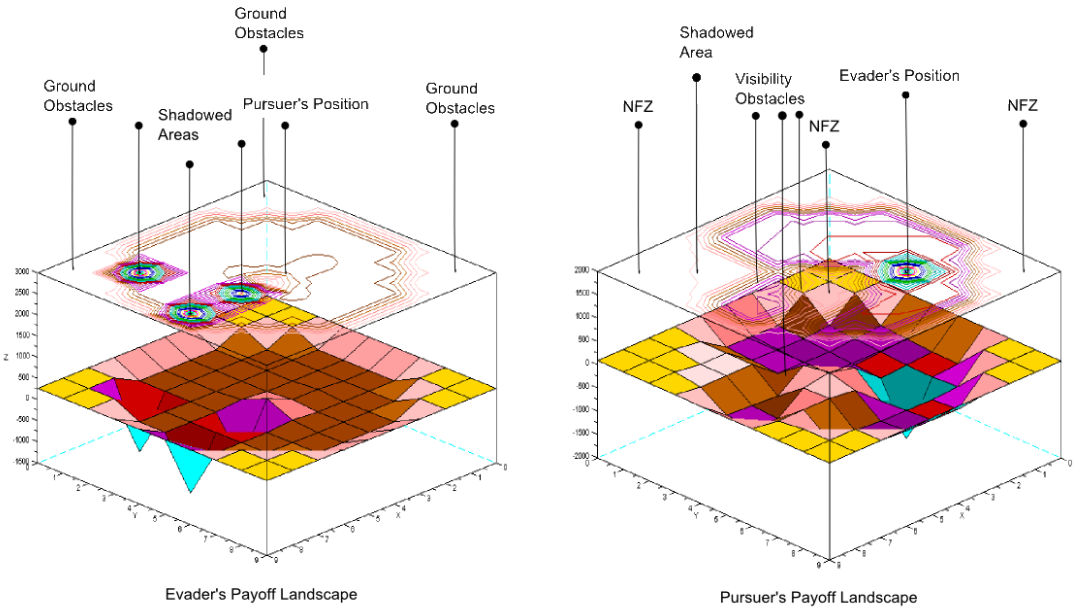

Figure 6.16: A snapshot of the pursuer's and evader's evaluation landscape for an area of 100 cells, during simulations. One can clearly see that what is for the evader a well and thus an attractive area, in exactly the opposite for pursuer: The area down left.

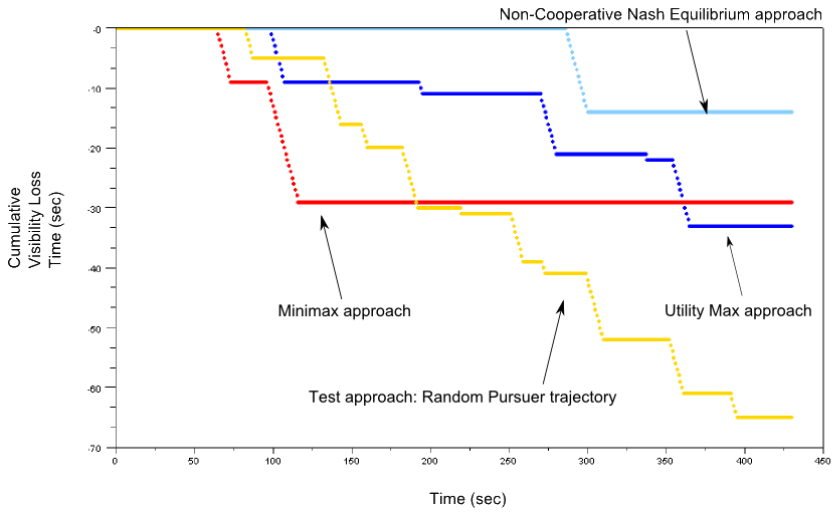

Figure 6.17: This is a simulation at a light obstacle map. The three approaches (Nash, Utility Max, Minimax and Random) are tested one against another: the result is measured by the amount of time that the pursuer looses sight of the evader during a 450 second duration test. It is clear that the Nash approach behaves better than all the others followed by the Utility Maximization approach, the Minimax and finally the Random test case is shown for reasons of comparison.

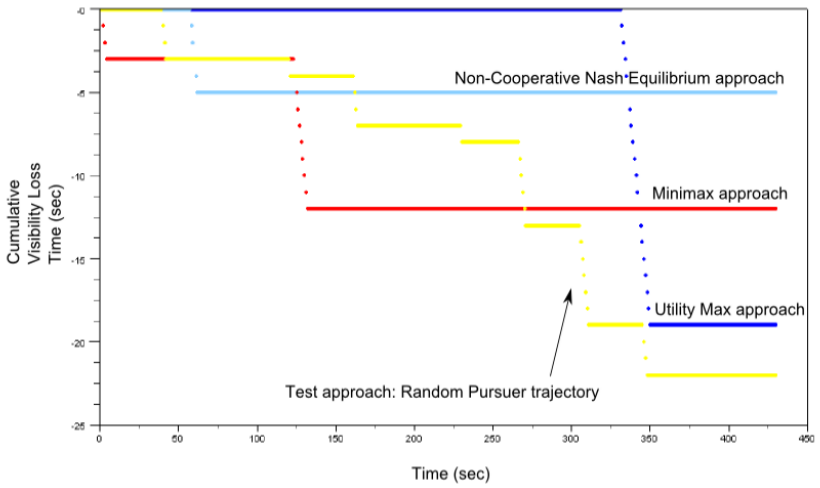

Figure 6.18: This is a simulation at a high obstacle map. The three approaches (Nash, Utility Max, Minimax and Random) are tested one against another: the result is measured by the amount of time that the pursuer looses sight of the evader during a 450 second duration test. It is clear that the Nash approach behaves better than all the others followed by the Utility Maximization approach, the Minimax and finally the Random test case is shown for reasons of comparison.

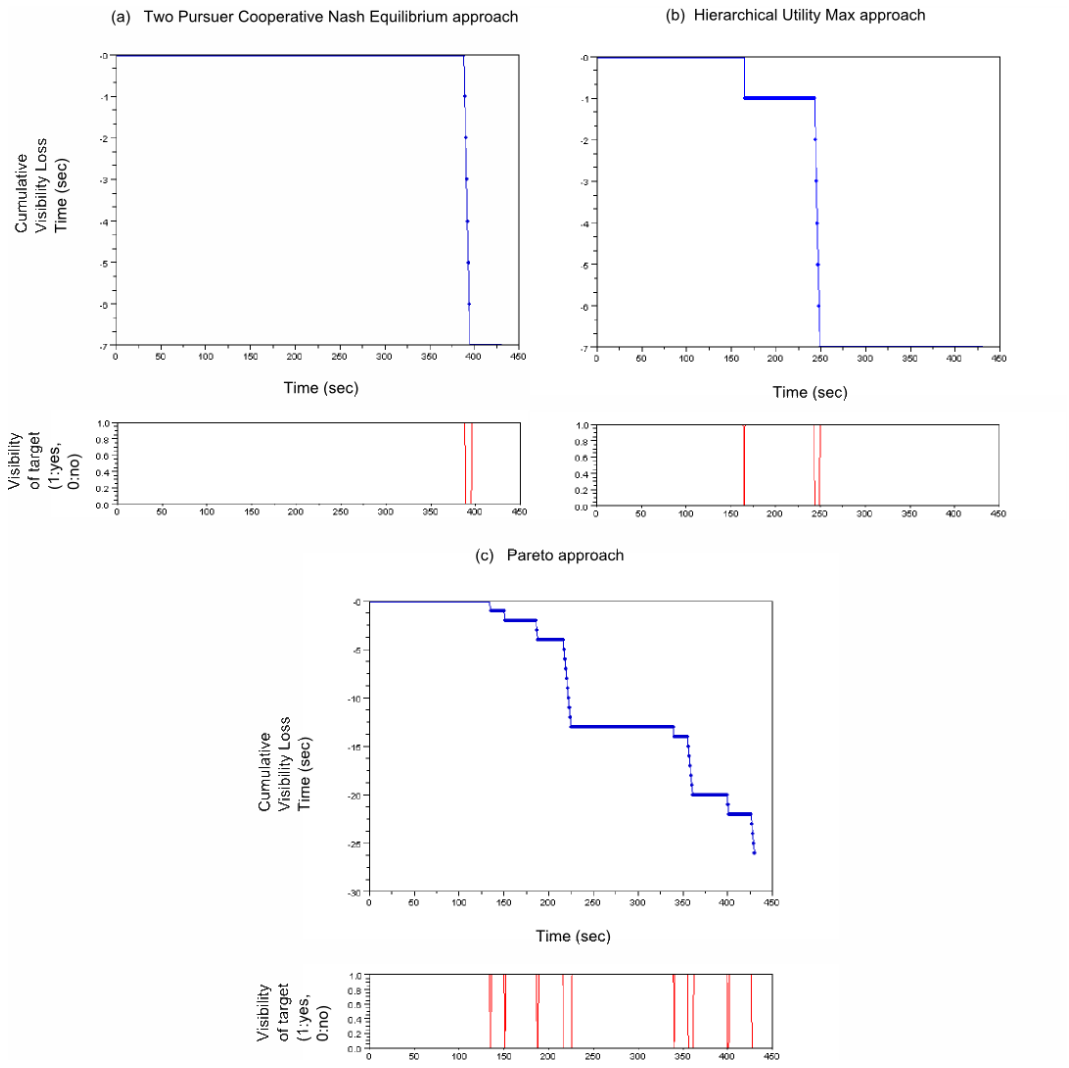

Figure 6.19: This is a simulation with only four visibility obstructing obstacles in the field and the drones in cooperating mode. The three approaches (Nash, Utility Max and Pareto) were compared with each other: the pursuer was performing random arcs around the obstacles, while the evader was actively trying to avoid visibility exposure. The results were measured in how much time did the pursuer loose the evader from sight. The Nash equilibrium method showed the best results.

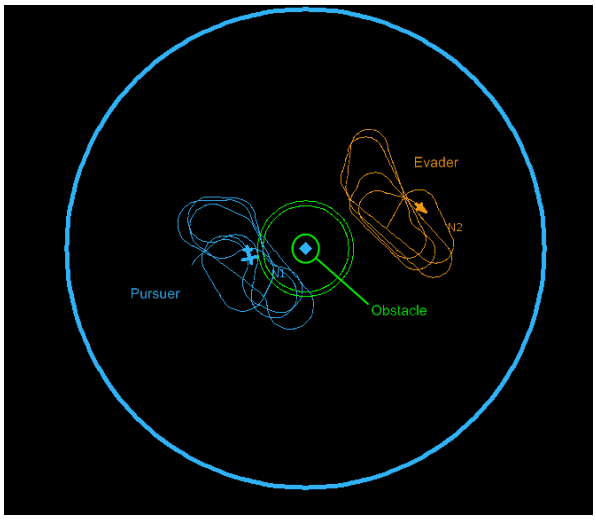

Figure 6.20: Snapshot of the pursuer while chasing the evader,while both using a max-utility approach. One can observe the limit cycles that are formed while the evader tries to hide itself behind the center obstacle.

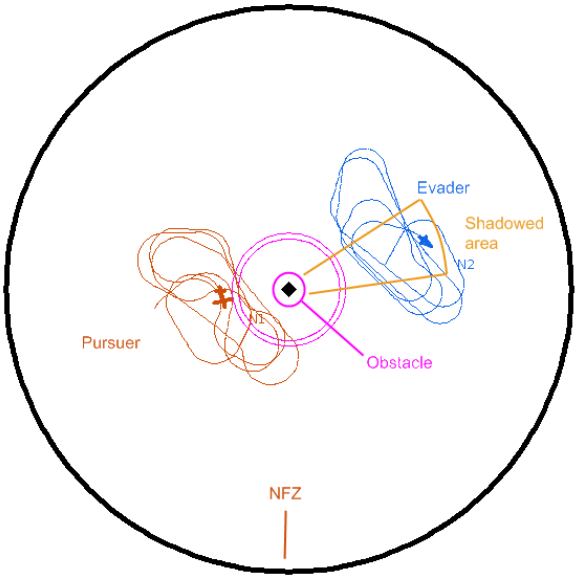

Figure 6.21: This is a picture showing how the shadow moves with the pursuer and thus causing an interesting hide and seek game around the obstacle.

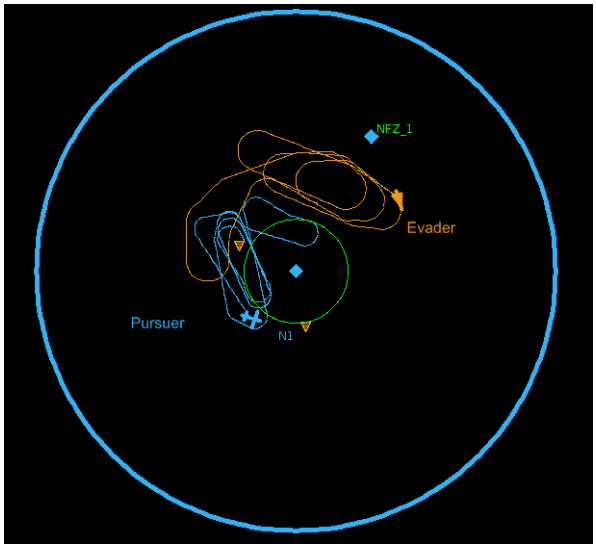

Figure 6.22: Here, one can observe the same type of game but as the Nash equilibrium method is used the switch between two equilibrium causes the move to be more 'square' as opposed to fluent, as we saw it in the utility max method.

Chapter 7

Conclusions

7.1 Summary

We have proposed three approaches for the autonomous target tracking bu UAVs.

- We started by tackling the problem from a very practical point of view for the case of a UAV that uses a camera with a limited FOV. We proposed a simple approach that permits a UAV not only to maximize its observation time but also to deal with NFZs, on the basis of a visual target pose estimation.
- We then proceeded by considering that the target is evasive. We used a predictive, iterative approach for this approach with an adversarial twist: before the UAV decides its future trajectory, it attempts to find the future target trajectory. We considered that the target tries to find the most covert path, by calculating the zones unseeable for the UAV. This scenario has been expanded to the case of two UAVs cooperating with each other, using a hierarchical approach in which the first UAV does not alter its decisions and the second UAV helps to maximize the overall tracking capability.
- Lastly, we tackled the problem from a game theoretic point of view, using a discrete representation of the environment. We used classical tools of game theory: Nash equilibrium, minimax algorithms, utility maximization and Pareto group optimality. This approach was then also expanded for the case of two UAVs.

7.2 Discussion and further work

7.2.1 Iterating on the thesis contributions

This thesis presents three different approaches. The first approach has very little to do with the other two, both of which are a bit more interconnected.

The main flaw of the first approach is that it addresses a very specific and simplified case, and not very scalable to more complex conditions, *e.g.* urban environments. Indeed, as the target moves on an open field and thus the UAV has enough space to perform its trajectories without having to avoid many NFZs. This however is not always the case. Although it may seem very specialized and obvious, it showed to behave better than other existing solutions, and complies with restricted FOVs, a constraint often encountered in UAV designs. Keeping

with a control based approach in an open environment, an interesting extension would be to control the flight altitude, *e.g.* to recover target sight after it escaped the FOV.

The second approach tackles a more complex problem: it can handle more obstacles, covert areas, road networks and occluding obstacles that we find in a real environment. While it addresses mostly evasive targets, it was shown that by omitting the target prediction step, we can also handle a neutral target. Its serious drawback is that it needs a precise UAV and target pose estimate with respect to the environment, which is can be a limit in some applications contexts. It does, however, offer a framework to combine safe navigation and evasive target tracking. A good point is that one does no needs to make an offline optimization before the application starts, a bad point is that it is platform oriented: one needs to know the exact model of the agent before the method is applied in order for the approach to work.

Possible future work in this framework could be the following:

- Integrate a system that estimates the position of the target that can replace the need for full information. This way one UAV would be enough for tracking a ground target without any external support - satellites etc.
- Perform a more in depth study for the case of limited FOVs.
- Assess the application in real flight using cameras that detect the target and can thus provide a realistic test of this approach.
- Consider a more realistic 3D environment model (either for the obstacles and the occluding obstacles), and extend the UAV control to the altitude.

The expansion for the two UAVs case is promising, but there still is plenty space for improvement of the framework by testing other type of cooperation schemes.

We then invested in game theory, aiming at finding more rational decision oriented solutions to the problem. The main advantage of the third approach is that it can be applied to a much larger field of applications than autonomous pursuit for fixed wing UAVs. The approach could be applied to multiple ground robots, to UAVs that can hover or any other form of robot that participates in some form of cooperation or competition. By manipulating the payoff functions one can achieve very different behaviors that can be used in various applications. One positive point of this approach is that it is independent of platform and of the number of players, since one can define payoff functions for individual players or for group of players.

Dealing with a group of players one could use game theoretic tools of much more advanced or complex cooperation schemes than the ones presented in this thesis: the Core, the stable set and the Shapley value come into mind. Similarly, there could exist an expansion of the minimax search depth to more complicated scenarios as well as the capability to apply algorithms as the trappy minimax into aero-terrestrial applications. There is a great potential for using game theoretical approaches to team cooperation or any other type of game that can be encoded to rules. The fact of using computers to find how games are played is not new, but it has not yet been used enough in robotics.

Finally, it should be mentioned that this approach should be tested with real flights, the only way to assess their practical efficiency.

7.2.2 Beyond the horizon of the thesis

During these three years we experimented other solutions without obtaining enough satisfactory results. The first domain of interest is target trajectory prediction: there exist numerous approaches to this problem (*e.g.* grey predictors, artificial neural networks...). These methods have both been proven to work well and they could be used to provide a prediction of the future position of the target without the need to know if it is neutral or evasive.

The second approach is differential game theory. There is a lot of work to be done within this domain: theoretical studies that include shadowed areas calculation could provide an insight of how to predict the movement of a target, application of algorithms that calculate a good position for the UAV under specific game rules and use of well known solutions differential games problems such as the game of two cars.

Appendix A

Implementation

This chapter presents the tools and platforms (hardware and software) we used to achieve the developments and experiments in this thesis.

A.1 The Paparazzi Flying System

A.1.1 Presentation

Figure A.1: The basic elements needed for the Paparazzi system to operate.

Paparazzi [Brisset et al.,] is a free and open-source hardware and software project intended to create a powerful and versatile autopilot system. The project includes not only the airborne hardware and software, from voltage regulators and GPS receivers to Kalman filtering code, but also an ever-expanding array of ground hardware and software including modems, antennas, and a highly evolved user-friendly ground control software interface (Fig. A.1).

All hardware and software is open-source and freely available to anyone under the GNU licensing agreement. Several vendors are currently producing and selling Paparazzi autopilots and popular accessories, making the system easy and affordable to all.

The key feature of the paparazzi autopilot is its unique combination of infrared thermopiles and inertial measurement for attitude sensing, providing a robust attitude estimate that requires no ground calibration and can recover from any launch attitude (Fig. A.4).

A.1.2 Hardware

Controller Board

The early versions of the hardware autopilot card were developed using Atmel-AVR and Philips ARM7 LPC microcontrollers. The actual versions are using one processor (Model Tiny Fig. A.2) or two processors (Model Classix). Every card has input lines for the sensors, output lines for the servos, a radio command receptor (R/C) and a radio modem.

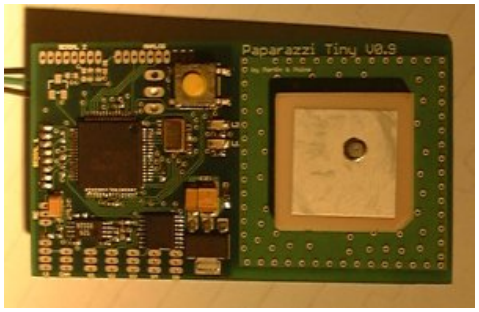

Figure A.2: The hardware card used on board the aircrafts and its GPS.

Sensors and Attitude decomposition

- **Infrared Sensors**: The Paparazzi auto pilot uses thermopiles (Fig. A.3) to estimate the posture of the aircraft by measuring the temperature difference between the sky and the ground .
- **GPS**: The global position of the aircraft is obtained using a GPS antenna.

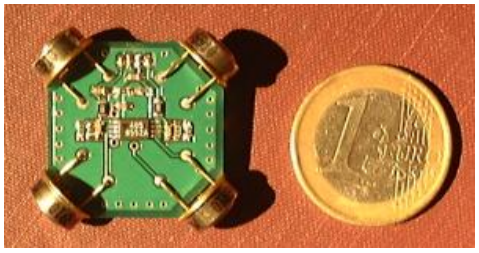

Figure A.3: The posture infrared sensor.

Aircraft

Small R/C off the shelf aircrafts can be modified as follows in order to be able to be used by the Paparazzi autopilot (Fig. A.5):

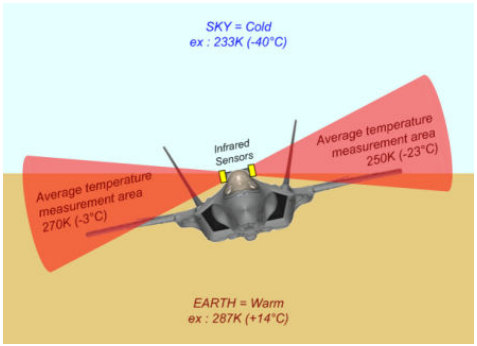

Figure A.4: The principle of function of the sensors: By measuring the temperature difference one can deduce the posture of the aircraft in any axis.

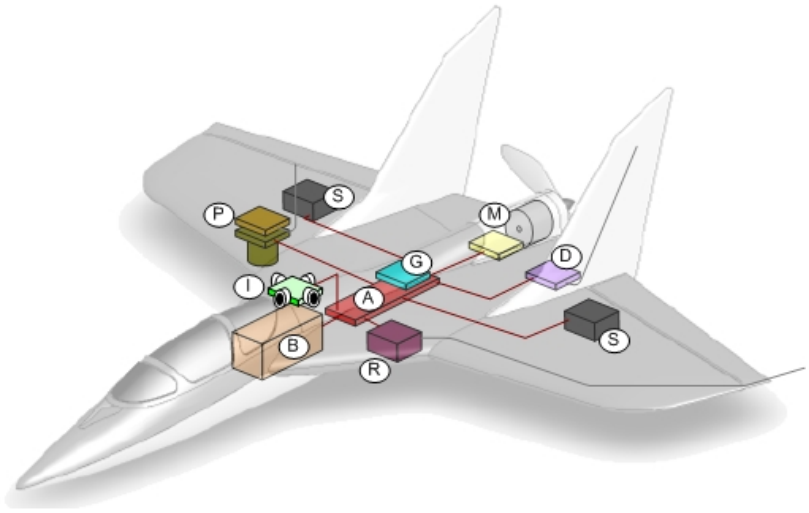

Figure A.5: Configuration of electronics on a Paparazzi aircraft

- Autopilot Control Board
- Battery
- Datalink Radio-Modem and Antenna
- GPS Receiver
- IR Sensor Board
- Motor and Controller
- RC Receiver and Antenna
- Servos
- Payload = Camera and Video Transmitter

A.1.3 Software

On-board Software

The auto-pilot uses three basic loops that can be seen in Fig. A.6 The first stabilizes the aircraft and maintains the desired posture. The second one controls the heading, the altitude and the position of the aircraft. Finally, the third keeps track of different indices like battery level etc.

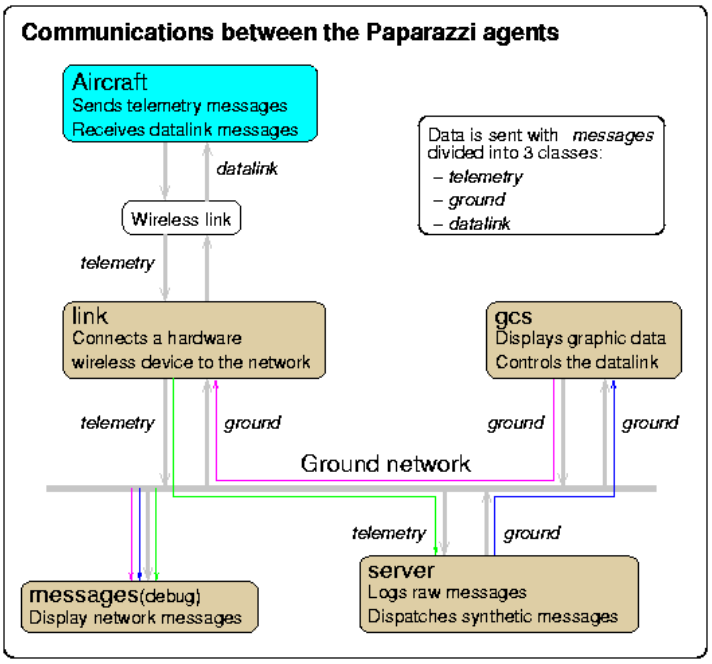

Figure A.6: Architecture and communication links of the Paparazzi system.

Ground Station

A graphic interface permits to control the flight plans of several airplanes simultaneously. The principal display permits to have a quick look at the position of the aircrafts and to fine tune fast the different gains of the system. The principle elements of the ground station are:

- **Link**. The Link enables the communication between the aircraft by using the different available modems.
- **Server**. This agent permits to store the different messages and distribute them to the relevant agents.
- **GCS**. This is the graphic interface of the ground station (Fig. A.7).

All agents communicate between them using an intermediate bus, known as Ivy [1]. It is a simple communication layer that permits connect different agents. This bus served as the connection between our C modules and the Paparazzi autopilot.

[1] http ://www.tls.cena.fr/products/ivy

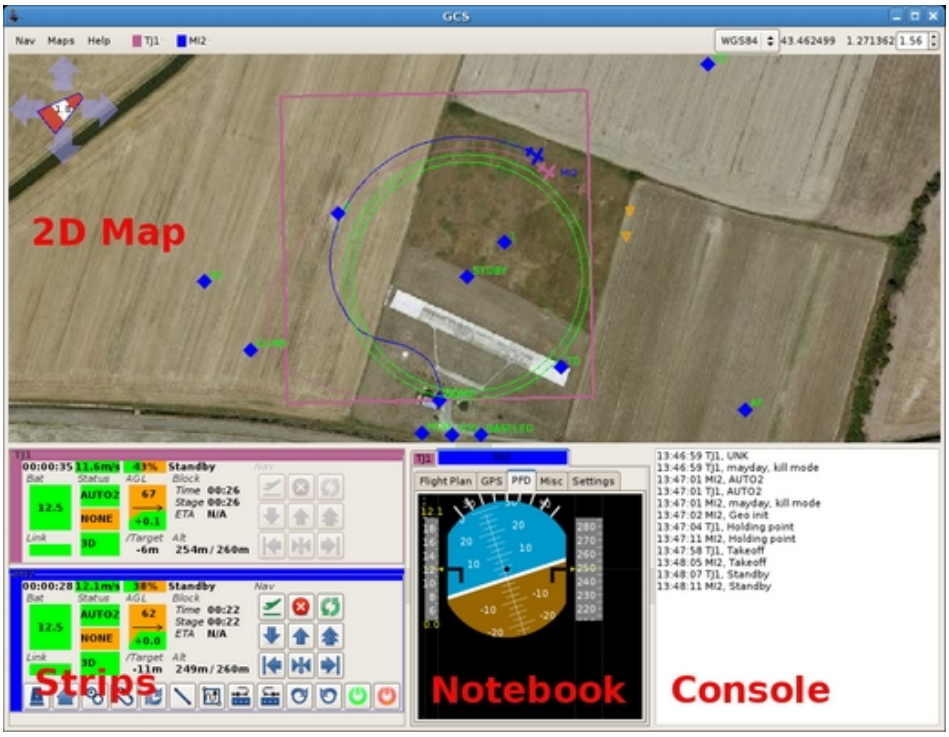

Figure A.7: The graphic interface of the Ground Station.

One very important aspect of the Paparazzi system apart from flying airplanes, is that it can also be used in simulation mode. During this mode, the software allows to simulate very realistic aircraft dynamics and to add randomly generated windy conditions. This greatly diminishes the differences in aircraft behavior from simulation to real flight.

A.2 Development and Simulation Environments

A.2.1 Algorithm encapsulation

Genom

Most of the algorithms developed during this thesis were coded using C ANSI. However, the algorithms were encapsulated using a specific architecture used in LAAS/CNRS that is known as Genom.

Genom [Alami et al., 1998] is a development framework that allows the definition and the production of modules that encapsulate algorithms. A module is a standardized software entity that is able to offer services which are provided by the algorithms. Modules can start or stop the execution of these services, pass arguments to the algorithms and export the data produced. In permits to easily integrate programming code without having to worry about operating system requirements and can greatly accelerate coding time.

Its use involves seven distinct steps (Fig. A.8):

1. Describe the module.
2. Generate the server.

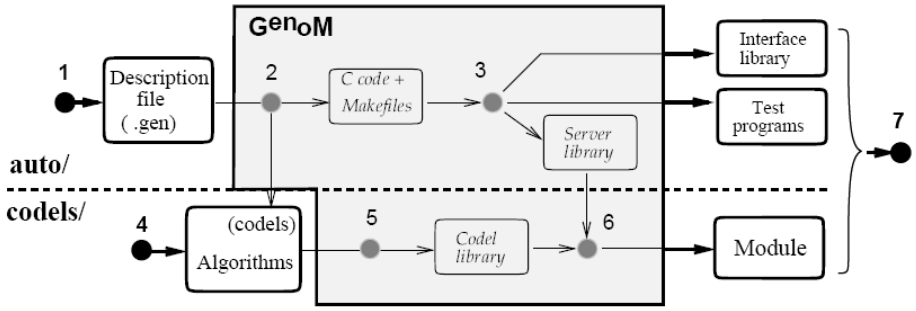

Figure A.8: Genom development cycle.

3. Compile.
4. Write the codels that will invoke the algorithms.
5. Compile the codels.
6. Link with the server.
7. Test and use the module

Each module has a structure that allows it to communicate with its environment either by control requests, by exporting data in a poster, by reading the posters of other modules or by exchanging data through the hardware's I/O ports (Fig. A.9).

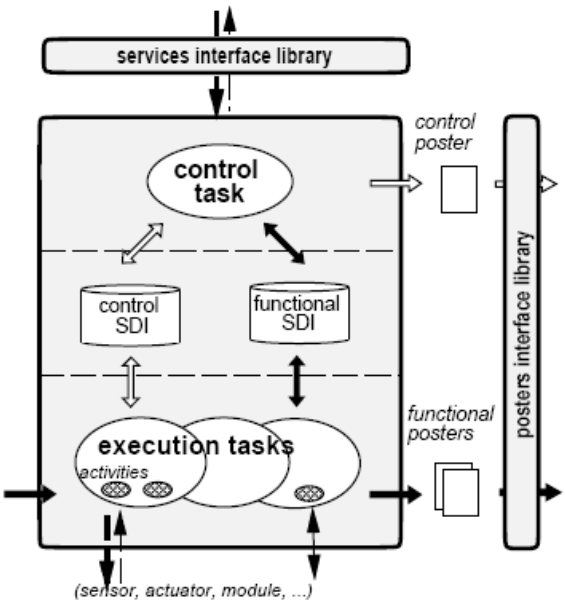

Figure A.9: Module structure.

It should be noted that much of the hierarchical part of the software uses scripts and widgets programmed using the scripting language package of Tcl/Tk [2]. This scripting language permits a fast programming loop as it does not need to be compiled and permits the different software parts to communicate with each other and to synchronize using a high level programming structure.

GDHE

GDHE is a tool for 3D visualization of robotic applications that uses the OpenGL library to display the 3D primitives. In order to program something one needs to use the scripting language of Tcl/Tk.

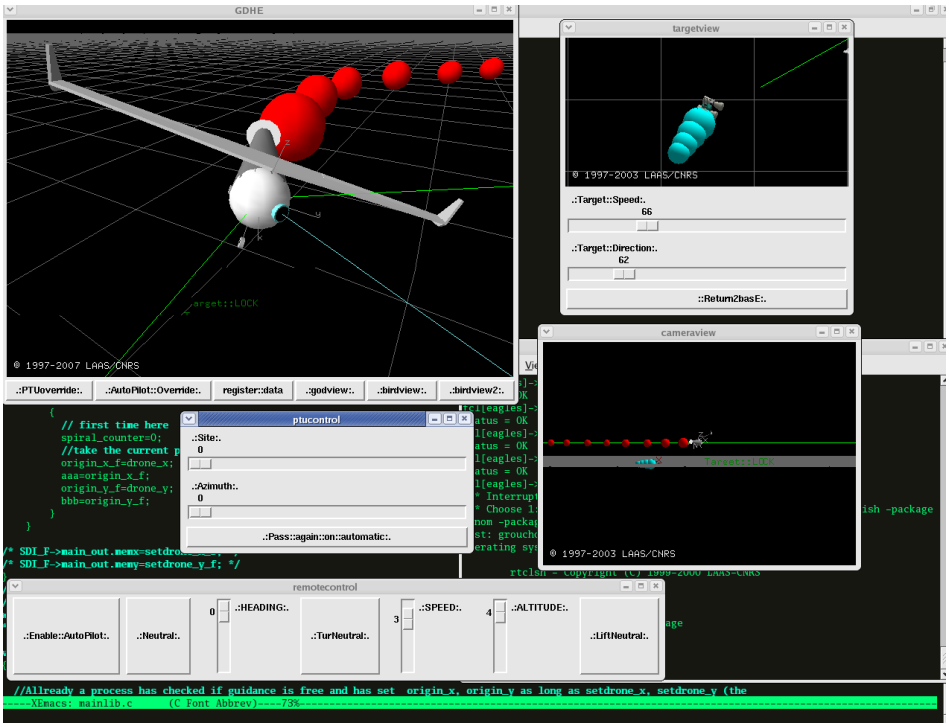

Figure A.10: The GDHE environment.

GDHE allows to build a 3D representation of the geometrical model of an environment and make it change with time. In order to achieve this, GDHE acts as a server that receives requests from a set of client processes. These requests describe the evolution of the model. Clients can be either modules that control a real system and that send data about the state of this system or simulation processes producing a simulated state of a virtual system. .

Overall Software Structure

Overall the architecture structure can be seen in (Fig. A.11). All decision algorithms were encapsulated within the tracking module and passed on their commands to a control and guidance layer. From there they communicated with the Paparazzi autopilot using an IVY

[2]http://tcl.sourceforge.net/

bus. At the same moment a Tcl script was coordinating all actions, by initializing different variables and by exporting all necessary information to the 3D simulation environment of GDHE. Finally, all decisions, simulations and guidance of the target were taking place within the target module.

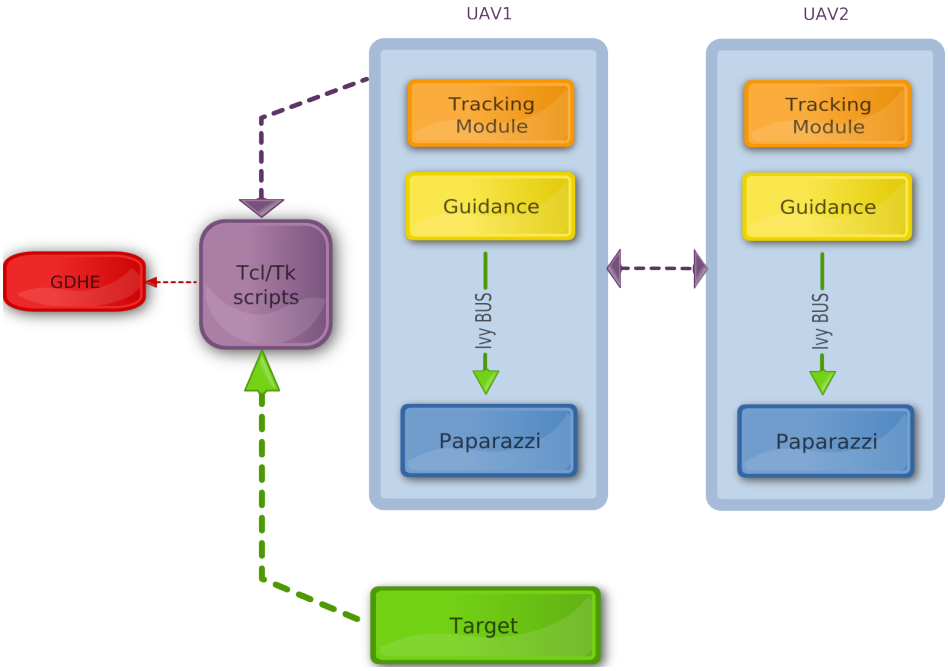

Figure A.11: Overall software organization of the modules.

A.2.2 Scilab simulation platform

Part of the work done in this thesis was done using the Scilab[3] free of cost numerical computational package developed by researchers from the INRIA and the Ecole nationale des ponts et chaussees (ENPC). Using this software, we performed all graphs contained in this thesis, we tested algorithms before we code them in C ANSI and checked that they give satisfying results.

A.3 Aircrafts-Robots

A.3.1 The Nirvana fleet

For most of our experiments we used some of the 3 Minimag aircrafts that the laboratory has obtained from the company Multiplex and we equipped them with the Paparazzi auto pilot system (hardware and software). Each of the aircrafts were connected via radio link with a laptop on the ground that gave all the commands using the software platforms presented earlier. The have a wing span of 1010 mm and a total length of 820 mm. Their stall airspeed is at about 6.1m/s and with a throttle of 70% they will travel with an airspeed of 8.4m/s.

[3]http://www.scilab.org/

Figure A.12: The 3 Minimag model aircrafts known as the Nirvana fleet were the most frequent platforms for testing this thesis algorithms.

Appendix B

Resumé

B.1 Motivations

Pendant les dix dernières annéés, il y a eu une augmentation significative d' intérêt vers les drones. Ce temps-ci , les UAVs peuvent décoller, voler et atterrir sans aucune intervention humaine. Suivant cette tendance, la production mondiale des drones (UAVs) a aussi augmenté: la production mondiale des UAVs a produit 477 types différents de UAVs , parmi eux, quelques sont civils mais la plupart sont militaires, alors que par 2007 cette figure a augmenté un peu prés de 65% en atteignant le chiffre 789 [International, 2007].

Certaines de ces plates-formes sont certainement impressionnantes, ils abordent beaucoup de questions importantes comme la capacité de vol, l'intégration d'espace aérien et les capacités opérationnelles . Cependant ils semblent manquer un point important : l'autonomie. Il existe souvent une idée fausse que comme un UAV a la capacité de voler avec un façon automatique pendant quelques parties de sa mission, que ceci est autonomie. La distinction doit être faite immédiatement ; D'une parte, le vol automatique offre à un système, la capacité d'exécuter des actions selon une série de règles prédétermines dans un environnement fortement prévisible, un exemple classique étant le pilote automatique . Tandis que d'autre part, la vraie autonomie offre à un UAV la capacité d'agir sans aucune surveillance dans un environnement qui change et qui peut agir face a une vaste gamme des conditions. Un système autonome devrait fournir un interface au niveau d'abstraction très élevé, L'utilisateur pourrait commander l'avion a "Explore (Area)" or "GoTo(City)" et le système pourrait être laissé complètement non surveillé afin d'exécuter ces taches. Pour obtenir une idée sur le niveau d'autonomie que les UAVs modernes fournissent, nous devrions remarquer que la plupart des plates-formes aériennes produites de nos jours, le plan après une perte de communication est de commander que l'avion aille à un certain secteur désert et exécutent des cercles jusqu'à ce que la communication soit rétablie ou jusqu'à ce que tout le carburant soit consommé et l'avion tombe.

L'avion s'est développé de l'avion piloté et peu importe leur vitesse et leur capacités a manœuvrer , ils ressemblent beaucoup aux leurs avions pilotés: il y a toujours quelque part un pilote qui prend les décisions et le fait qu'il n'est pas sur l'avion ne change pas beaucoup des choses. D'une part, la recherche de robotique s'est avancée significativement au cours des années passées et il peut maintenant offrir plusieurs solutions élaborées dans le domaine de la vision artificielle, la recherche du chemin, la prise de décision, l'action d'éviter les obstacles et la modélisation d'environnement - tous travaillent en visant à équiper

Figure B.1: Aerosonde: Premier robot aérien de survoler l'océan Atlantique.

les machines mobiles avec l'autonomie. L'un des principales motivations de cette thèse est d'examiner les technologies menant a l'autonomie d'avion et essaient de développer certains d'entre d'eux.

La deuxième motivation vient de la caractéristique de la plupart des applications d'UAVs. Ils sont disponible dans une vaste gamme: la surveillance aérienne, reconnaissance militaire, les actions militaires offensives, la recherche et la sauvetage autonome, le contrôle des lignes qui emmenant le gaz naturelle et la surveillance autonome des incendies des forets .

Le point commun de toutes ces applications consiste en ce qu'ils sont définis par rapport au sol: le repérage du cible terrestre est une tache essentielle pour les UAVs. Pour telles taches, les avions de voilure fixe ont un avantage sur des hélicoptères pour atteindre des grandes vitesses avec une consommation faible , mais leur dynamique contrainte la visibilité de la cible . Cette thèse prend en considération ce problème , et introduit des stratégies variés pour obtenir le repérage visuel de la cible terrestre visant à optimiser la visibilité de cible.

B.2 Les défis et les objectifs

B.2.1 La problématique

Dans l'ensemble, l'objectif de cette thèse est de fournir des méthodes pour équiper un UAV pour tracer automatiquement une cible mobile, terrestre, selon les conditions ci-dessous.

- La dynamique d'UAV: On considéré les UAVs de voilure fixe. Ce type d'avion a la limitation de mouvement et la contrainte dynamique et l'avion se comporte comme un véhicule non-holonome.Un avion d'aile fixé ne peut pas s'arrêter et changer la direction aussi facilement qu'une cible terrestre et cela provoke des difficultés. On considère aussi que le mouvement des UAV est contrainte par la présence de zones de vol interdites.

- La présence des obstacles: Il est improbable que les UAVs soient permis de voler dans tous les régions et dans le cas des micro UAVs volant à une altitude basse , cette supposition est plus souvent vraie. Il faudrait considérer qu'il existent des zones sur lequel l'avion ne peut pas s'aventurer parce qu'il y a de zones de vols interdites (NFZs) ou parce qu'il y a des bâtiments à éviter. Dans chacun des cas mentionnés ci-dessus , on

devrait trouver une méthode de piloter l' UAV d'une telle façon pour qu'il maintienne la visibilité sans entrer dans une NFZ.

- Les limitations du champ de vision: Dans beaucoup des cas, les UAVs et spécialement les micro-UAVs n'ont pas un champ de vue (FOV) menant à un problème de plus grand complexité. Nous examinerons les conséquences de la limitation du champ de vision pour le UAV (panoramique vue entière du champ par rapport a vue de champ limite).

- La dynamique de la cible et le comportement: La cible pourrait être mobile dans un terrain ouvert ou sur un réseau routière , et il a aussi les contraints dynamiques (le modèle utilise dans ce thèse est une voiture). Cela peut être neutre ou évasive : dans le deuxième cas, il peut profiter de la présence des obstacles, nommé "les ombres", afin d'éviter d'être vu par le UAV, faisant le problème apparente au jeu du cache cache .

B.2.2 Les objectives de la thèse

Globallement, c'est nécessaire que le UAV prevoit les mouvements de la cible afin d'être " un pas en avant ". Cela évoque l'intérêt au niveau du processus de la décision : une cible évasive va essayer vivement à cacher derrière des obstacles et utiliser le réseau de la rue d'une manière prévisible. Afin de maximiser le temps de la visibilité, les UAVs doivent être capables de prévoir tells situations et réagir ou se préparer avant qu'il arrive pour maintenir la visibilité.

L'objectif du thèse est de fournir une structure de décision qui optimise la visibilité de la cible face à ces contraints. En plus, on va aussi considérer le cas de deux UAVs qui coopèrent pour la poursuite de la cible.

B.3 La contribution et le plan du thèse

Les contributions principales de cette thèse sont :

- La présentation d'une méthode de la navigation contrôle, qui permet au UAV de tracer une cible terrestre , à l'absence des obstacles et des ombres .

- La présentation d'une méthode prédictive qui permet à un UAV de tracer une cible basée sur la position , en évitant les obstacles et des régions que la cible peut utiliser comme des cachettes. La méthode considére le cas d'une cible évasive qui se déplace, soit dans un environnement tout terrain soit dans un réseau de route et est prolongé vers les cas de deux UAVs coopératives.

- La présentation de l'approche de théorie des jeux qui permet a l'UAV de tracer une cible basée sur la position parmi les NFZs et les obstacles indistincts et l'extension de cette approche de collaboration de l'UAV.

B.4 L'état de l'art

- La premiere partie de recherche est basée sur la Guidace, la Navigation et le Contrôle. Son intérêt principal consiste de changer une trajectoire afin d'atteindre la visibilité maximum de la cible. La plupart des articles scientifiques dans ce domaine s'occupent

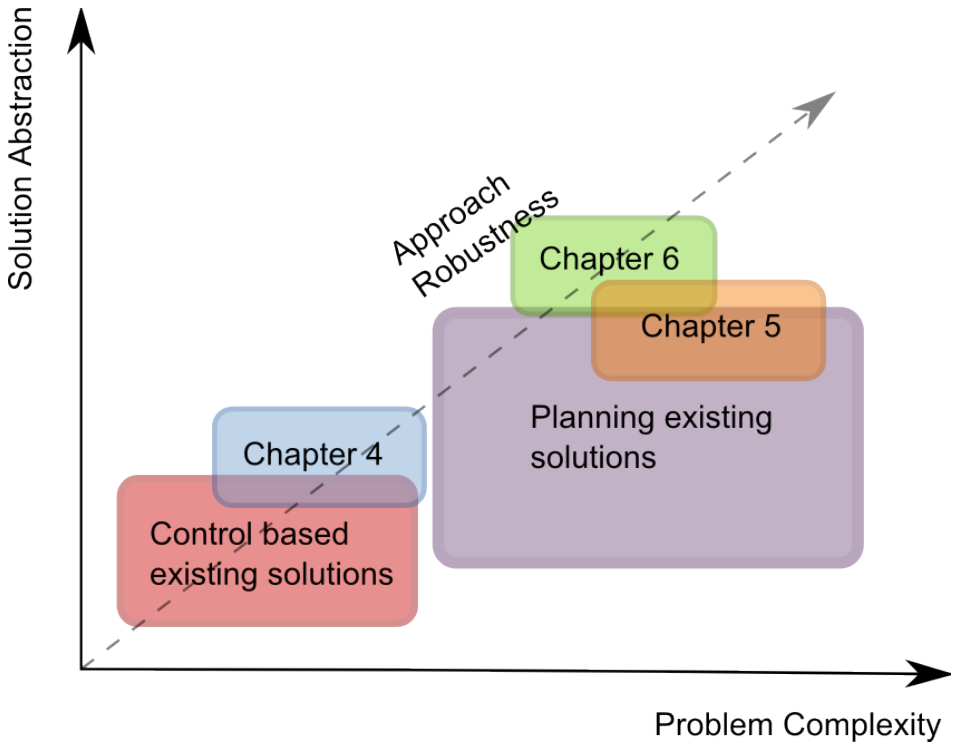

Figure B.2: Graphe des contributions de cette thèse: le graphe démontre la complexité et l abstraction des solutions.

aussi de cet aspect de contrôle du problème . Cependant, peu de ces articles s'occupent de la présence des zones de vols interdites (NFZs) ou d'autre type des obstacles cachant la visibilité. Le scénario typique est le suivant: une cible se déplace dans un terrain ouvert , alors que le UAV vole à une altitude constante et à une vitesse régulié , en essayant d'être proche d'une cible terrestre .

- La deuxième partie de recherche tire ses forces des solutions basées sur le planning et s'approche un pas vers l'autonomie.
- La troisième partie de travail présenté dans ce chapitre consiste la théorie des jeux , celui qui est historiquement l'un des plus influentielles dans le domaine de la prise de décision.
- Ce chapitre serait incomplète sans la présentation de l'environnement des multi-agents et les deux types principales des interactions qui peuvent soulever: les interactions compétitives , dans ce sens , la cible et l'UAV essaient de se jouer au plus fin et les interactions coopératives , dans ce sens, deux ou plus de deux UAVs peuvent coopérer afin d'optimiser une utilité commune.

B.5 Synthèse du problème

Dans l'ensemble, le but de cette thèse est d'équiper un système composé d'un ou deux UAV capable d'optimiser la visibilité ration d'une cible terrestre. Le système de l'UAV doit être

capable de s'adapter ses décisions et sa trajectoire basée sur l'information suivante.

- Le type de caméra qu'il pourrait utiliser, celle qui dans certains cas ne pourrait pas assurer un champ de vision plein.
- La prédiction de la position de la cible terrestre qui peut changer la direction et la vitesse.
- La prédiction de la position de la cible terrestre peut être évasive ou neutre. Non seulement une cible évasive pourrait essayer de maximiser la distance entre les UAVs et la cible , mais il peut aussi utiliser n'importe quel obstacle afin de se cacher des UAVs.
- Une carte de tous les obstacles naturels ou humains terrestres est contruite , qui peuvent empêcher la visibilité de l'air et donc fournit des endroits que la cible peut se cacher des UAVs.
- Une carte en détail de tous les Zones de Vols Interdites que l'on doit éviter .

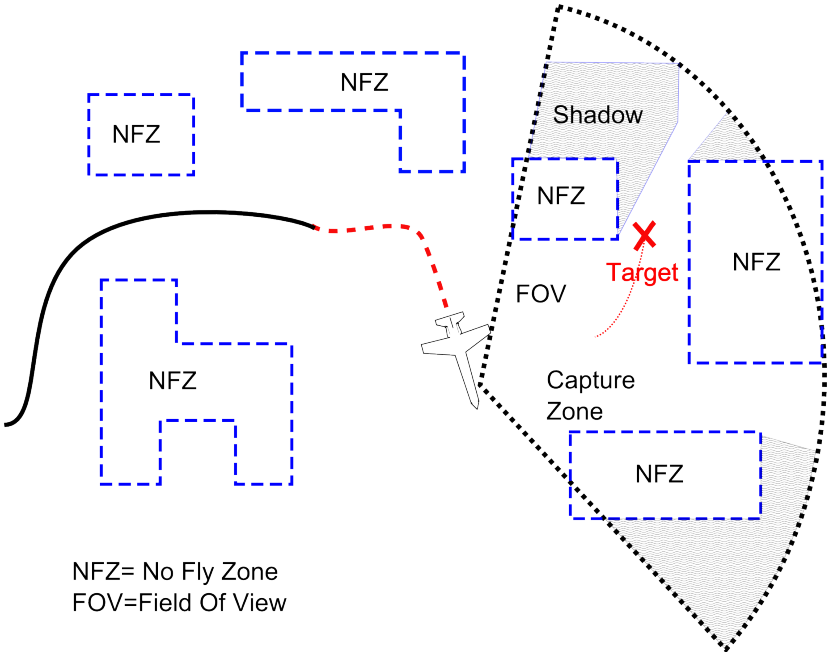

Figure B.3: En réalité l'UAV doit trouver des situations plus difficiles et doit joggler entre NFZs, la proximité de la cible, d'avoir une camera a la bonne angle et que aucun obstacle cache la cible.

B.6 Suivre une cible terrestre

Dans cette partie, nous considérons le cas d'un UAV qui vole a une altitude constante et a une vitesse stable et essaient de suivre la trajectoire d'une cible mobile et neutre . 'Neutre', cela veut dire, la cible n'est pas consciente d'être surveillé, et donc, elle n'essaie pas ni de coopérer ni de échapper d un UAV. Il se déplace terrestre à une vitesse variable. Il n'y a

pas d'obstacles qui empêchent la visibilité et dans la plupart des chapitres , on considérera que l'UAV a la capacité totale de déplacement dans l'espace (pas NFZs). Cependant, le cas de NFZs est traité brevement , pour apporter une méthode qui marche bien dans tous les conditions. Fig. B.4

B.6.1 Vue d'ensemble de notre approche

On considéré deux cas ici:

- **La cible hors de la vue :** Si la cible est hors du champ de vision actuel, le UAV utilise une stratégie simple qui le guide vers la dernière position que la cible était vue la dernière fois . En absence des informations quant à l'emplacement de la cible, on a proposé des stratégies de recherche diverses dans la littérature . (*e.g.* [Quigley et al., 2005a, Wong et al., 2005]).
- **La cible en vue:** Quand la cible est dans la vue de champ actuel , notre approche consiste a guider l' UAV afin d'optimiser la visibilité de la cible. Dans ce but, une stratégie de contrôle en deux étapes est définie :

  1. **Une angle optimale $\eta^\star$ est calculé.** Le mot 'Optimalité' veut dire que $\eta^\star$ est mis en ordre pour définir un commerce entre le distance de la cible , ce qui doit rester dans la $[\rho_{min}, D_{max}]$ portée et l'erreur de dépistage latérale x_f de la cible qui doit être réduite au minimum. D_{max} est le rayon de détection maximal et ρ_{min} c'est le minimum rayon tournant.
  2. **Une loi d'orientation latérale aide le UAV à maintenir le titre désirable $\eta^\star$.** Dans ce but, l'angle actuel η et la distance horizontale à la cible L_1 sont calculés, et ils sont insérés dans un algorithme de contrôle latéral qui définit la commande roulis u_{cmd}.

Fig.B.5 demontre le flow-chart de l'approche.

B.6.2 Strategie

Dans le cas le plus simple, la ou il y a visibilité; on a $\eta^\star = \frac{\pi}{2}$. Ailleurs; $\eta^\star = \alpha \pm \frac{||FOV||}{2}$, ca depend de la coté que la cible est approché. On doit aussi guarantir la distance maximale D_{max}, et le rayon de tournage ρ_{min} qui est contraint peut se satisfaire. (Fig. B.6):

B.7 Une appoche itérative predictive pour suivre la cible

Dans ce chapitre, on considère un UAV qui vole à une altitude constante et à une vitesse stable et essaye de suivre la trajectoire d'une cible évasive terrestre. Naturellement, la cible essayera d'utiliser n'importe quels obstacles terrestre pour se cacher de l'UAV. Tous les obstacles terrestre sont considérés comme dômes d'un rayon fixé. De plus, l'UAV doit respecter la présence de certains NFZs, qu'il devra éviter a tout prix. On considère les UAVs d'avoir une visibilité de type 'sphère' jusqu'à ce que certaine distance maximale.

L'approche fondamentale consiste en un boucle de deux étapes (Fig. B.8):

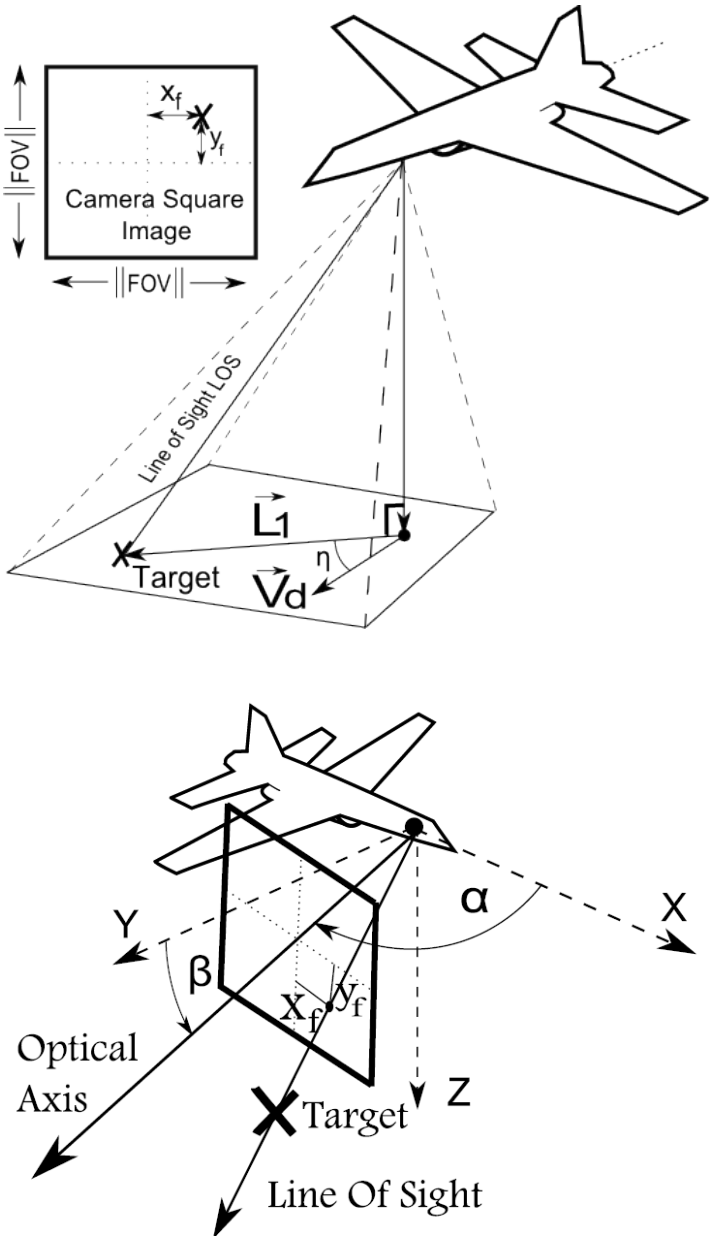

Figure B.4: Géométrie Avion-cible. Top: projection de la ligne de vue au sol et définition de l'angle η. Bottom: orientation des angles (α, β) de l'axe vision de la camera, et des coordonnees des images (x_f, y_f) de la cible.

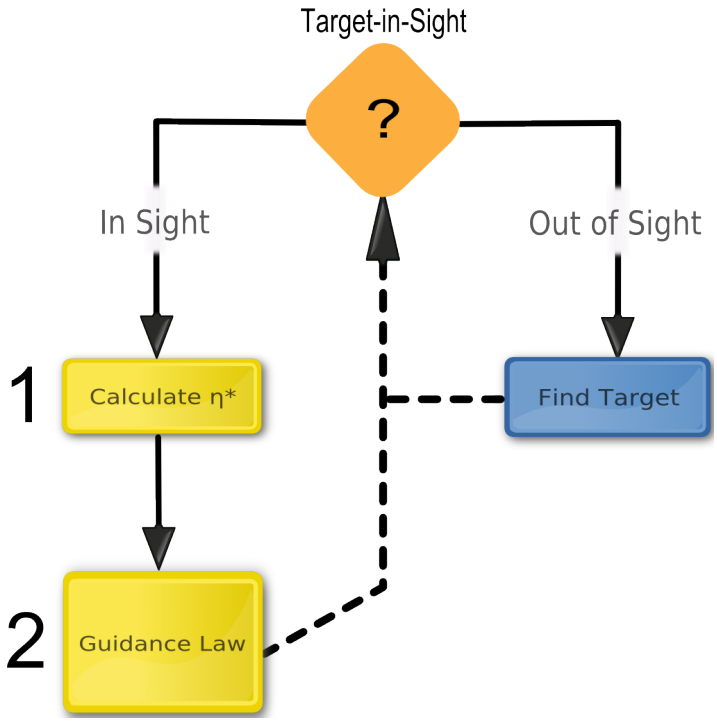

Figure B.5: Flow-chart de l'approche.

- *La prédiction de la cible*: L' UAV essaie de predire la voie de la cible dans l'avenir proche (l'horizon de prédiction) , ou cela utilise sa position réelle (estimée ou bien-connue).
- *La prédiction d'UAV*: Au cours de la prédiction d'horizon, le UAV doit choisir un chemin futur qui maximise la prise de la cible, en respectant la présence d'observation.

B.8 Fonctions d'évaluation

Nous utilisons le mot 'agent' à décrire le UAV ou la cible. Leur position et leur pose sont connu comme l'état x_k de l'agent au moment k. Étant donne la position des objets environnants et les

	$V_t = 0$	$V_t = 0.5V_d$	$V_t = 0.9V_d$
Spiral and SpiralIn	97%	91%	82%
Two Arcs	78%	81%	97%
Circle	67%	69%	81%
Rose Loitering Mode	55%	60%	66%
Target Aim	51%	52%	55%

Table B.1: Ratios de visibilités en simulations pour des vitesses differentes: SpiralI = [Rysdyk, 2003], Arcs =[Stolle and Rysdyk, 2003], Circle=[Stolle and Rysdyk, 2003][Quigley et al., 2005b][Rafi et al., 2006], Rose Loitering Mode = [Lee et al., 2003]

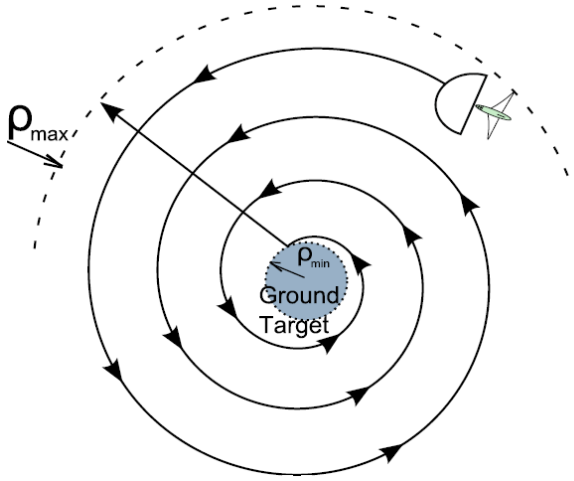

Figure B.6: L strategie spirale.

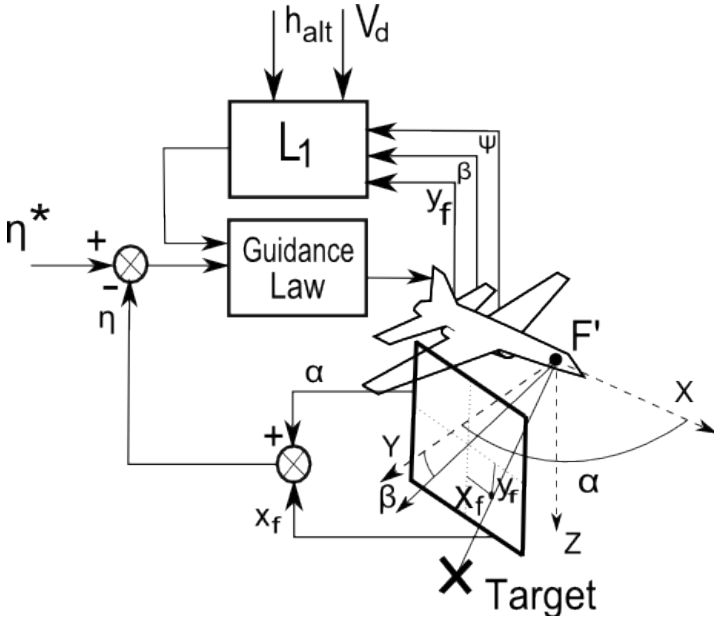

Figure B.7: La boucle d'asservisement laterale.

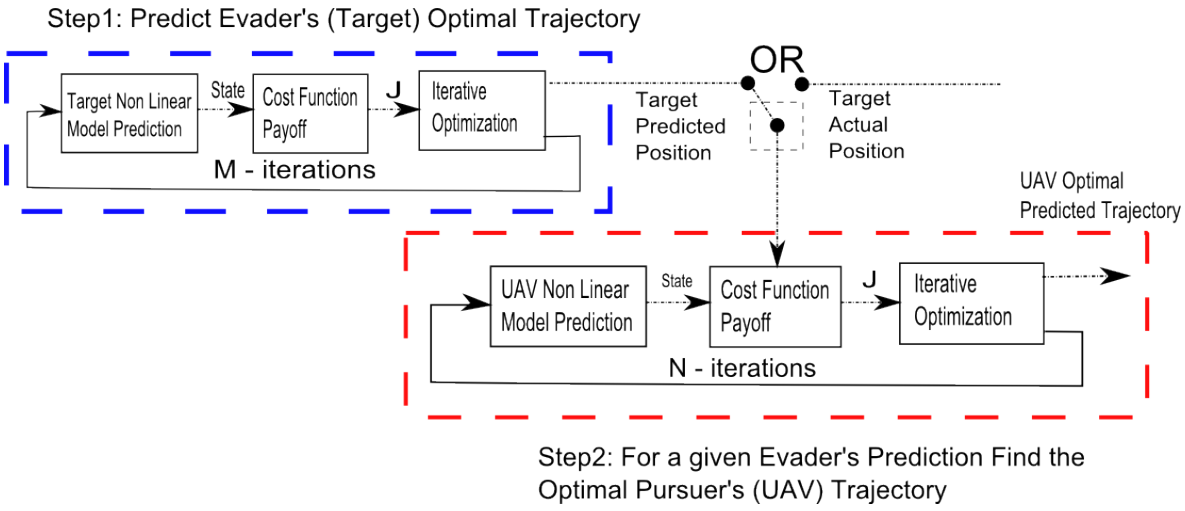

Figure B.8: The algorithm is a two step process. In the first step a prediction of how the target should behave is calculated and according to that information the second step is initiated. During the second step the UAV chooses the optimum path according to the predicted target trajectory and according to a series of criteria that define the cost function. Both steps make use of an iterative optimization scheme where different paths are compared in terms of payoff value (cost function).

agents en relation avec cet agent, on peut attribuer une valeur numérique à cette configuration de la cible d'UAV. Ce coût de la configuration est connu comme $J(x_k^{UAV}, x_k^{Target})$.

Maintenant, selon la méthode que l'on veut assigner une valeur aux configurations, on peut guider les agents à poursuivre certaines configurations, en évitant d'autres méthodes. Ce type de codage peut promouvoir n'importe quel comportement simplement en recomposant les configurations souhaitables et en pénalisant les contraires . Une valeur petite entrainera à une meilleure repérage d'une cible.

- Fonctions d'évaluations d'évasion de NFZ.
- UAV / Fonction d'evaluations de la distance cible.
- Fonction qui protege le drone de configuration ou la visibilite est perdue.
- L'orientation correct de FOV. La fonction qui est utile si l'UAV n'a pas de FOV entière.

B.8.1 Coût de configuration

À chaque itération un coût est calculé pour ce manoeuvre. Le coût général estime pour ce chemin spécifique consiste en la somme de la valeur de tous les coûts de chaque point.

B.8.2 Le coût de configuration dans l'ensemble

Le but de cette étape est de trouver une série des commandes de roulis qui minimisent le coût de fonction J_p pour un horizon de prédiction donné. Une fois que ce chemin est trouvé , la

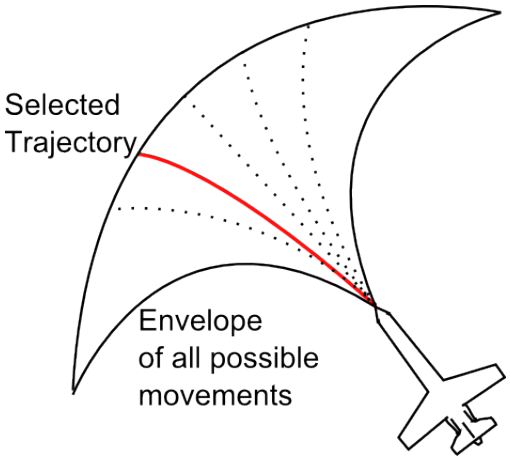

Figure B.9: The Greedy Method: Different trajectories are projected in the future and the one with the least cost is selected

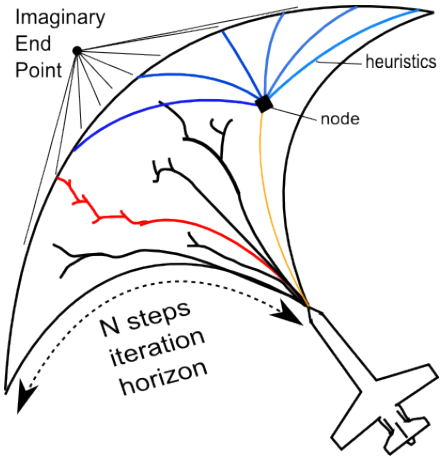

Figure B.10: The A* Method applied to UAV path planning.

première étape de ce chemin est passée comme une commande au UAV et la procedure est répété .

Dans l'ensemble, on étudie trois types de cibles:

- Une cible qu'on connait que sa position actuelle(estimée).
- Une cible qui se déplace sur un réseau routiere (Urbaine et semi-urbaine)
- Une cible qui se deplace dans un éspace ouvert (les véhicules tout terrain) dans la présence des obstacles terrestre.

Il n'y a pas de doute que n'importe quel serait le niveau d'efficacité de stratégies de mono-UAVs, il y aura toujours certaines configurations où la visibilité est perdue. Il est aussi très évident que l'introduction de l'un ou plus UAVs augmentera la visibilité de cible. Cette partie propose une rallonge de notre approche aux cas de plusieurs UAVs. (Fig. B.12).

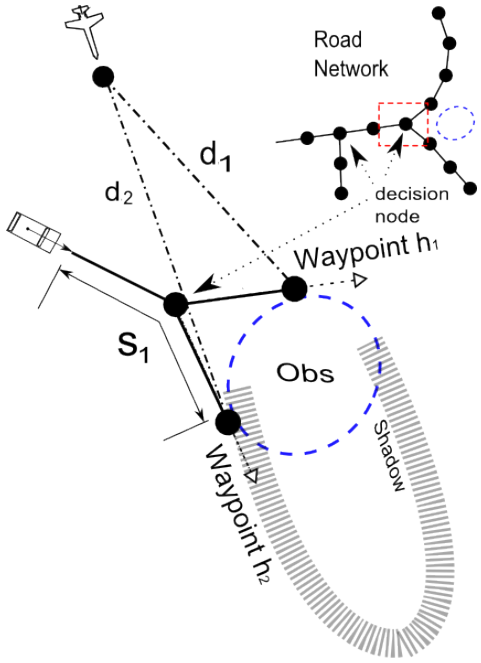

Figure B.11: *Decision-Waypoints*: La cible essaye de decider quel chemin elle doit prendre (h_1 or h_2). Sa decision est basée sur la possibilité de mieux se cacher.

B.9 Une approache pour suivre la cible basée sur la théorie de jeux

Cette partie présente une vue d'ensemble de la méthode proposée . La méthode appliqué à chaque joueur consiste en une boucle de trois étapes distinctes .
(Fig. B.13):

1. *La discrétisation des espaces des jeux:* Cette étape consiste de créer un monde virtuel discret basé sur la configuration du monde réel pour qu'une analyse de jeu théorique puisse se passer. Cette étape considère comme les données les positions de poursuiveurs , d'obstacles, NFZs et comme le rendement une carte virtuelle discrète qui sera connu comme le tableau de jeu. (Fig. B.14). Cette carte est locale et est remis a jour quand l'UAV se déplace.

2. *L'analyse et la décision :* Cette étape analyse le tableau, compare la proportion de forces et décide un manœuvre optimal pour chaque agent , c'est a dire, si l'architecture est utilisée pour un UAV seul ,on va calculer un manœuvre optimal pour cet UAV seul.

3. *L'exécution de commande discrète:* Ce bloc produit une commande de roulis pour l'UAV(ou une commande tournant si l'algorithme est utilise pour la cible terrestre qui s'échappe) ce qui va apporter à la region souhaitable.

B.9.1 La prise de la décision pour un seul avion

Cette partie présente les solutions pour un poursuivant suivant une cible terrestre. Donc, les mots 'équipe' et 'joueur' ont exactement le même sens . L'objectif de cette partie est d'utiliser

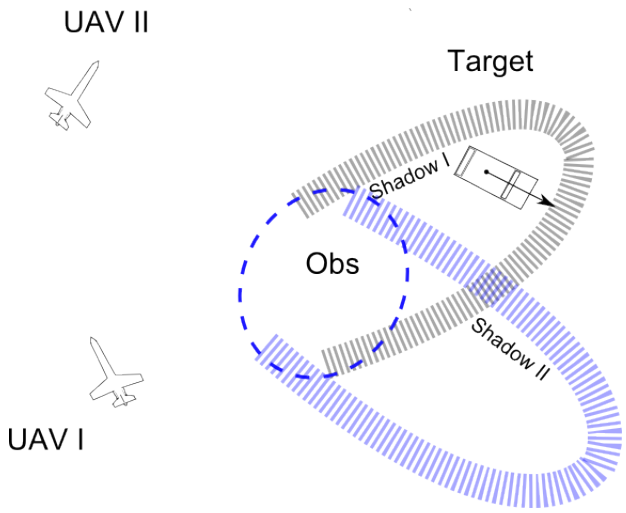

Figure B.12: La cooperation multi drones: Chaque fois que UAV I va perdre la cible, UAV II guarantiee la visibilité.

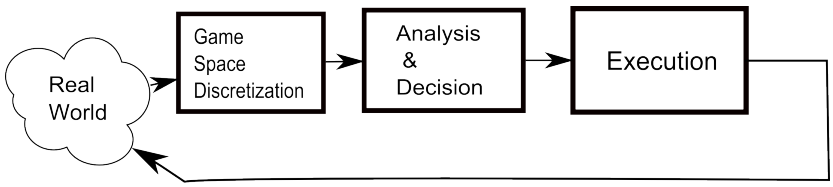

Figure B.13: La approche en trois étapes.

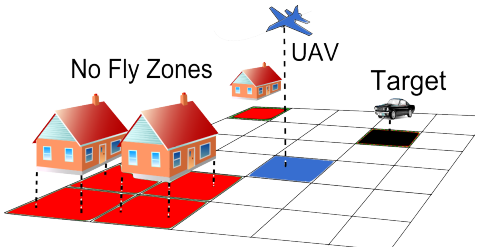

Figure B.14: Le tableau du jeu.

de différents outils de théorie de jeux afin de prendre des décisions pour chaque équipe .

- **La maximisation d'utilité** C'est la méthode la plus simple: Le poursuivant examine tous les autres possibilités (les cellules sur le tableau) et décide d'utiliser celle qui offrir le plus haut récompense possible argmax $argmax(P_P)$ en utilisant sa fonction d'évaluation basée sur la position actuelle de la cible .
- **La recherche antagonique** La méthode la plus encourageant dans le domaine de la recherche antagonique est l'algorithme minimal comme est déjà présenté dans le chapitre de la technologie de pointe. En utilisant cette méthode , soit on peut résoudre l'arbre en utilisant les méthodes locales, c'est a dire , l'examination des destinations probables, seulement les cellules a cote de cellule en question, soit considéré tous les cellules comme

les options.

- **Nash equilibria et les stratégies de dominance** Trouver le Nash équilibre est considéré d'êtree NP hard ,même si cela n'est pas prouve encore[?]. L'algorithme utilisé dans ce thèse peut être se trouver dans [Porter et al., 2008]. . L'algorithme utilisé dans ce thèse peut être se trouver dans [Sureka and Wurman, 2005] - et il est présenté dans la partie 2.2.2. l'algorithme est un tabou, gradient algorithme de recherche basé sur la matrice de jeu.

B.9.2 Decision pour drônes multiples

Ici les drônes cooperent en utilisant:

- La maximisation d'utilité
- La recherche antagonique
- Une approche d'optimalité Pareto

Bibliography

[Alami et al., 1998] Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. (1998). An Architecture for Autonomy. *The International Journal of Robotics Research*, 17(4):315.

[Barraquand et al., 1992] Barraquand, J., Langlois, B., and Latombe, J. (1992). Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241.

[Beard and McLain, 2003] Beard, R. and McLain, T. (2003). Multiple UAV cooperative search under collision avoidance and limited range communication constraints. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1.

[Beard et al., 2002] Beard, R., McLain, T., Goodrich, M., and Anderson, E. (2002). Coordinated target assignment and intercept for unmanned air vehicles. *Robotics and Automation, IEEE Transactions on*, 18(6):911–922.

[Berg, 1983] Berg, R. (1983). Estimation and prediction for maneuvering target trajectories. *Automatic Control, IEEE Transactions on*, 28(3):294–304.

[Bopardikar et al., 2007] Bopardikar, S., Bullo, F., and Hespanha, J. (2007). Sensing limitations in the Lion and Man problem. *Proceedings of American Control Conference, New York*, page 5958.

[Borenstein and Koren, 1990] Borenstein, J. and Koren, Y. (1990). Real-time obstacle avoidance for fast mobile robots in cluttered environments. *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 572–577.

[Borenstein and Koren, 1991] Borenstein, J. and Koren, Y. (1991). The vector field histogram-fast obstacle avoidance for mobile robots. *Robotics and Automation, IEEE Transactions on*, 7(3):278–288.

[Bortoff et al., 2000] Bortoff, S., Center, U., and Hartford, E. (2000). Path planning for UAVs. *American Control Conference, 2000. Proceedings of the 2000*, 1(6):364–368.

[Bourquardez and Chaumette, 2007] Bourquardez, O. and Chaumette, F. (2007). Visual servoing of an airplane for auto-landing. *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1314–1319.

[Brisset et al.,] Brisset, P., Drouin, A., Gorraz, M., Huard, P., and Tyler, J. The Paparazzi Solution. http://paparazzi.enac.fr.

[Bruce and Veloso, 2003] Bruce, J. and Veloso, M. (2003). Real-Time Randomized Path Planning for Robot Navigation. *Lecture Notes in Computer Science*, pages 288–295.

[Bui et al., 1994] Bui, X., Souerès, P., Boissonnat, J., and Laumond, J. (1994). The shortest path synthesis for non-holonomic robots moving forward. Technical report, INRIA.

[Burgard et al., 2002] Burgard, W., Moors, M., and Schneider, F. (2002). Collaborative Exploration of Unknown Environments with Teams of Mobile Robots. *Lecture Notes in Computer Science*, pages 52–70.

[Campbell et al., 2002] Campbell, M., Hoane, A., and Hsu, F. (2002). Deep Blue. *Artificial Intelligence*, 134(1-2):57–83.

[Campbell and Ousingsawat, 2002] Campbell, M. and Ousingsawat, J. (2002). On-line Estimation and Path Planning for Multiple Vehicles in an Uncertain Environment. In *Proceedings of the 2002 AIAA Conference on Guidance*.

[Chandler et al., 2001] Chandler, P., Pachter, M., and Rasmussen, S. (2001). UAV cooperative control. *American Control Conference, 2001. Proceedings of the 2001*, 1.

[Chitsaz and LaValle, 2007] Chitsaz, H. and LaValle, S. (2007). Time-optimal paths for a dubins airplane. *Decision and Control, 2007 46th IEEE Conference on*, pages 2379–2384.

[Chong et al., 2000] Chong, C., Garren, D., Grayson, T., Allen, B., and Inc, H. (2000). Ground target tracking-a historical perspective. In *Aerospace Conference Proceedings, 2000 IEEE*, volume 3.

[D.Ghose, 1994] D.Ghose (1994). True proportional navigation with maneuvering target. *Aerospace and Electronic Systems, IEEE Transactions on*, 30(1):229–237.

[Dogan, 2003] Dogan, A. (2003). Probabilistic approach in path planning for UAVs. *Intelligent Control. 2003 IEEE International Symposium on*, pages 608–613.

[Dubins, 1957] Dubins, L. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516.

[Ducard et al., 2007] Ducard, G., Kulling, K., and Geering, H. (2007). A simple and adaptive on-line path planning system for a uav. *Control and Automation, 2007. MED '07. Mediterranean Conference on*, pages 1–6.

[Edelkamp and Schrodl, 2003] Edelkamp, S. and Schrodl, S. (2003). Route Planning and Map Inference with Global Positioning Traces. *Lecture Notes in Computer Science*, pages 128–151.

[Eppstein, 1994] Eppstein, D. (1994). Finding the k Shortest Paths. In *Annual Symposium on Foundations of Computer Science*, volume 35, pages 154–154. IEEE Computer Society Press.

[F.Belkhouche and B.Belkhouche, 2007] F.Belkhouche and B.Belkhouche (2007). Wheeled mobile robot navigation using proportional navigation. *Advanced Robotics*, 21(3-4):395–420.

[Frew and Lawrence, 2005] Frew, E. and Lawrence, D. (2005). Cooperative stand-off tracking of moving targets by a team of autonomous aircraft. In *AIAA Guidance, Navigation, and Control Conference*.

[Fudenberg and Tirole, 1991] Fudenberg, D. and Tirole, J. (1991). *Game Theory*. Mit Press.

[Ganapathy and Passino, 2003] Ganapathy, S. and Passino, K. (2003). Agreement strategies for cooperative control of uninhabited autonomous vehicles. In *American Control Conference, 2003. Proceedings of the 2003*, volume 2.

[Gendreau, 2003] Gendreau, M. (2003). An Introduction to Tabu Search. *International Series In Operations Research and Management Science*, pages 37–54.

[Gennery, 1999] Gennery, D. (1999). Traversability Analysis and Path Planning for a Planetary Rover. *Autonomous Robots*, 6(2):131–146.

[Gerkey et al., 2006] Gerkey, B., Thrun, S., and Gordon, G. (2006). Visibility-based Pursuit-evasion with Limited Field of View. *The International Journal of Robotics Research*, 25(4):299.

[Girard et al., 2004] Girard, A., Howell, A., and Hedrick, J. (2004). Border patrol and surveillance missions using multiple unmanned air vehicles. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1.

[Glover and Laguna, 1993] Glover, F. and Laguna, M. (1993). Tabu search, Modern heuristic techniques for combinatorial problems.

[G.M.Siouris, 2004] G.M.Siouris (2004). *Missile Guidance and Control Systems*. Springer-Verlag New-York, Inc.

[Gordon et al., 2006] Gordon, V., Reda, A., and CSU, S. (2006). Trappy Minimax-using Iterative Deepening to Identify and Set Traps in Two-Player Games. *Computational Intelligence and Games, 2006 IEEE Symposium on*, pages 205–210.

[Grocholsky et al., 2000] Grocholsky, B., Durrant-Whyte, H., and Gibbens, P. (2000). An Information-Theoretic Approach to Decentralized Control of Multiple Autonomous Flight Vehicles. In *Proceedings of SPIE*, volume 4196, pages 348–359. SPIE.

[Hattenberger, 2008] Hattenberger, G. (2008). *Vol en formation sans formation: contrôle et planification pour le vol en formation des avions sans pilote*. PhD thesis, University of Toulouse.

[Helble and Cameron, 2007] Helble, H. and Cameron, S. (2007). 3-D Path Planning and Target Trajectory Prediction for the Oxford Aerial Tracking System. *Proceedings of the IEEE International Conference of Robotics and Automation (ICRA'07)*.

[International, 2007] International, U. (2007). Unmanned aircraft systems - the global perspective 2007/2008.

[Isaacs, 1999] Isaacs, R. (1999). *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Courier Dover Publications.

[Isler et al., 2005] Isler, V., Sun, D., and Sastry, S. (2005). Roadmap Based Pursuit-Evasion and Collision Avoidance. *Proc. Robotics, Systems, & Science*.

[Jin et al., 2003] Jin, Y., Minai, A., and Polycarpou, M. (2003). Cooperative real-time search and task allocation in UAV teams. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 1.

[Jung and Sukhatme, 2001] Jung, B. and Sukhatme, G. (2001). A Region-based Approach for Cooperative Multi-Target Tracking in a Structured Environment. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[Kahneman and Tversky, 2004] Kahneman, D. and Tversky, A. (2004). Prospect theory: an analysis of decision under risk. *Preference, Belief, and Similarity: Selected Writings*.

[Kim and Shim, 2003] Kim, H. and Shim, D. (2003). A flight control system for aerial robots: algorithms and experiments. *Control Engineering Practice*, 11(12):1389–1400.

[Kim and Kim, 2008] Kim, J. and Kim, Y. (2008). Moving ground target tracking in dense obstacle areas using UAVs. *Proceedings of the 17th World Congress*.

[Kim et al., 2008] Kim, Y., Gu, D., and Postlethwaite, I. (2008). Real-time path planning with limited information for autonomous unmanned air vehicles. *Automatica*, 44(3):696–712.

[Kitamura et al., 1993] Kitamura, Y., Chauang, Z., Tatsumi, S., Okumoto, T., and Deen, S. (1993). A cooperative search scheme for dynamic problems. In *Systems, Man and Cybernetics, 1993.'Systems Engineering in the Service of Humans', Conference Proceedings., International Conference on*, pages 120–125.

[Koren and Borenstein, 1991] Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404.

[Korf, 1985] Korf, R. (1985). Depth-First Iterative-Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, 27(1):97–109.

[Kuffner Jr and LaValle, 2000] Kuffner Jr, J. and LaValle, S. (2000). RRT-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2.

[Lacroix et al., 2002] Lacroix, S., Mallet, A., Bonnafous, D., Bauzil, G., Fleury, S., Herrb, M., and Chatila, R. (2002). Autonomous Rover Navigation on Unknown Terrains: Functions and Integration. *The International Journal of Robotics Research*, 21(10-11):917.

[Langer et al., 1994] Langer, D., Rosenblatt, J., and Hebert, M. (1994). A behavior-based system for off-road navigation. *Robotics and Automation, IEEE Transactions on*, 10(6):776–783.

[Latombe, 1991] Latombe, J. (1991). *Robot Motion Planning*. Kluwer Academic Publishers.

[LaValle, 1998] LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. *Computer Science Dept, Iowa State University, Tech. Rep. TR*, pages 98–11.

[LaValle and Hinrichsen, 2001] LaValle, S. and Hinrichsen, J. (2001). Visibility-based pursuit-evasion: the case of curved environments. *Robotics and Automation, IEEE Transactions on*, 17(2):196–202.

[Lee et al., 2003] Lee, J., Huang, R., Vaughn, A., Xiao, X., Hedrick, J., Zennaro, M., and Sengupta, R. (2003). Strategies of Path-Planning for a UAV to Track a Ground Vehicle. *AINS Conference*.

[Lemke and Howson, 1964] Lemke, C. and Howson, J. (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423.

[Letchner et al., 2006] Letchner, J., Krumm, J., and Horvitz, E. (2006). Trip Router with Individualized Preferences (TRIP): Incorporating Personalization into Route Planning. In *Proceedings of the International Conference on Artificial Intelligence*, volume 21, page 1795. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Li and Payandeh, 2003a] Li, Q. and Payandeh, S. (2003a). Multi-agent cooperative manipulation with uncertainty: a neural net-based game theoretic approach. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3.

[Li and Payandeh, 2003b] Li, Q. and Payandeh, S. (2003b). Planning for dynamic multiagent planar manipulation with uncertainty: a game theoretic approach. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 33(5):620–626.

[Lipschutz, 1968] Lipschutz, S. (1968). *Schaum's Outline of General Topology*. McGraw-Hill.

[Liu et al., 2003] Liu, Y., Simaan, M., and Cruz Jr, J. (2003). Game theoretic approach to cooperative teaming and tasking in the presence of an adversary. In *American Control Conference, 2003. Proceedings of the 2003*, volume 6.

[L.M.Hocking, 2001] L.M.Hocking (2001). *Optimal Control: An Introduction to the theory with applications*. Oxford University Press.

[Luo et al., 2001] Luo, R., Chen, T., and Su, K. (2001). Target tracking using a hierarchical grey-fuzzy motion decision-making method. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 31(3):179–186.

[Marsland et al., 1991] Marsland, T., of Computing Science, D., and of Alberta, U. (1991). *Computer Chess and Search*. Dept. of Computing Science, University of Alberta.

[Marzouqi and Jarvis, 2003a] Marzouqi, M. and Jarvis, R. (2003a). Covert path planning for autonomous robot navigation in known environments. *Proc. Australasian Conference on Robotics and Automation, Brisbane*.

[Marzouqi and Jarvis, 2003b] Marzouqi, M. and Jarvis, R. (2003b). Covert robotics: Covert path planning in unknown environments. *Proceedings of the Australasian Conference on Robotics and Automation, Canberra, Australia*.

[Maza and Ollero, 2004] Maza, I. and Ollero, A. (2004). Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *7th International Symposium on Distributed Autonomous Robotic Systems*. Springer.

[McLain et al., 2002] McLain, T., Beard, R., and Kelsey, J. (2002). Experimental Demonstration of Multiple Robot Cooperative Target Intercept.

[McLain et al., 2000] McLain, T., Chandler, P., and Pachter, M. (2000). A decomposition strategy for optimal coordination of unmanned airvehicles. In *American Control Conference, 2000. Proceedings of the 2000*, volume 1, pages 369–373.

[M.Guelman, 1971] M.Guelman (1971). A qualitative study of proportional navigation. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-7(4):637–643.

[Mitchell et al., 2005] Mitchell, I., Bayen, A., and Tomlin, C. (2005). A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *Automatic Control, IEEE Transactions on*, 50(7):947–957.

[Muppirala et al., 2005] Muppirala, T., Hutchinson, S., and Murrieta-Cid, R. (2005). Optimal Motion Strategies Based on Critical Events to Maintain Visibility of a Moving Target. *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3826–3831.

[Nash, 1950] Nash, J. (1950). Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49.

[Nash, 1988] Nash, J. (1988). Non-cooperative games. *Cournot Oligopoly: Characterization and Applications*.

[Niculescu, 2001] Niculescu, M. (2001). Lateral track control law for aerosonde uav. Aerospace Sciences Meeting and Exhibit, 39th, Reno, NV.

[Norvig, 1992] Norvig, P. (1992). *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann.

[Olfati-Saber, 2007] Olfati-Saber, R. (2007). Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility. In *Proc. of the 2006 American Control Conference, July*.

[Pannetier, 2006] Pannetier, B. (2006). *Fusion des données pour la surveillance du champ de bataille*. PhD thesis, Univ. Joseph Fourier - Grenoble 1.

[Park, 2004] Park, S. (2004). *Avionics and control system development for mid-air rendezvous of two unmanned aerial vehicles*. PhD thesis, MIT.

[Park et al., 2004] Park, S., Deyst, J., and How, J. (2004). A new nonlinear guidance logic for trajectory tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Providence, Rhode Island.

[Parker, 2002] Parker, L. (2002). Distributed Algorithms for Multi-Robot Observation of Multiple Moving Targets. *Autonomous Robots*, 12(3):231–255.

[Payeur et al., 1995] Payeur, P., Le-Huy, H., and Gosselin, C. (1995). Trajectory prediction for moving objects using artificial neural networks. *Industrial Electronics, IEEE Transactions on*, 42(2):147–158.

[Pearson,] Pearson, L. Developing the Flying Bomb.

[Pettersen and Lefeber, 2001] Pettersen, K. and Lefeber, E. (2001). Way-Point Tracking Control of Ships. *IEEE Conference on Decision and Control*, 1:940–945.

[Pimenta et al., 2005] Pimenta, L. C. A., Fonseca, R. R., Pereira, G. A. S., Mesquita, R. C., Silva, E. J., Caminhas, W., and Campos, M. F. M. (2005). On computing complex navigation functions. In *in Proc. IEEE Int. Conf. Robot. Automat., 2005*, pages 3463–3468.

[Porter et al., 2008] Porter, R., Nudelman, E., and Shoham, Y. (2008). Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior*, 63(2):642–662.

[Quigley et al., 2005a] Quigley, M., Barber, B., Griffiths, S., Goodrich, M., and UT, B. Y. U. P. (2005a). Towards Real-World Searching with Fixed-Wing Mini-UAVs.

[Quigley et al., 2005b] Quigley, M., Goodrich, M., Griffiths, S., Eldredge, A., and Beard, R. (2005b). Target Acquisition, Localization, and Surveillance Using a Fixed-Wing Mini-UAV and Gimbaled Camera. *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2600–2605.

[Rafi et al., 2006] Rafi, F., Khan, S., Shafiq, K., and Shah, M. (2006). Autonomous target following by unmanned aerial vehicles. *Proceedings of SPIE*, 6230:325–332.

[Rathinam and Sengupta, 2004] Rathinam, S. and Sengupta, R. (2004). A safe flight algorithm for unmanned aerial vehicles. In *Aerospace Conference, 2004. Proceedings. 2004 IEEE*, volume 5.

[Ravela et al., 1994] Ravela, S., Weiss, R., Draper, B., Pinette, B., Hanson, A., and Riseman, E. (1994). Stealth navigation: Planning and behaviors. *Proceedings of ARPA Image Understanding Workshop*, 10931100.

[Reimann, 2007] Reimann, J. (2007). *Using Multiplayer Differential Game Theory to Derive Efficient Pursuit-Evasion Strategies for Unmanned Aerial Vehicles*. PhD thesis, Georgia Institute of Technology.

[Richards and How, 2003] Richards, A. and How, J. (2003). Model predictive control of vehicle maneuvers with guaranteed completion time and robust feasibility. *American Control Conference, 2003. Proceedings of the 2003*, 5.

[Ridley et al., 2003] Ridley, M., Nettleton, E., Goktogan, A., Brooker, G., Sukkarieh, S., and Durrant-Whyte, H. (2003). Decentralised Ground Target Tracking with Heterogeneous Sensing Nodes on Multiple UAVs. *Lecture Notes in Computer Science*, pages 545–565.

[Ridley et al., 2002] Ridley, M., Nettleton, E., Sukkarieh, S., and Durrant-Whyte, H. (2002). Tracking in decentralised air-ground sensing networks. In *Information Fusion, 2002. Proceedings of the Fifth International Conference on*, volume 1.

[Russell et al., 1995] Russell, S., Norvig, P., Canny, J., Malik, J., and Edwards, D. (1995). *Artificial intelligence: a modern approach*. Prentice Hall Englewood Cliffs, NJ.

[Ryan and Hedrick, 2005] Ryan, A. and Hedrick, J. (2005). A mode-switching path planner for UAV-assisted search and rescue. *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 1471–1476.

[Rysdyk, 2003] Rysdyk, R. (2003). Uav path following for constant line-of-sight. In *Proc. 2nd AIAA Unmanned Unlimited Conf.*

[Schumacher et al., 2002] Schumacher, C., Chandler, P., and Rasmussen, S. (2002). Task allocation for wide area search munitions. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3.

[Seraji and Howard, 2002] Seraji, H. and Howard, A. (2002). Behavior-based robot navigation on challenging terrain: A fuzzylogic approach. *Robotics and Automation, IEEE Transactions on*, 18(3):308–321.

[Shapley, 1953] Shapley, L. (1953). A value for n-person games. *The Shapley value-Essays in Honor of Lloyd S. Shapley*, pages 31–40.

[Shim et al., 2005] Shim, D., Chung, H., Kim, H., and Sastry, S. (2005). Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles. *AIAA Guidance, Navigation, and Control Conference and Exhibit,(San Francisco, CA), Aug*, pages 2005–6478.

[Shim et al., 2003] Shim, D., Kim, H., and Sastry, S. (2003). Decentralized reflective model predictive control of multiple flying robots in dynamic environment. *Proceedings of the IEEE Conference on Decision and Control*.

[Singh et al., 2001] Singh, L., Fuller, J., Center, U., and Hartford, E. (2001). Trajectory generation for a UAV in urban terrain, using nonlinearMPC. *American Control Conference, 2001. Proceedings of the 2001*, 3.

[Singh et al., 2000] Singh, S., Simmons, R., Smith, T., Stentz, A., Verma, V., Yahja, A., and Schwehr, K. (2000). Recent progress in local and global traversability for planetaryrovers. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, 2.

[Sinha et al., 2005] Sinha, A., Kirubarajan, T., and Bar-Shalom, Y. (2005). Autonomous Ground Target Tracking by Multiple Cooperative UAVs. In *Aerospace, 2005 IEEE Conference*, pages 1–9.

[Spires and Goldsmith,] Spires, S. and Goldsmith, S. Exhaustive Geographic Search with Mobile Robots Along Space-Filling Curves.

[Stolle and Rysdyk, 2003] Stolle, S. and Rysdyk, R. (2003). Flight path following guidance for unmanned air vehicles with pan-tilt camera for target observation. *Digital Avionics Systems Conference, 2003. DASC'03. The 22nd*, 2.

[Sujit and Ghose, 2004] Sujit, P. and Ghose, D. (2004). Search Using N-person Game Models.

[Sujit and Ghose, 2005] Sujit, P. and Ghose, D. (2005). Search by Two UAVs with Flight Time Constraints Using Game Theoretical Models.

[Sun et al., 2008] Sun, M., Zhu, R., and Yang, X. (2008). UAV Path Generation, Path Following and Gimbal Control. *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, pages 870–873.

[Sureka and Wurman, 2005] Sureka, A. and Wurman, P. (2005). Using tabu best-response search to find pure strategy nash equilibria in normal form games. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 1023–1029. ACM New York, NY, USA.

[Sutton and Bitmead, 2000] Sutton, G. and Bitmead, R. (2000). Computational implementation of NMPC to nonlinear submarine. *Nonlinear Model Predictive Control*, pages 461–471.

[Tan et al.,] Tan, C., Sutton, R., and Chudley, J. Quasi Random Manoeuvre Based Motion Planning Algorithm for Autonomous Underwater Vehicles. In *Proceedings in the 16th IFAC World Conference*.

[Tang and Ozguner, 2005] Tang, Z. and Ozguner, U. (2005). Motion planning for multitarget surveillance with mobile sensor agents. *Robotics, IEEE Transactions on [see also Robotics and Automation, IEEE Transactions on]*, 21(5):898–908.

[Teng et al., 1993] Teng, Y., DeMenthon, D., and Davis, L. (1993). Stealth terrain navigation. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(1):96–110.

[Tews et al., 2004] Tews, A., Mataric, M., and Sukhatme, G. (2004). Avoiding detection in a dynamic environment. *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, 4.

[Theodorakopoulos and Lacroix, 2008] Theodorakopoulos, P. and Lacroix, S. (2008). A strategy for tracking a ground target with a UAV. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice (France)*.

[Theodorakopoulos and Lacroix, 2009] Theodorakopoulos, P. and Lacroix, S. (2009). UAV target tracking using an adversarial iterative prediction. In *IEEE/RSJ International Conference on Robotics and Automation, Kobe (Japan)*.

[Thomasson, 1998] Thomasson, P. (1998). Guidance of a Roll-Only Camera for Ground Observation in Wind. *Journal of Guidance, Control and Dynamics*, 21:39–44.

[Tomlin et al., 2000] Tomlin, C., Lygeros, J., and Shankar Sastry, S. (2000). A game theoretic approach to controller design for hybrid systems. *Proceedings of the IEEE*, 88(7):949–970.

[Tversky, 2004] Tversky, A. (2004). *Preference, Belief, and Similarity: Selected Writings*. Bradford Books.

[Ulrich and Borenstein, 1998] Ulrich, I. and Borenstein, J. (1998). VFH+: reliable obstacle avoidance for fast mobile robots. *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, 2.

[U.S.Shukla and P.R.Mahapatra, 1990] U.S.Shukla and P.R.Mahapatra (1990). The proportional navigation dilemma-pure or true? *Aerospace and Electronic Systems, IEEE Transactions on*, 26(2):382–392.

[von Neumann and Morgenstern, 1953] von Neumann, J. and Morgenstern, O. (1953). Theory of Games and Economic Behavior. *New York*.

[Wong et al., 2005] Wong, E., Bourgault, F., and Furukawa, T. (2005). Multi-vehicle Bayesian Search for Multiple Lost Targets. In *Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3169–3174.

[Xi and Baras, 2007] Xi, W. and Baras, J. (2007). MPC based motion control of car-like vehicle swarms. *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pages 1–6.

[Yang and Yang, 1997] Yang, C.-D. and Yang, C.-C. (1997). Optimal pure proportional navigation for maneuvering targets. *Aerospace and Electronic Systems, IEEE Transactions on*, 33(3):949–957.

[Zengin and Dogan, 2007] Zengin, U. and Dogan, A. (2007). Real-Time Target Tracking for Autonomous UAVs in Adversarial Environments: A Gradient Search Algorithm. *IEEE Transactions on Robotics*, 23(2):294.