Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Modeling Air Combat with Influence Diagrams

av

Christopher Bergdahl

LIU-IDA/LITH-EX-A--13/031--SE

2013-06-07

Linköpings universitet

Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings universitet
581 83 Linköping

Examensarbete

Modeling Air Combat with Influence Diagrams

av

Christopher Bergdahl

LIU-IDA/LITH-EX-A--13/031--SE

2013-06-07

Handledare: Fredrik Heintz

Examinator: Andrzej Szalas

Abstract

Air combat is a complex situation, training for it and analysis of possible tactics are time consuming and expensive. In order to circumvent those problems, mathematical models of air combat can be used. This thesis presents air combat as a one-on-one influence diagram game where the influence diagram allows the dynamics of the aircraft, the preferences of the pilots and the uncertainty of decision making in a structural and transparent way to be taken into account. To obtain the players' game optimal control sequence with respect to their preferences, the influence diagram has to be solved. This is done by truncating the diagram with a moving horizon technique and determining and implementing the optimal controls for a dynamic game which only lasts a few time steps.

The result is a working air combat model, where a player estimates the probability that it resides in any of four possible states. The pilot's preferences are modeled by utility functions, one for each possible state. In each time step, the players are maximizing the cumulative sum of the utilities for each state which each possible action gives. These are weighted with the corresponding probabilities. The model is demonstrated and evaluated in a few interesting aspects. The presented model offers a way of analyzing air combat tactics and maneuvering as well as a way of making autonomous decisions in for example air combat simulators.

Acknowledgements

My first acknowledgement goes to Saab Aeronautics for providing me with the essential equipment and information for writing this thesis. I owe a lot to my supervisors Tina Erlandsson and Fredrik Heintz for their rapid and sharp feedback and guidance throughout the thesis process. The thoughts and feedback from my examiner Andrzej Szalas and my opponent Mikael Niemi are equally appreciated. I am also grateful for anyone else at Saab who has in some way aided me in my work, none mentioned none forgotten.

Of course my friends and family deserves big thanks for occupying my mind with other activities apart from writing this thesis, a help which not should be underestimated.

Linköping, June 2013

Christopher Bergdahl

Table of content

1. Introduction

Close range air combat, where the combatants are too close to use any long-range missiles and need to rely on automated canons, can be seen as a one-on-one game. In the game, the two participants are trying to place themselves in a good firing position, in order for the fired rounds to hit, while at the same time denying their opponent to do the same. By assuming that the opponent acts rationally, i.e. is trying to optimize its own motion, a player can in turn calculate how to act in order to win the battle, or at least not lose it. However, uncertainties about the opponent’s acting and planning makes the calculation problematic. Even if full information about the opponent’s position and possible actions were acquired, there is no way of telling exactly how the opponent will act. Another problem is to create artificial pilots that act in a controlled, explainable and understandable manner. Air combat today is indeed a complicated situation where the decisions of pilots concern maneuvering, using weapons systems as well as utilizing the onboard devices. The outcome of air combat depends on the decisions of the pilots as well as the aircraft performance and available weapons. Since analyses of air combat tactics and techniques as well as training of pilots are both time consuming and expensive, mathematical models of the kind described above is an interesting method of circumventing those problems.

The first purpose of this work is to create and implement a two player influence diagram game model of an air combat which works as briefly described above. Influence diagrams are directed acyclic graphs in which probabilistic inference and decision problems can be modeled and solved. The report introduces a multi-agent influence diagram game which describes the control decisions of pilots in a one-on-one combat where they both try to reach a good firing position as fast as possible. By using a multi-agent influence diagram game to model the air combat, the dynamics of the aircraft, the preferences of the pilots and the uncertainty of decision making are all taken into account in a structural and transparent way. In the game, a player is assumed to have won when it has reached a position where it is possible for the player to open fire on its opponent. Sometimes, if for example a player finds itself in a disadvantageous situation in the game, not loosing the game is considered a success. This since, in a real life situation, it would save both man and aircraft.

The second purpose is to analyze the model. This is done by a simulation of the model for a set of scenarios and an analysis of the result. It is important that the model is consistent in its results and that it provides reasonable results. Therefore is a basic evaluation conducted where the model’s sensitivity to changes in the players’ initial state is tested. The moving horizon technique plays an important part in the model, so how the number of look-ahead steps of the players affect the outcome is of interest to evaluate. How well the model predicts its future states is another way to evaluate the technique and both are conducted in this work. There exist different strategies for modeling the opponent’s controls, acting as if it is equally probable that the opponent chooses any of its possible actions is the method used in this work, but the last part of the evaluation compares different techniques against each other to see what the result is.

In order to obtain the players’ game optimal control sequence with respect to their preferences, the influence diagram has to be solved. There are several known solutions to ordinary influence diagrams, see [15] for one example. But for a game representation of an influence diagram, as in this case, the situation is different. Koller and Milch present in [4] a divide-and-conquer strategy which breaks the diagram into smaller subparts and solves them iteratively. But the game at hand cannot be divided since every optimal game control for a player depends on its future decision. Furthermore, the decisions may affect all future probability distributions and other variables in the influence diagram. Since the controls will depend on future decisions, feedback solutions are preferred. Therefore is dynamic programming [16] the solution which is used in this work. Dynamic programming has the drawback of combinatorial expansion of the computation. In order to deal with this is the influence diagram truncated and the computations of the actions are limited to a short time horizon which includes only the next few time steps. This approach has been applied to dynamic games before, see [17] for an example.

This work is heavily influenced by [1] where an air combat is modeled by a moving horizon influence diagram game. There are however a few modifications both in the model and the numerical examples; A different set of differential equations describing the players motion are used in this thesis, they are taken from a similar work ([6]) so they have been proven to work in this type of task before. The control variables are not identical either. This is in order to fit with the new set of differential equations describing the motion of the players. The physical constraints of the players are new in this work, this in order to get more realistic results from the model. Above all is there a ground constraint which does not allow the players to have an altitude below zero. Violating this constraint is considered to lose the game. But the main difference is how the evaluation of the model is conducted. Totally new aspects, which are described above, are investigated in this thesis.

The structure of the thesis is as follows. First, the reader is introduced to the two major techniques used in this work; influence diagrams and the receding horizon control. Then chapter 3 formulates the air combat influence diagram game, explains the approach for obtaining moving horizon feedback solutions, gives a numerical example of the model and briefly describes a simple form of visualization which is done in Matlab. In chapter 4, an evaluation of the model is done to see if the model is robust in terms of initial states, what parameters in terms of cognition and perception are most critical in the affect of the outcome and how much the uncertainties of future states affect the accuracy in prediction in the model. In the end, some concluding remarks and thoughts are presented in chapter 5.

2. Presentation of techniques to be used

In this chapter, the two major techniques that are used in this work are briefly described. The techniques and methods used to model the air-combat game in this thesis are the same as those being used in [1]. Influence diagrams and especially the extension multi-agent influence diagrams are used in the decision making of the players. A receding horizon control is used to truncate the decision horizon in order to make the calculation of the modeling computationally easier. There has been no reason to exchange any methods or techniques as they have already been proven to work well in this type of modeling.

2.1 Influence Diagrams

Influence diagrams were introduced by Howard and Matheson [2] as a tool to simplify the modeling and analysis of decision trees. The diagram can be represented by a Bayesian network [9] extended with decision nodes, often represented by squares, and with utility nodes, often represented by diamonds shapes.

Similar to Bayesian networks, the order of the decisions and the order of the set of observations between decisions are important. Edges that point into a decision node in an influence diagram are sometimes called information links, and they indicate that the state of the parent must be known prior to making the decision. In Figure 1, the state of C must be known before the decision in D can be made.

The utility nodes have no children and no states, instead they indicate the utility or the usefulness of the given network configuration. This is done by mapping each permutation of the states of its parents to one utility value. In Figure 1, if B and D have 2 different states each, the utility node U will have 2*2=4 different utility configurations. The goal is to maximize the expected value of the utility node. So the decision, or the link of decisions (depending on whether there is one or several decision nodes in the influence diagram) chosen in the influence diagram should be the one(s) that give the highest expected value in the corresponding utility-node. For further explanation and more information on how to evaluate influence diagrams and how to find the optimal policies see [3].

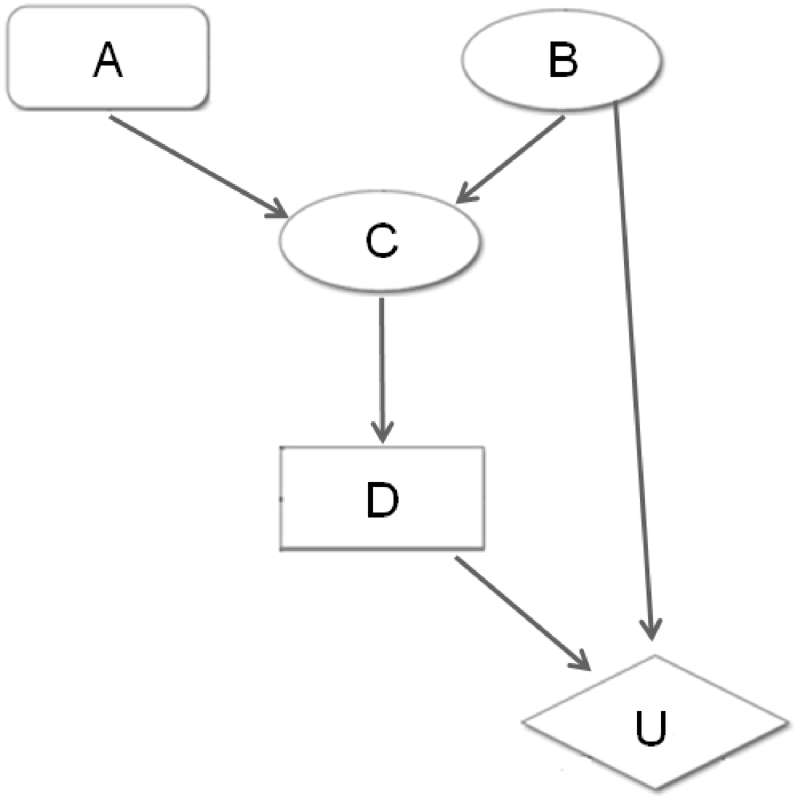

Figure 1: A simple example of an Influence Diagram where A is a deterministic node, B and C are uncertainty nodes, D is a decision node and U is a utility node.

One of the extensions to Influence diagrams are multi-agent influence diagrams, presented by Koller and Milch [4]. As the name implies, this extension allows for more than one agent to be a part of the diagram and decision for multiple agents can be taken into consideration. For this to be possible, every decision- and utility node must be associated with a particular agent. The multi-agent influence diagram must make explicit the dependencies between decision variables. That is, if a decision variable *x* relies on another decision variable *y* the agent making decision *x* must take the decision rule of *y* into consideration in order to optimize its own decision rule.

2.2 Receding horizon control

When the actions for obtaining near-optimal feedback controls are required, a technique called Receding horizon control (RHC) is used in this work. The technique is also known as moving horizon control and model predictive control [6]. The controls are optimized online by using a limited planning horizon and approximating the utilities of the controls to go. Compared to other methods which compute optimal feedback controls ([10] for example) RHC saves much computational work when truncating the planning horizon. A drawback is that those computational savings are at the expense of non-optimal controls being used.

The principle idea behind RHC is visualized in Figure 2. Based on the data received at time step t_k, the future evolvement of the system is predicted and all near-optimal feedback controls are obtained online for all time steps up to step t_k+T. Due to the lack of accuracy in the utilized model against the true system, only the first near-optimal feedback control in time step t_{k+1} is implemented. This provides a feedback mechanism that takes uncertainty regarding the differences from the utilized model to the true system into account. Of course even if the model would correspond perfectly to the actual system, a global optimal solution is not guaranteed because of the limited planning horizon. Even though continuous control values could be obtained for each time step, constant and discrete control values are often applied for numerical reasons, an issue that is taken into consideration further into the rapport.

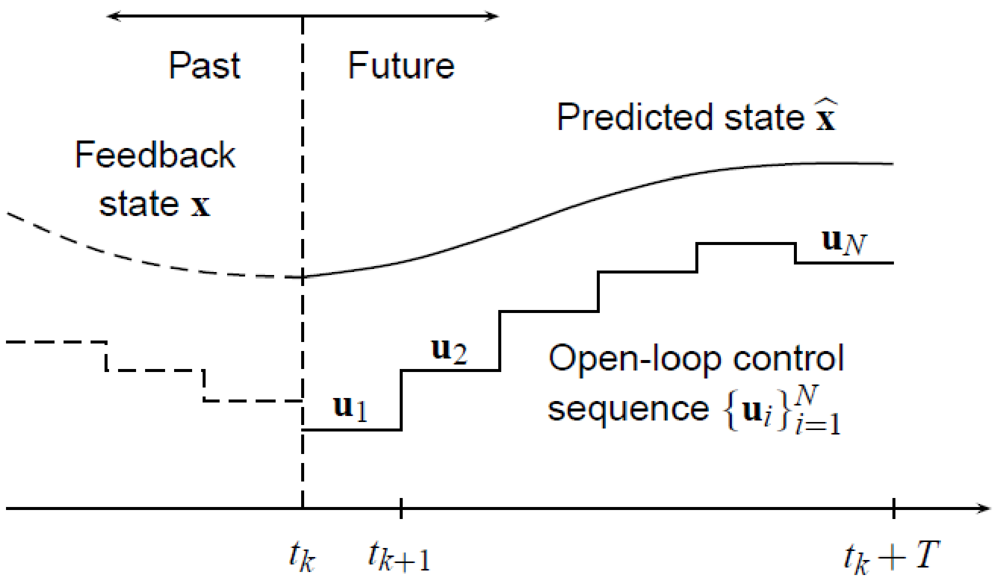

Figure 2: Principle idea of receding horizon control. The system is at stage t_k and calculates future controls $u_1 \ldots u_N$ for every time step until t_k+T where T is predetermined.

3. Modeling of the air combat game

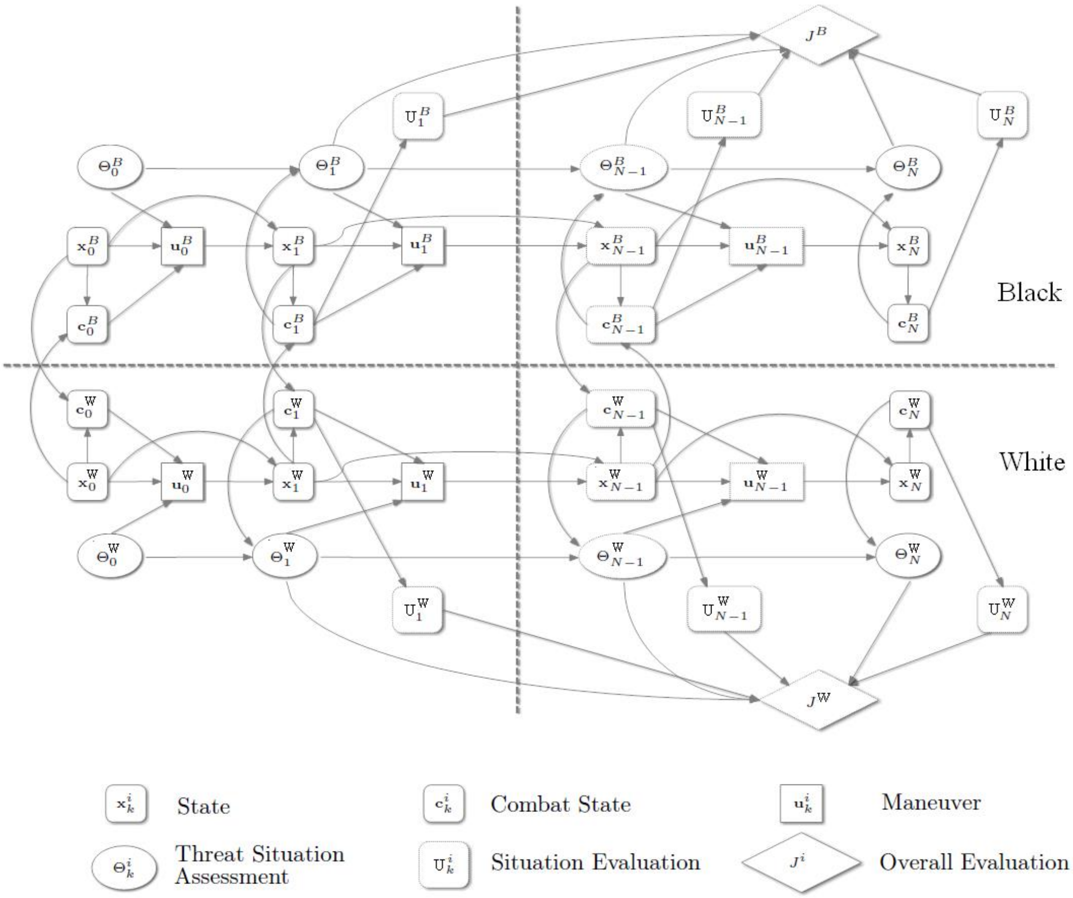

Figure 3 Influence diagram of the air combat game.

In this work, air combat is modeled as a game between two players, black and white player. The influence diagram representation of the air combat game between the two players is shown in Figure 3. The diagram is in most part taken from Fig. 1 in [1], the representation for calculating the overall evaluation is modified to make more sense with the calculations which follows and red/blue player is exchanged to black/white player. The upper and lower parts of the diagram represent the two players and their variables at discrete time steps respectively. The variables are the decision, chance, deterministic and value nodes depicted by squares, ovals, rounded squares and diamonds respectively. These variables represent the decisions to be made, uncertain probabilistic variables, deterministic inputs and payoffs to be optimized. The arcs into a decision node indicates the information that is available before the decision is made, arcs into a chance node means that that node is conditionally dependent on the information from the arc. Arcs directed into a deterministic or value node says that the value of that node is partially determined by the input from the arc.

State node

The first thing to know about the modeling of the air combat game is that the players move in their three dimensional space according to the following equations of motion, [6]

$$x' = v \cos\gamma \cos\chi, \tag{1}$$

$$y' = v \cos\gamma \sin\chi, \tag{2}$$

$$h' = v \sin\gamma, \tag{3}$$

$$\gamma' = \frac{g}{v}(\mathrm{n} \cos\mu - \cos\gamma), \tag{4}$$

$$\chi' = \frac{g}{v}\frac{\mathrm{n}\cdot\sin(\mu)}{\cos(\gamma)}, \tag{5}$$

$$v' = \frac{1}{m}(\eta\, T_{max} - D(M(v,h))) - g \sin\gamma =$$

$$\frac{1}{m}\left(\eta\, T_{max} - \frac{1}{2} C_D v^2 S \varsigma(h)\right) - g \sin\gamma, \tag{6}$$

where x and y are the are the horizontal coordinates, h is the height, γ is the flight path angle, χ is the heading angle and v is the velocity. There are some constant values in the equations, such as the gravity constant g and the mass of the aircraft m. The remaining variables are the load factor n, the bank angle μ, the throttle setting η, the maximum thrust available T_{max}, the mach number M and the drag force D. The variables defining the drag force D are the zero drag coefficient C_D, the reference wing area S, the air density ς (h) and the velocity v. The values of the zero drag coefficient C_D as well as the maximum thrust available T_{max} are taken from Fig. 3 in [11]. The air density is taken from the International Standard Atmosphere [12] and the reference wing area from [7].

These equations are not identical to the ones used in [1], instead they are taken from a similar work [6]. This diversion is made because the equations in [1] required tabular data which was not provided which, in turn, made it difficult to implement them correctly.

The state of a player in a certain time step can be described by the following state vector

$$\mathbf{x}_k = [\, x_k \;\; y_k \;\; h_k \;\; \gamma_k \;\; \chi_k \;\; v_k \,]^T \tag{7}$$

Maneuver node

Each player has control variables which they can affect. The variables are the throttle setting η, the bank angle μ and the load factor n. These variables form a control vector

$$\mathbf{u}_k = [\eta_k \;\; \mu_k \;\; n_k]^T \tag{8}$$

The throttle setting represents the throttle of an aircraft. The bank angle is the angle between the aircraft's normal (vertical) axis and the Earth's vertical plane containing the aircraft's longitudinal axis. The load factor is defined as the ratio of the lift of an aircraft to its weight. It is dimensionless but commonly expressed in *g* units.

Given this control vector, the position of each player is updated at every stage of the game by integrating equations (1)-(6) as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \int_{t_k}^{t_k+\Delta t} \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \approx \mathbf{x}_k + \Delta t * \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \tag{9}$$

where the function **f** consist of [x' y' h' γ' χ' v'] calculated by equations (1)-(6) respectively. The approximation from an integral to just a function is made by the Euler method (see [18] or [19]). So a new state of a player depends on its previous state and what maneuver it has made, as shown in Figure 3.

The modeling must give reasonable results, therefore the state- and control vector of a player are constrained by a set of constraints. Example values of such constraints are given in section 3.2 where a numerical example of the game is presented.

Combat state node

In order to describe the relationship between the two players, each plane is assigned a combat state vector $\mathbf{c}_k$ which resides in the combat state node in Figure 3. The combat state vector depends only on the current states of the player and its opponent, this is also shown in Figure 3. This combat state vector can be defined in many different ways and the definition depends on which variables that might be of interest to compare. See [1, 6, 8] for examples. The choice of combat vector for this work is made identical to the one used in [1], this since much of the other parts of this work is taken from that paper but also because it was an easy and relevant choice of vector. In this case the combat state vector for black player is defined as

$$\mathbf{c}_\mathbf{k}^\mathbf{B} = [\,\omega_k^B \;\; \theta_k^B \;\; d_k^B\,]^T, \tag{10}$$

where ω_k^B is the bearing angle and θ_k^B is the angle-off, i.e. the angles between the line of sight vector of black player and the velocity vector of black player and white player respectively. ω_k^i, $\theta_k^i \in [0, \pi]$. $d_k^i > 0$, i=B,W, is the distance between the players. The variables are shown in Figure 4 and are calculated according to the following equations

$$\omega_k^B = \arccos\left\{\frac{(x_k^W - x_k^B)\cos\gamma_k^B \cos\chi_k^B + (y_k^W - y_k^B)\cos\gamma_k^B \sin\chi_k^B + (h_k^W - h_k^B)\sin\gamma_k^B}{d_k^B}\right\} \tag{11}$$

$$\theta_k^B = \arccos\left\{\frac{(x_k^W - x_k^B)\cos\gamma_k^W \cos\chi_k^W + (y_k^W - y_k^B)\cos\gamma_k^W \sin\chi_k^W + (h_k^W - h_k^B)\sin\gamma_k^W}{d_k^B}\right\} \tag{12}$$

$$d_k^B = \sqrt{(x_k^W - x_k^B)^2 + (y_k^W - y_k^B)^2 + (h_k^W - h_k^B)^2} \tag{13}$$

The combat state vector for white player is calculated by swapping the indices from B to W and vice versa in equations (11)-(13).

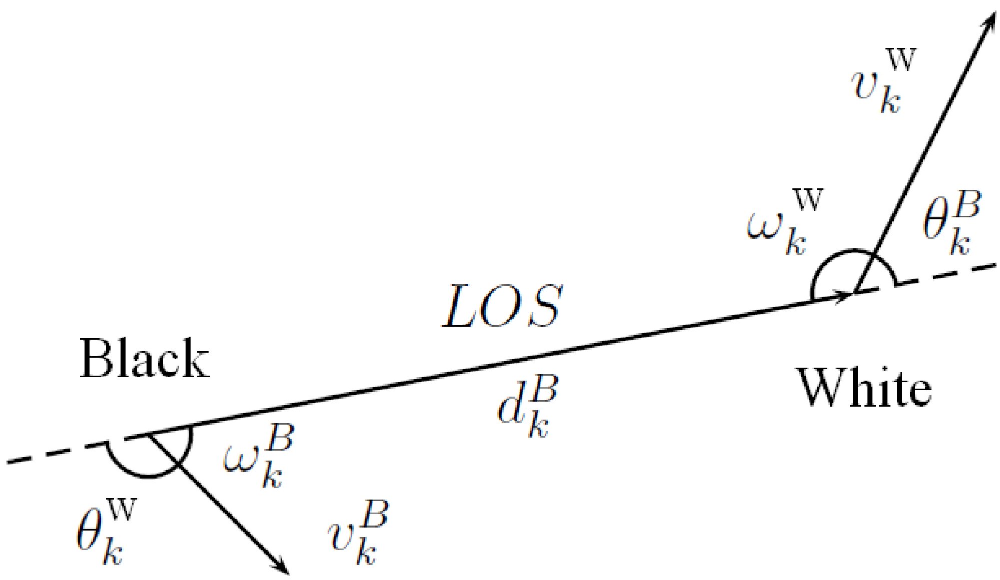

Figure 4 Combat state variables. The picture is taken from [1].

During the initialization of the game each player is assigned a target set and the goal of the game is for the players to drive their combat state vector into their own target set. The target set of a player i is defined as

$$\mathrm{T}^i = \{ \mathbf{c}_\mathbf{k}^\mathbf{i} \mid \mathrm{g}(\mathbf{c}_\mathbf{k}^\mathbf{i}) \leq 0 \}, \tag{14}$$

$$\mathbf{g}(\mathbf{c}_\mathbf{k}^\mathbf{i}) = [\, \omega_k^i - \omega_T^i \quad \theta_k^i - \theta_T^i \quad d_k^i - d_T^i \,]^{\mathrm{T}}, \tag{15}$$

where the target variables ω_T^i, θ_T^i and d_T^i are fixed and determined at the initialization of the game. The constraint in (14) holds per element. If either or both of the players succeeds in driving their combat state vector into its own target set the game terminates. The game also terminates if the number of stages in the game has reached the maximum number of stages in the game, N_{max}. This means that neither of the players has been able to drive its combat state vector into its own target set during N_{max} number of stages. There are some physical conditions that may result in a termination of the game, for instance if the distance between the players d_k^i is greater than 12 000m. This results in a draw. There are other conditions which, if violated, result in a loss of the game. These physical conditions are used to get a more realistic game and not strange results with for example negative velocity. The termination conditions give four possible outcomes of the game which are presented in Table 1.

Table 1: Possible outcomes of the air combat game.

Outcome	Condition
Black player wins	$\mathbf{c_k^B} \in T^B$ and $\mathbf{c_k^W} \notin T^W$ or $h_k^W \leq 0$ or $h_k^W > 40\,000$m or $v_k^W < 50$ m/s
White player wins	$\mathbf{c_k^W} \in T^W$ and $\mathbf{c_k^B} \notin T^B$ or $h_k^B \leq 0$ or $h_k^B > 40\,000$m or $v_k^B < 50$ m/s
Joint capture	$\mathbf{c_k^B} \in T^B$ and $\mathbf{c_k^W} \in T^W$
Draw	$\mathbf{c_k^B} \notin T^B$ and $\mathbf{c_k^W} \notin T^W$ and ($N=N_{max}$ or $d_k^i > 12\,000$m) or ($h_k^W \leq 0$ or $h_k^W > 40\,000$m or $v_k^W < 50$ m/s) and ($h_k^B \leq 0$ or $h_k^B > 40\,000$m or $v_k^B < 50$ m/s)

Up to here has everything in the construction of the model been deterministic.

Threat situation assessment node

At each stage of the game both players assesses the threat of the situation they are in with help from their combat state vector and the assessment of the threat in the previous time stage as seen in Figure 3. The assessment gives probability values which represent the probability that the player is in one of four different states. These states describes in what type of situation a player is in relation to its opponent. The assessment is modeled by a discrete random variable Θ_k^i given in the threat situation assessment node (Figure 3) and the different states for black player are listed and described in Table 2. The states of white player are obtained by switching the description of the second and third row.

Table 2: States given the threat situation assessment of black player.

State	Θ_k^B	Description
Neutral	1	Either the players are a big distance apart or the players are headed away from each other.
Advantage	2	Black player is pursuing white player at a short distance.
Disadvantage	3	White player is pursuing black player at a short distance.
Mutual disadvantage	4	Both players are headed towards each other at a short distance.

The probability that a player is in a given state given the combat state vector in the current stage is computed by P(Θ_k^i = j | **C** = $\mathbf{c_k^i}$), j=1..4, where the elements of **C** are variables for the corresponding combat state variables. The probabilities sum up to a unity, i.e. $\sum_{n=1}^{4}$ P(Θ_k^i = n | **C** = $\mathbf{c_k^i}$) = 1. As shown in Figure 3, these probabilities work as prior beliefs to the succeeding stage's threat assessment probabilities, so the succeeding probability P(Θ_{k+1}^i = j) are equal to the posterior probabilities

P(Θ_k^i = j | **C** = $\mathbf{c_k^i}$).

The elements of the vector **C** are assumed to be independent, so the probability density function of the combat state given the threat assessment situation can be written as

$$\mathrm{P}(\mathbf{c_k^i} \mid \Theta_k^i = \mathrm{j}) = \mathrm{p}^{\omega,\mathrm{i}}(\omega_k^i \mid \Theta_k^i = \mathrm{j})\mathrm{p}^{\theta,\mathrm{i}}(\theta_k^i \mid \Theta_k^i = \mathrm{j})\mathrm{p}^{\mathrm{d,i}}(d_k^i \mid \Theta_k^i = \mathrm{j}) \tag{16}$$

The likelihood functions $\mathrm{p}^{\omega,\mathrm{i}}(\omega_k^i \mid \Theta_k^i = \mathrm{j})$, $\mathrm{p}^{\theta,\mathrm{i}}(\theta_k^i \mid \Theta_k^i = \mathrm{j})$ and $\mathrm{p}^{\mathrm{d,i}}(d_k^i \mid \Theta_k^i = \mathrm{j})$ should represent the distribution of the combat state variables given the players threat assessment outcome j. An example of such functions is given in section 3.2.

The probabilities of the next stage are calculated by using Bayes' formula [14] as

$$\mathrm{P}(\Theta_{k+1}^i = \mathrm{j} \mid \mathbf{C} = \mathbf{c}_{\mathrm{k+1}}{}^{\mathrm{i}}) = \frac{\mathrm{P}\left(\Theta_{\mathrm{k+1}}^{\mathrm{i}} = j\right)\mathrm{P}\left(\mathbf{c_{k+1}^i} \mid \Theta_{k+1}^i = j\right)}{\sum_{n=1}^{4}\mathrm{P}\left(\Theta_{\mathrm{k+1}}^{\mathrm{i}} = n\right)\mathrm{P}\left(\mathbf{c_{k+1}^i} \mid \Theta_{k+1}^i = n\right)} =$$

$$\frac{\mathrm{P}\left(\Theta_{\mathrm{k}}^{\mathrm{i}} = j \mid C = \mathbf{c_k^i}\right)\mathrm{P}\left(\mathbf{c_{k+1}^i} \mid \Theta_{k+1}^i = j\right)}{\sum_{n=1}^{4}\mathrm{P}\left(\Theta_{\mathrm{k}}^{\mathrm{i}} = n \mid C = \mathbf{c_k^i}\right)\mathrm{P}\left(\mathbf{c_{k+1}^i} \mid \Theta_{k+1}^i = n\right)} \tag{17}$$

The probabilities at stage k can be written in vector form as

$$\mathbf{p_k^i}(\mathbf{c_k^i}) = [\mathrm{P}(\Theta_k^i{=}1 \mid \mathbf{C}{=}\mathbf{c_k^i})\ \mathrm{P}(\Theta_k^i{=}2 \mid \mathbf{C}{=}\mathbf{c_k^i})\ \mathrm{P}(\Theta_k^i{=}3 \mid \mathbf{C}{=}\mathbf{c_k^i})\ \mathrm{P}(\Theta_k^i{=}4 \mid \mathbf{C}{=}\mathbf{c_k^i})]^{\mathrm{T}}, \tag{18}$$

where the P's are calculated according to equation (17).

Situation evaluation node

In each stage, an action is evaluated by a utility function. In the diagram shown in Figure 3 this represents the situation evaluation node. The utility values are calculated according to which threat assessment situation the player is assumed to be in, so in vector form the situation evaluation node can be written as

$$\mathbf{U}_{\mathbf{k}}^{\mathbf{i}}(\mathbf{c}_{\mathbf{k}}^{\mathbf{i}}) = [\mathrm{U}^{\mathrm{i}}(1,\mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \ \mathrm{U}^{\mathrm{i}}(2,\mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \ \mathrm{U}^{\mathrm{i}}(3,\mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \ \mathrm{U}^{\mathrm{i}}(4,\mathbf{c}_{\mathbf{k}}^{\mathbf{i}})]^{\mathrm{T}}, \tag{19}$$

where each element is calculated as

$$\mathrm{U}^{\mathrm{i}}(\mathrm{j},\mathbf{c}_{\mathbf{k}}^{\mathbf{i}}) = w_j^{\omega,i}\, u_j^{\omega,i}(\omega_k^i) + w_j^{\theta,i}\, u_j^{\theta,i}(\theta_k^i) + w_j^{d,i}\, u_j^{d,i}(d_k^i),\ \mathrm{j}=1..4 \tag{20}$$

Each of the single attribute functions $u_j^{\omega,i}$, $u_j^{\theta,i}$ and $u_j^{d,i}$ maps the value of a combat state variable to a utility scale where the best possible value corresponds to the utility value 1 and the worst possible combat state value corresponds to utility value 0. Each utility function is multiplied by a given positive weight value. The weights for a given threat assessment outcome j sum up to a unity. Examples of these single attribute utility functions and corresponding weights are also given in section 3.2.

Overall evaluation node

The solution to the game is the sequence of control vectors which provides the highest possible cumulative utilities, called payoff, for the players contained in their respective overall evaluation node in the diagram in Figure 3. They are calculated as follows,

$$\mathbf{J}^{\mathrm{i}}(\mathbf{u}_0^B,\ldots,\mathbf{u}_{N-1}^B,\mathbf{u}_0^W,\ldots,\mathbf{u}_{N-1}^W) = \sum_{K=1}^{N} \left[\mathbf{p}_{\mathbf{k}}^{\mathbf{i}}\left(\mathbf{c}_{\mathbf{k}}^{\mathbf{i}}\right)\right]^T \mathbf{U}_{\mathbf{k}}^{\mathbf{i}}\left(\mathbf{c}_{\mathbf{k}}^{\mathbf{i}}\right),\ \mathrm{i=B,W} \tag{21}$$

3.1 Moving horizon control

The length of the influence diagram in Figure 3 is, depending on the outcome, very large. In either case it is safe to say that it is too large to be computed all at once. For this reason, an approximate method must be considered. In this work it is the moving horizon control, where the horizon of the original influence diagram game in Figure 3 is truncated and optimal control sequences are computed for only a few stages ahead from the current stage, which is used. This is called a k-step look-ahead strategy. Thereafter only the first component of the optimal controls is implemented and the process is repeated until the game has finished.

To make the computation even easier the control variables are now discretized. At every stage a player can change its controls by a predefined rate of change. At stage k the possible control values are

$$\mathbf{S}_{\mathbf{k}}^{\mathbf{i}} = \mathbf{u}_{\mathbf{k}+1}^{\mathbf{i}} \pm \Delta \mathbf{u}^{\mathbf{i}}, \tag{22}$$

where $\Delta\mathbf{u}^{i}$ denotes the steps within the maximum rate of change for each control variable, see section 3.2 for example values of these.

When solving a one-step look-ahead strategy the players are maximizing the payoff

$$\mathbf{J}_{\mathbf{k},\mathbf{k}+1}^{\mathbf{i}}\left(\mathbf{u}_{\mathbf{k}}^{\mathbf{B}}, \mathbf{u}_{\mathbf{k}}^{\mathbf{W}}\right) = \left[\mathbf{p}_{\mathbf{k}+1}^{\mathbf{i}}\left(\mathbf{c}_{\mathbf{k}+1}^{\mathbf{i}}\right)\right]^{T} \mathbf{U}_{\mathbf{k}+1}^{\mathbf{i}}\left(\mathbf{c}_{\mathbf{k}+1}^{\mathbf{i}}\right), \qquad \text{i = B, W} \tag{23}$$

The states of the players, x_k^i, and the probabilities $\mathbf{p}_{\mathbf{k}}^{\mathbf{i}}$ are known at the current stage k. The states, combat state vectors and probabilities for the next stage k+1 are calculated by (9), (11-13) and (18). To make sure that the players stay away from states which violates the constraints a penalty value is added to the utility value(s) corresponding to the infeasible state(s).

Now the players want to maximize their own controls in relation to the possible states in which the opponent could reside in. At stage k, black player’s optimal control is given by

$$\max\left(\sum_{\forall \mathbf{u}_{\mathbf{k}}^{\mathbf{W}} \in \mathbf{S}_{\mathbf{k}}^{\mathbf{W}}} \mathbf{J}_{\mathbf{k},\mathbf{k}+1}^{\mathbf{B}}\left(\mathbf{u}_{\mathbf{k}}^{\mathbf{B}}, \mathbf{u}_{\mathbf{k}}^{\mathbf{W}}\right)\right), \forall \mathbf{u}_{\mathbf{k}}^{\mathbf{B}} \in \mathbf{S}_{\mathbf{k}}^{\mathbf{B}} \tag{24}$$

When solving a two-step look-ahead strategy, the players maximizes the following payoffs

$$\mathbf{J}_{\mathbf{k},\mathbf{k}+2}^{\mathbf{i}}\left(\mathbf{u}_{\mathbf{k}}^{\mathbf{B}}, \mathbf{u}_{\mathbf{k}+1}^{\mathbf{B}}, \mathbf{u}_{\mathbf{k}}^{\mathbf{W}}, \mathbf{u}_{\mathbf{k}+1}^{\mathbf{W}}\right) = \sum\nolimits_{n=k+1}^{k+2} \left[\mathbf{p}_{\mathbf{n}}^{\mathbf{i}}\right]^{T} \mathbf{U}_{\mathbf{n}}^{\mathbf{i}}\left(\mathbf{c}_{\mathbf{n}}^{\mathbf{i}}\right), \qquad \text{i = B,W} \tag{25}$$

Once again from black player’s point of view, since a control vector from one step ahead is needed to compute the control vector two steps ahead, an assumption is made that the opponent acts in an optimized manner. So first the payoffs for the opponent’s possible actions in step k are calculated. Then the action of the opponent with the highest payoff value, $\mathbf{u}_{k}^{W*}$, is used against all possible actions for black player in step k when its optimal control vector in step k+1 is computed, this is done as follows,

$$\max\left(\sum_{\forall \mathbf{u}_{\mathbf{k}+1}^{\mathbf{W}} \in \mathbf{S}_{\mathbf{k}+1}^{\mathbf{W}}} \mathbf{J}_{\mathbf{k},\mathbf{k}+2}^{\mathbf{B}}\left(\mathbf{u}_{\mathbf{k}}^{\mathbf{B}}, \mathbf{u}_{\mathbf{k}+1}^{\mathbf{B}}, \mathbf{u}_{\mathbf{k}}^{\mathbf{W}}, \mathbf{u}_{\mathbf{k}+1}^{\mathbf{W}}\right)\right), \forall \mathbf{u}_{\mathbf{k}+1}^{\mathbf{B}} \in \mathbf{S}_{\mathbf{k}+1}^{\mathbf{B}} \tag{26}$$

This way the optimal control vector for black player in step k+1 is found for every possible action in step k. The utility for black player's computed optimal control in step k+1 is then added to the corresponding utility value of the control vector in step k. The optimal control is then given by

$$\max\left(\sum_{\forall \mathbf{u}_{\mathbf{k}}^{\mathbf{W}} \in \mathbf{S}_{\mathbf{k}}^{\mathbf{W}}} \mathbf{J}_{\mathbf{k},\mathbf{k}+2}^{\mathbf{B}}\left(\mathbf{u}_{\mathbf{k}}^{\mathbf{B}}, \mathbf{u}_{\mathbf{k}+1}^{\mathbf{B}*}, \mathbf{u}_{\mathbf{k}}^{\mathbf{W}}, \mathbf{u}_{\mathbf{k}+1}^{\mathbf{W}*}\right)\right), \forall \mathbf{u}_{\mathbf{k}}^{\mathbf{B}} \in \mathbf{S}_{\mathbf{k}}^{\mathbf{B}} \tag{27}$$

When computing the optimal controls for the white player, the indices are switched from B to W.

The solutions for the two-step look-ahead strategy are then the control vectors ($\mathbf{u}_{\mathbf{k}}^{\mathbf{i}*}, \mathbf{u}_{\mathbf{k}+1}^{\mathbf{i}*}$), i = B,W. However, only the first control vector $\mathbf{u}_{\mathbf{k}}^{\mathbf{i}*}$ is implemented.

When computing optimal controls for any arbitrary K-step look-ahead strategy, the optimal control vector for the opponent in the current step is always used against all possible control vectors of the player in the current step when approximating the next step. The optimal control vector and its payoff value are always sent back to the previous step in order to add that payoff value to the corresponding control vector in the previous step.

For a numerical example and pseudo code of the algorithm, see the Appendix.

If no applicable control vector can be found, i.e. if all possible control vectors leads to states which violates the constraints, then a hard-coded function kicks in. This function returns one control vector which values depends on what type of constraint the state violates. The players do no planning ahead if in such a state. A simple explanation to the function is that when a player needs higher altitude, lower velocity or needs to rise it increases the load factor. Similarly when a player needs to lower its altitude, to increase its velocity or to stop its ascent, it decreases the load factor. Depending on what value the load factor has the bank angle is used to maximize the effect of the control variables.

The Moving horizon control technique for a K-step look-ahead strategy can be summarized in the following way:

1. Set k=0 and set the initial states of the system $\mathbf{x}_{\mathbf{0}}^{\mathbf{i}}, \mathbf{u}_0^{\mathbf{i}}$ and $\mathbf{p}_0^{\mathbf{i}}$, i = B,W.

2. Solve the optimal control sequence ($\mathbf{u}_{\mathbf{k}}^{\mathbf{i}*}$, $\mathbf{u}_{\mathbf{k}+1}^{\mathbf{i}*}$,…, $\mathbf{u}_{\mathbf{k}+\mathbf{K}-1}^{\mathbf{i}*}$) for both players using the computations described above.
3. Set $\mathbf{u}_{\mathbf{k}}^{\mathbf{B}} = \mathbf{u}_{\mathbf{k}}^{\mathbf{B}*}$ and $\mathbf{u}_{\mathbf{k}}^{\mathbf{W}} = \mathbf{u}_{\mathbf{k}}^{\mathbf{W}*}$ and update the state vector $\mathbf{x}_{\mathbf{k}+1}^{\mathbf{i}}$, the combat state vector $\mathbf{c}_{\mathbf{k}+1}^{\mathbf{i}}$ and the probabilities $\mathbf{p}_{\mathbf{k}+1}^{\mathbf{i}}$ according to equations (9), (11-13) and (18) respectively using the control vectors.
4. If either player has reached its target set (14) with its own combat state vector or if $k=N_{max}$ or any other termination condition seen in Table 1 has been fulfilled the game terminates. Otherwise set k=k+1 and go to step 2.

3.2 Numerical example

In this section, a concrete numerical example of the model is presented. Some attributes will change during the different scenarios in the evaluation but the overall numerical example is described here. The modeling was done using Matlab. All angle variables are in radians.

The control vector for both players are initially set as $\mathbf{u}_0^{\mathbf{i}}$ = [0.5 0 1]. This is the control vector for flying straight ahead, i.e. if the players where to implement this control vector for every stage of the game there would be two aircrafts just flying straight ahead. The variables of the control vector move within the intervals [0, 1], [$-\frac{\pi}{2}$, $\frac{\pi}{2}$] and [-4, 9] respectively. Since the work has been done in close contact with Saab, JAS Gripen is the most interesting aircraft to simulate in the modeling and the load factor values are therefore taken from [7]. The throttle setting η^i can move through its entire interval from one time step to another, varying between the values 0, 0.5 and 1. The bank angle μ^i has a maximum rate of change $\frac{\pi}{2}$, i.e. an aircraft is only allowed to bank with an angle of $\pm\frac{\pi}{2}$ at either direction between two time steps. The load factor n^i has a maximum rate of change of ±1 between two time steps.

The initial threat probability vector $\mathbf{p}_0^{\mathbf{i}}$ is set the same way as in all examples in [1], namely

$$\mathbf{p}_0^{\mathbf{i}} = [\ 0.25\ 0.25\ 0.25\ 0.25\ ]^T \tag{28}$$

The initial states, $\mathbf{x}_{\mathbf{0}}^{\mathbf{i}}$ i=B,W, of the players are given for each evaluation of the model in section 4.

The target set variables in (15) are, for both players, set to $\omega_T^i = \frac{\pi}{6}$, $\theta_T^i = \pi$ and d_T^i = 1000. The maximum number of stages in the game, N_{max} is set to 300.

The constraints of the states variables within the state vector defined by (7) are as follows:

$$150 \leq v^i \leq 640, \tag{29}$$

$$1000 \leq h^i \leq 30000 \tag{30}$$

$$-\frac{\pi}{3} \leq \gamma^i \leq \frac{\pi}{3} \tag{31}$$

$$h^i < 1200 \Rightarrow \gamma^i > -\frac{\pi}{18} \tag{32}$$

$$h^i < 2000 \Rightarrow \gamma^i > -\frac{\pi}{6} \tag{33}$$

$$h^i > 25000 \Rightarrow \gamma^i < \frac{\pi}{6} \tag{34}$$

Constraints (29), (30) concerning the velocity and altitude of the aircraft speak for themselves. Constraint (31) exists because if the plane is allowed to fly upside-down, it might find itself in an infeasible state and unable to ever recover from it. This will eventually inflict a violation of the altitude constraint. Therefore is constraint (31) necessary. This is also the reason why the bank angle only is allowed within the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The constraints (32) and (33) make sure that the aircraft is not diving too much at a too low altitude and constraint (34) make sure that the aircraft does not rises with too high altitude.

The likelihood functions in (16) are taken from Table 3 in [1] and are shown in Table 3. The variable a^i defines the steepness of the functions and is here set to 0.08 for both players. The value of a^i is lower from 0.1 as is used in [1], this is because in some cases 0.1 did not give the players a real chance to change opinion about which state it resided in. This way the players are more flexible during a test run. The variable D defines the maximum allowed distance between the aircrafts and is set to 12 000 m. This is higher than the 10 000m which is used in [1]. The choice to have a higher value on D is made because in some cases, having a higher value gave more interesting result since the game might last longer.

For the advantage outcome, where one player is chasing the other, it is highly probable that the bearing angle ω, the angle-off θ are and the distance d between the players all are small. Therefore monotonously decreasing functions, where lower values give high probability is a good choice for function. The reasoning behind the choices of functions for all other outcomes are done in a similar manner and are explained further in section 5.A in [1].

Table 3: Likelihood functions for different threat situation outcomes

j	Likelihood function	Range
1,3	$p^{\omega,i}(\omega^i \mid \Theta^i = j)=(a^i\omega/\pi+1-a^i/2)/\pi$	$\omega \in [0, \pi]$
2,4	$p^{\omega,i}(\omega^i \mid \Theta^i = j)=(-a^i\omega/\pi+1+a^i/2)/\pi$	$\omega \in [0, \pi]$
3,4	$p^{\theta,i}(\theta^i \mid \Theta^i = j)=(a^i\theta/\pi+1-a^i/2)/\pi$	$\theta \in [0, \pi]$
1,2	$p^{\theta,i}(\theta^i \mid \Theta^i = j)=(-a^i\theta/\pi+1+a^i/2)/\pi$	$\theta \in [0, \pi]$
1	$p^{d,i}(d^i \mid \Theta^i = j)=1/D$	$d \in [0, D]$
2,3,4	$p^{d,i}(d^i \mid \Theta^i = j)=(-a^id/D+1+a^i/2)/D$	$d \in [0, D]$

Monotonously decreasing or increasing likelihood functions might not always be the best way to go though. Arguments against it could be that if one variable should be more weighted in the sense of which state a player is in, i.e. the bearing angle should for example be more weighted if it tells a player more about which state it is in than what the angle-off or the distance does.

The utility functions and their corresponding weights from equation (20) are taken from Table 4 in [1] and they are shown in Table 4.

In a disadvantage situation, a player should try to turn away from the opponent's velocity vector, i.e. decrease the angle-off. The player should also try to increase the distance to the opponent, therefore small values on the angle-off variable and a large distance should result in a big utility value. If a player is in the disadvantage situation, its only focus is to escape from its adversary with no regard to where that escape-path leads, hence the weight for the bearing angle is zero.

The reasoning behind the choices of functions for all other outcomes are done in a similar manner and are explained further in section 5.B in [1].

Table 4: Utility functions and corresponding weights

Outcome	**j**	$w_j^{\omega,i}$	$w_j^{\theta,i}$	$w_j^{d,i}$	$u_j^{\omega,i}(\omega)$	$u_j^{\theta,i}(\theta)$	$u_j^{d,i}(d)$
Neutral	1	0.2	0.1	0.7	$(\pi-\omega)/\pi$	$(\pi-\theta)/\pi$	$(D-d)/D$
Advantage	2	0.3	0.0	0.7	$(\pi-\omega)/\pi$	$(\pi-\theta)/\pi$	$(D-d)/D$
Disadvantage	3	0.0	0.7	0.3	ω/π	θ/π	d/D
Mutual disadvantage[a]	4	0.2	0.1	0.7	$(\pi-\omega)/\pi$	$(\pi-\theta)/\pi$	$(D-d)/D$
Mutual disadvantage[b]	4	0.2	0.1	0.7	ω/π	θ/π	d/D

[a]Offensive tactic, the player prefers joint capture to draw. [b]Defensive tactic, the player prefers draw to joint capture

3.3 Visualization

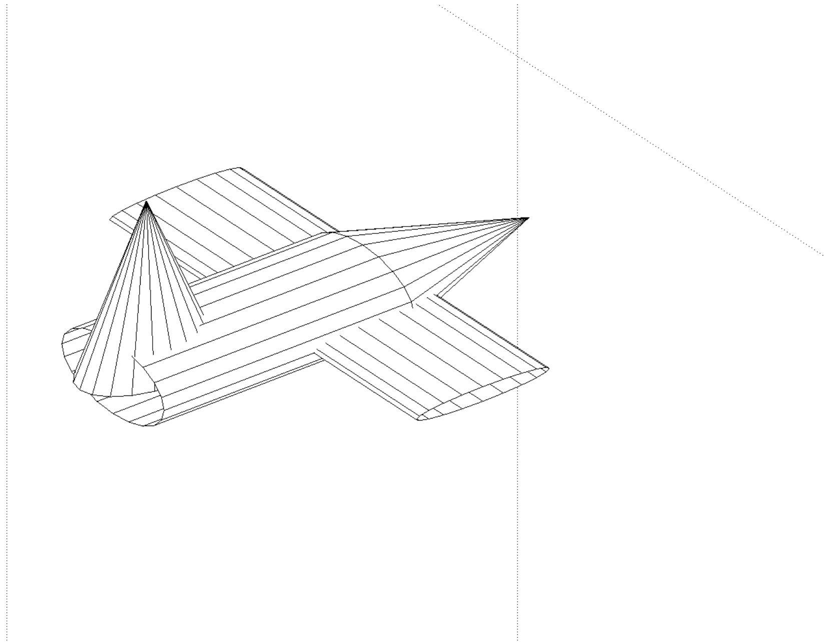

Figure 5: Plane used for the visualization.

In order to better understand the trajectories of the aircrafts obtained the model, a simple form of visualization was implemented using Matlab. For plane, cylinders were used for the body and the wings and cones were used for a nose and the “tail”. Such a plane is shown in Figure 5. The different parts of the plane were assembled using an hgtransform-object [5] and are thereby able to move and rotate as one unit. The only demand on the graphical part was that the rotation and movement of the aircrafts were displayed and that the trajectories of the planes could be followed. The hgtransform-object was used since it fulfilled all the requirements on the graphical part and it also proved to be easy to manipulate and rotate the aircraft using this technique.

An alternative solution to the visualization was to use Matlab’s own Simulink where there exists a demo that draws the trajectory of an aircraft. This solution would have been a better-looking one, graphically. But the work with setting the camera and making the aircraft rotate with different angles around different axes was estimated to take too much time and be too big of an excursion from the actual work since no weight was being laid on the graphical part.

4. Evaluation of the model

In this chapter, the model is evaluated in a number of scenarios and the results is interpreted and analyzed. For each of the different evaluations, three different scenarios are used: Chase, parallel and head-to-head. These scenarios are made up by the author as interesting cases to use.

In the chase scenario one player (black player in the following tests) starts directly behind its opponent (white player) giving it an advantage from the start, this means that the other player starts directly in front of its opponent and turned away from it. This is shown in Figure 6.

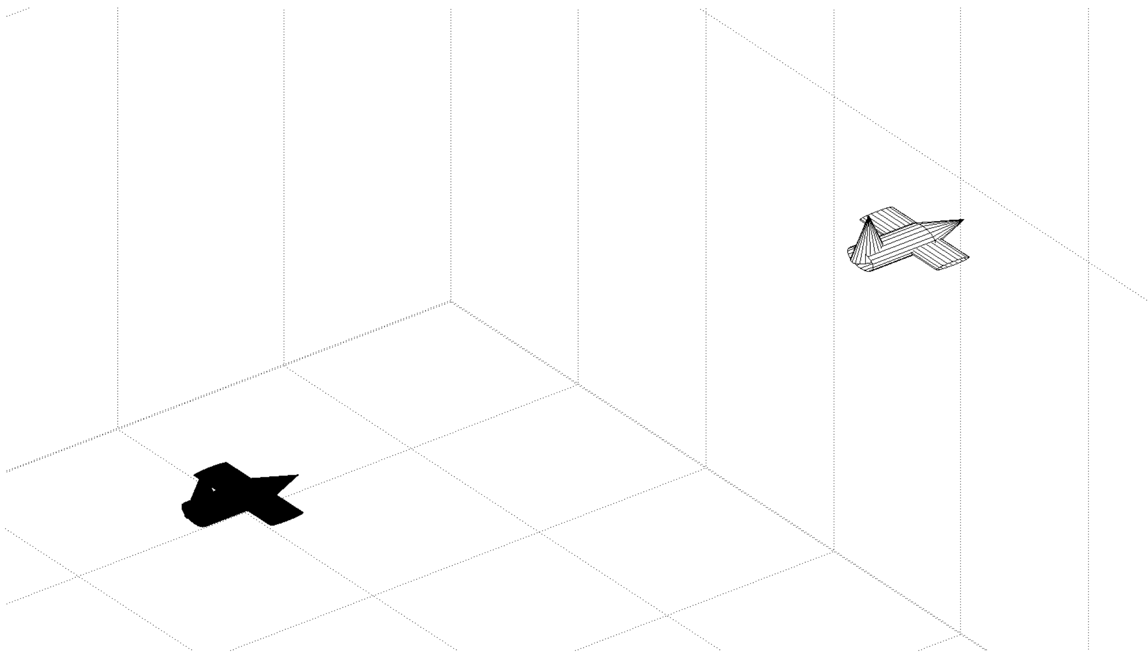

Figure 6 Initial states of the players in the chase scenario.

The parallel scenario states that the players start at a certain distance from each other on the x-axis but at the same coordinate on the y-axis. They start with equal altitude and with a velocity vector parallel to the y-axis. This means that, initially, none of the players has an advantage over the other. An example is shown in Figure 7.

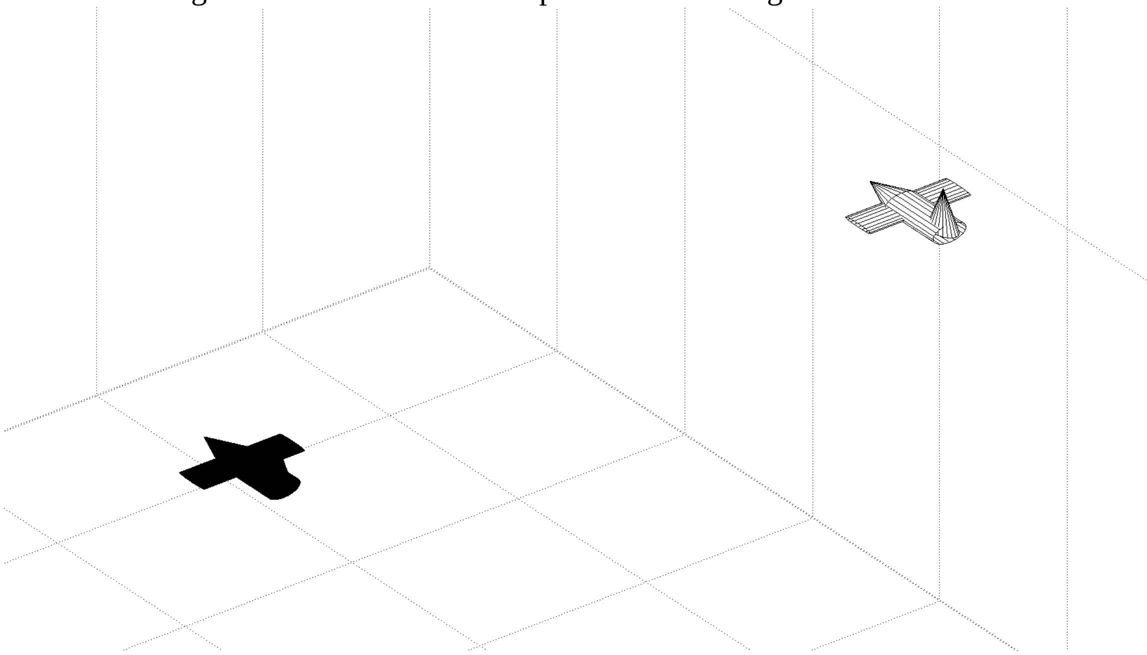

Figure 7 Initial states of the players in the parallel scenario.

In the head-to-head scenario the players start at a certain distance from each other on the x-axis but at the same coordinate on the y-axis, with equal altitude and both players are pointed straight to its opponent, as shown in Figure 8. In this scenario, as well as in the previous one, initially none of the players has an advantage over the other.

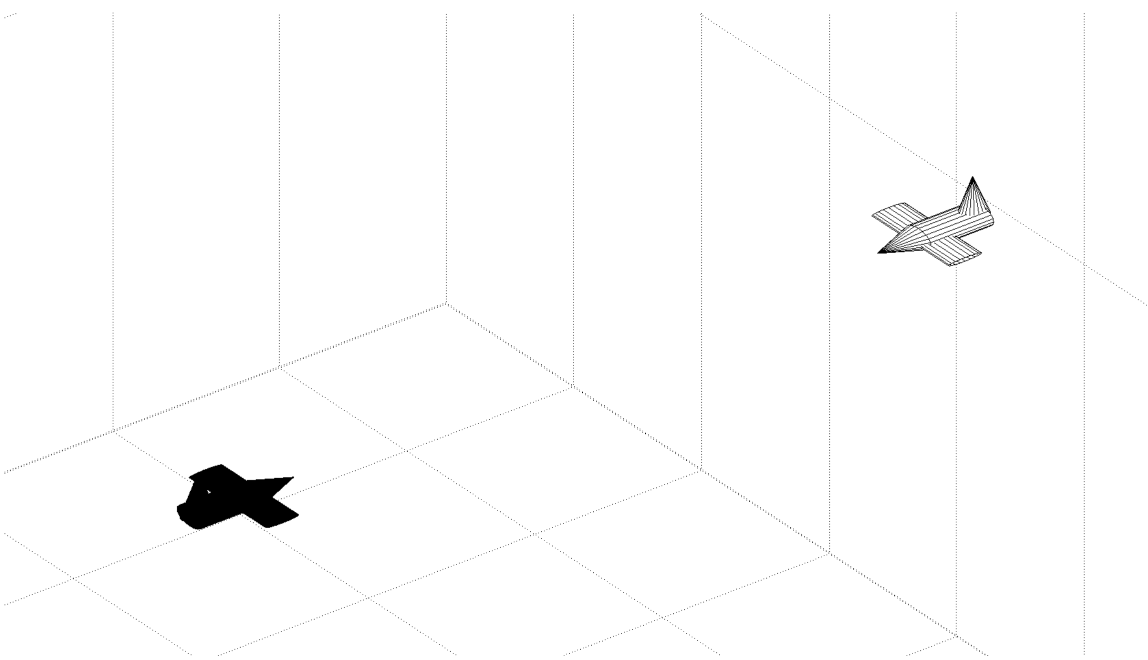

Figure 8 Initial states of the players in the head-to-head scenario.

In this chapter, a player that catches its opponent means that that player's combat vector (10) has reached its target set (14).

One thing to remember when examining the visualized trajectories of the players is that the lines are in 3D and might in some cases not seem realistic in 2D prints.

4.1 Varying the initial states of the players

The initial states of the players can be chosen freely. Therefore an evaluation of how sensitive the model is to changes in a players' initial state is performed. One aspect is to see if the result is symmetric if the initial states of the players are symmetrically shifted. When an initialization is said to be symmetric in this chapter, it means that if for example the velocity is varied and black player has speed x and white player speed y in the first case, then black player have speed y and white player have speed x in the second case.

As stated above the test runs take place in three different scenarios, chase, parallel and head-to-head. Within these scenarios the relative altitude, velocity and distance of the players are varied. The altitude of the players varies between the values 3000m, 6000m and 9000m, the velocity between 240, 300 and 360 m/s and the distance between 2000m, 5000m and 8000m. The distance is represented by the x-coordinate. The values of the variables are selected by the author as interesting values to use, they are chosen so to give the players room to act, i.e. not too high and not too low in regard to the constraints.

A 3-step look-ahead strategy is used for both players in all scenarios.

4.1.1 Chase

In Figure 6 the initial states of the players in a chase scenario is shown. From here, first the relative altitude is varied, followed by variation of relative speed and distance respectively.

The default initial states of the players are shown in Table 5. The altitude variable h, the velocity variable v and the distance variable x all varies between the values given in the introduction of this chapter. This gives a total of 21 test runs for this scenario.

Table 5: Initial states of the players in the chase scenario.

	x_0, m	y_0, m	h_0, m	γ_0, rad.	χ_0, rad.	v_0, m/s
Black	3000	5000	5000	0	0	240
White	5000	5000	5000	0	0	240

Results

When the two players initially had the same velocity and altitude the result of the game turned out to be very similar, namely that the chasing player caught its opponent within

23 time steps at the highest. An example of such a case is presented in Figure 9. The same result was achieved when the chasing black player started with higher velocity.

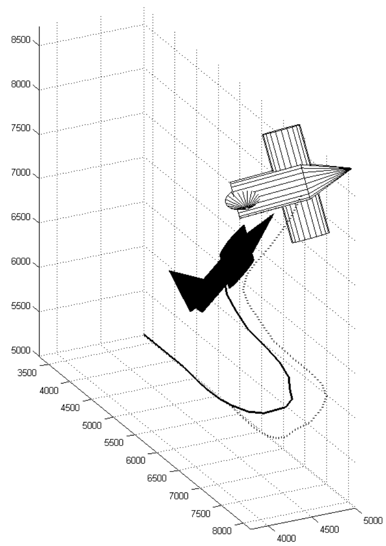

Figure 9 Black player catches white player when both start with equal altitude and velocity.

When the chased player initially was given higher altitude or higher velocity than its opponent it manages better than when starting on equal conditions. Now the chased player manages either to keep away from the opponent until the maximum number of steps is reached (N_{max}=300) or in some cases it even manages to fly away from its opponent making the distance between them bigger than the maximum allowed distance (D=12 000m) between the players which terminates the game. An example of this type of result is presented in Figure 10.

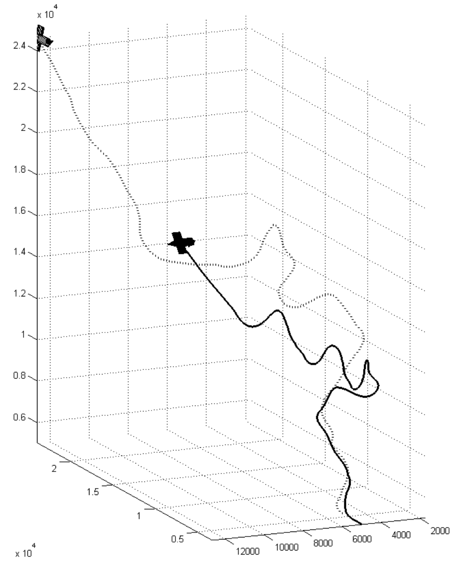

Figure 10 White player starts with higher velocity and manages to escape from its chasing opponent.

Another thing to observe in Figure 10 is that the trajectory of black player is behaving rather odd towards the end. It goes up and down like a roller coaster which cannot be considered to be the optimal way to fly. This behavior is a result of constraint (29) in section 3.2, during its ascent black player gets too low velocity than the constraint allows. So the player needs to gain more velocity, the best way to do this is to dive. After the player has gained enough velocity not to exceed the constraint, it takes up the pursuit of its opponent. In this case this happens multiple times since the player's opponent has higher altitude.

When the chasing player starts with higher altitude than its opponent the result is varying. Starting with altitude 6km and 3km respectively the white player crashes into the ground, i.e. its altitude drops below zero.

Raising both players' altitudes with 3km gives the result that white player manages to keep away from black player until N_{max} is reached. The most interesting result however is when the players start with altitude 9km and 3km respectively. Then, after a few time steps, black player suddenly becomes the chased player but manages to keep away from white player. This last result is shown in Figure 11.

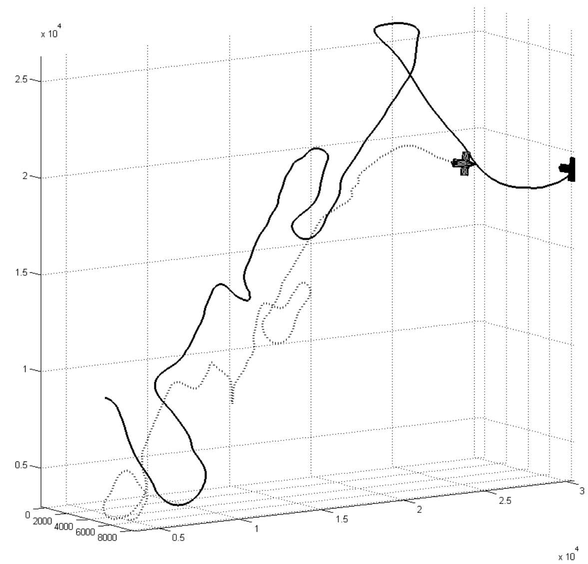

Figure 11 The chasing black player starts with 6km higher altitude but end up being chased.

At first, varying the distance did not seem to help the chased white player since both the cases with a distance of 2km and 5km between the players ended up with results very similar to the one shown in Figure 9. When the distance was increased to 8km however, white player manages to keep away from black player until N_{max} was reached. This type of result is shown in Figure 10.

The combined outcome of the chase scenario is shown in Figure 12.

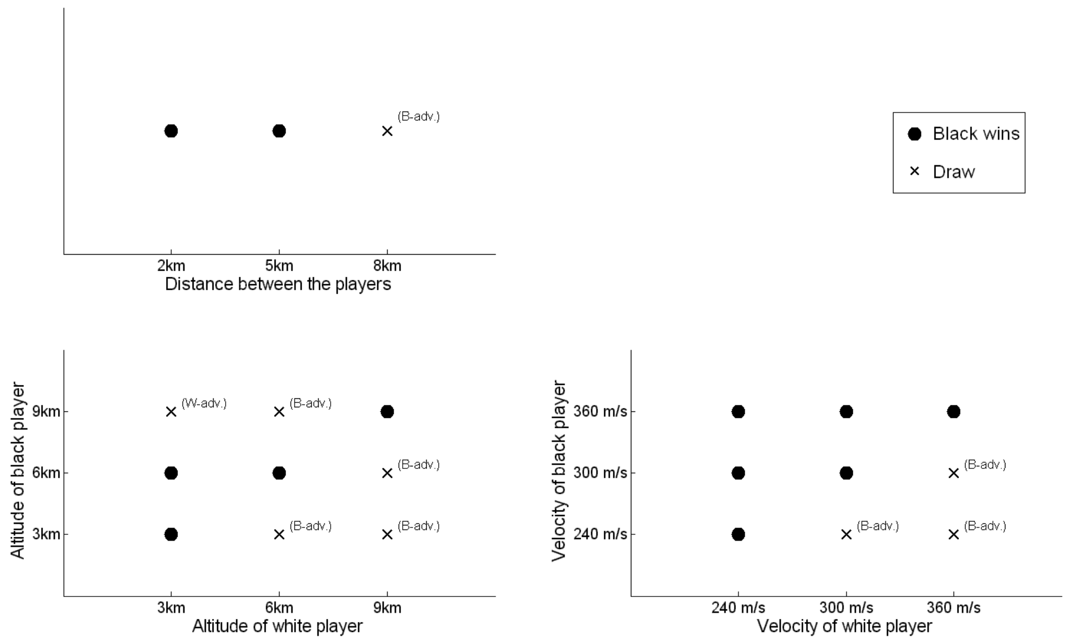

Figure 12 Combined outcome of the game in the chase scenario. B-adv. means that black player has the advantage in the corresponding test case.

Discussion

When starting on equal conditions one might argue that the player being chased should be able to at least fly straight ahead and thus avoiding its opponent. But one explanation to this behavior is that both players start with a uniform probability distribution as seen in (28) in section 3.2. This means that the player being chased believes that it has equal probability to be in any of the four states listed in Table 2 and therefore it might try to turn around towards its opponent. A trial run with a predetermined probability distribution as follows,

$$\mathbf{p}_0^B = [\ 0.1\ 0.7\ 0.1\ 0.1\ ]^T,$$
$$\mathbf{p}_0^W = [\ 0.1\ 0.1\ 0.7\ 0.1\ ]^T,$$

was examined and indeed the result turned out different. In a case with equal starting conditions the player being chased now managed to keep clear of its opponent until N_{max} was reached.

To have higher altitude or higher velocity than an opponent is an advantage in air-combat since it grants higher energy level for the aircraft [13]. So it is a good rating for the model when the player who is being chased initially start with higher altitude or velocity manages to escape from the opponent.

It is not a good rating however that the chasing player does not manage to take advantage of its advantageous position when it starts with higher altitude. The reason behind this is probably that the chasing player dives towards its opponent but misses it. Then due to its high velocity from the diving, it ends up in front of its opponent and in a disadvantageous situation. It is shown in Figure 13 that the probability that a player is in an advantageous/disadvantageous state flips over time for both players.

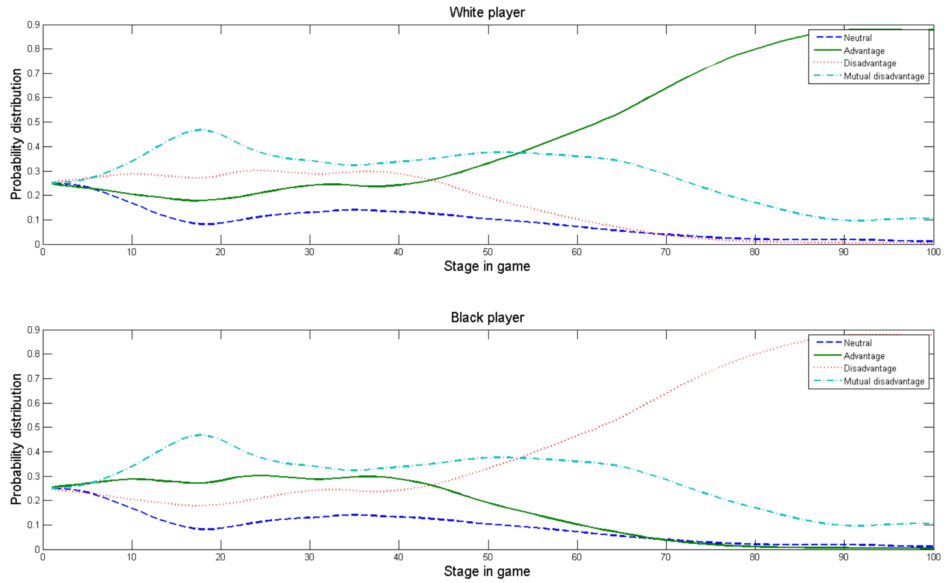

Figure 13 Probabilities of white (top) and black (bottom) player being in the possible states of the game seen in Table 2 during the first 100 time steps of a test run. The reason why only 100 time steps are shown is because after that it is one line (the one already peaking for both players) that goes towards one and the rest goes towards zero.

Neither is it good that one of the players crashes when the initial altitudes are 6km and 3km respectively even though totally avoiding crashes might be hard in such a simplified model. One problem could be that the players have so few control variable values to choose from. More possible values grants more flexibility for the players and might make it possible for a player to save itself from a crash. A test with more feasible values for the control vector (8) was therefore conducted. Now the bank angle variable is allowed to assume the value $\pm\pi/4$ in addition to the previous feasible values, for both players. The maximum rate of change is still $\pm\pi/2$ however. This time the result is different. White player does not only manage to stay in the air but ends up winning the game, even though the change to the control variables where identical for both players. The reason this particular solution is not used in this work is because of the high computational cost which additional possible values for the control variables brings. In a case where both players use a 3-step look-ahead strategy and where the game lasts for 150 time steps, the runtime of the model is increased with 600% when only adding $\pm\pi/4$ to the feasible control values. Another solution could be to implement additional or harder constraints to the ones presented in section 3.2.

Symmetrical initial states of the players should not give symmetric results in this test scenario since one player is starting with a distinct advantage over its opponent in all runs, so it is good that it does not.

4.1.2 Parallel

In Figure 7 the initial states of the players in a parallel scenario is shown. The initial default states of the players in this scenario are shown in Table 6. The altitude h, velocity v and distance x are varied the same way as in the chase scenario.

Table 6: Initial default states of the players in the parallel scenario.

	x_0, m	y_0, m	h_0, m	γ_0, rad.	χ_0, rad.	v_0, m/s
Black	0	5000	5000	0	π/2	240
White	8000	5000	5000	0	π/2	240

Results

When both players start on equal conditions regarding altitude and velocity, the results are again very similar. A game only last ~20 time steps and then ends in a joint capture. This result is presented in Figure 14.

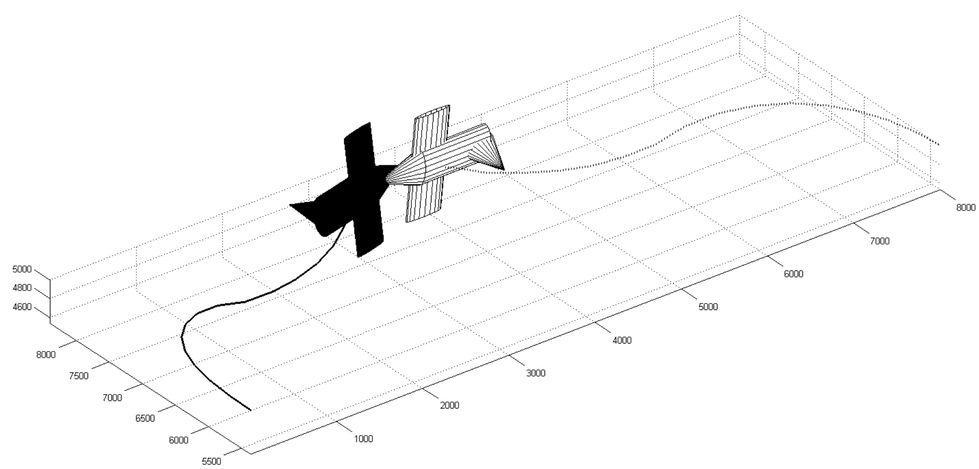

Figure 14 The players start on equal condition regarding height and speed and catch each other.

When the initial distance is varied the result is similar to the one shown in Figure 14. Only the case when the initial distance is set to 2km the result appears a little different. The game still ends in a joint capture but in this case the aircrafts can not turn against each other fast enough resulting in a little longer game. So instead of catching each other like in Figure 14 the players pass each other and then turn towards the opponent again and this time they catch each other.

If the initial altitude difference between the players is 3km, the player with the highest starting altitude wins and the trajectories of the players look very similar in all cases. The result is shown in Figure 15. When the initial altitude is symmetric, also the result is symmetric.

However if the altitude difference is 6km the result is not symmetric. When white player starts low the result is similar to the one in Figure 15 with the difference that it is the player with lowest initial altitude (white) that wins the game. But when black player starts low it is white player who ends up being hunted after a few time steps but still manages to keep clear of black player until N_{max} is reached, an outcome similar to the one in Figure 10.

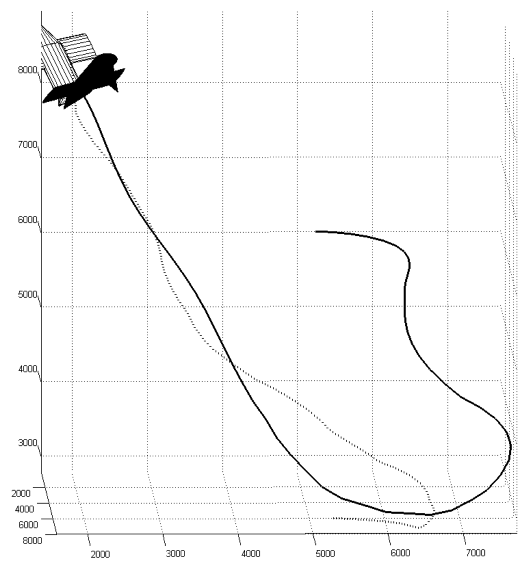

Figure 15 The players start parallel with 3km altitude difference and the one with highest initial altitude wins.

If a player has initially higher velocity than its opponent it ends up being hunted after a few time steps. It still manages to keep away from that same opponent, shown in Figure 16.

In one of the cases (and its symmetric case), when the velocity is initially at its highest difference, the player with initially the highest velocity gets in a disadvantage situation and dives to about 1km altitude. Then, when its chasing opponent tries to dive after, it does so too steep and fast making it crash into the ground (altitude below zero).

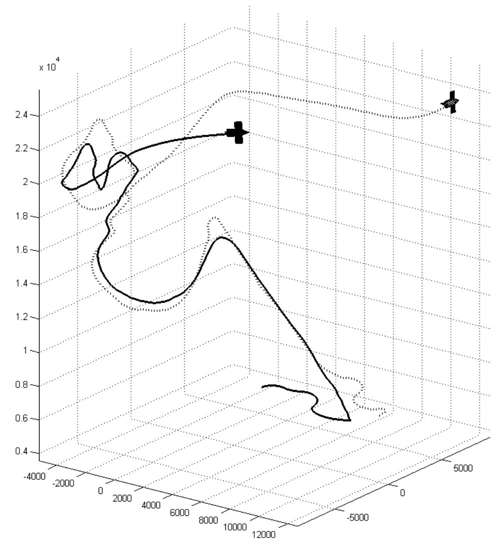

Figure 16 White player has initially the higher velocity, gets chased but manages to keep away from the opponent.

The combined outcome of the parallel scenario is shown in Figure 17.

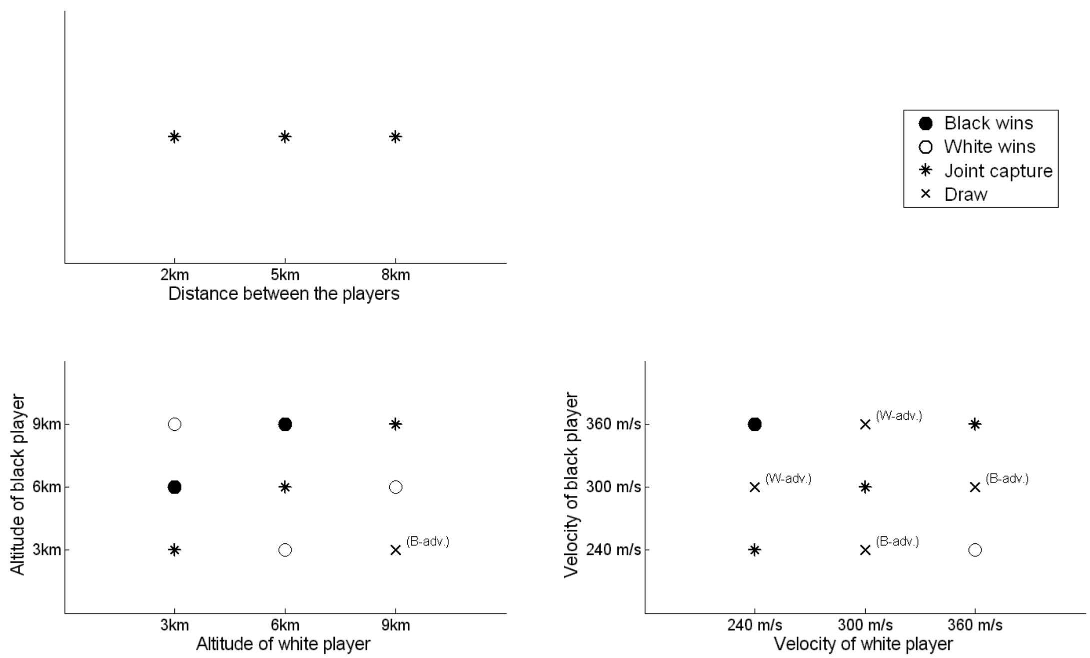

Figure 17 Combined outcome of the game in the parallel scenario. W/B-adv. means that black player has the advantage in the corresponding test case and vice versa.

Discussion

The first results presented seem reasonable. When none of the players have an initial advantage over the other they act identical. The same goes for when the distance is varied since this does not give any advantage to either of the players.

In this scenario the player with the altitude advantage manages to take advantage of the situation which is a good rating for the model. The case with initial altitude difference of 6km is not. With symmetrical initial values a result with symmetric outcome is wanted. From Figure 18 it becomes clear that in the beginning the two players starting with altitude 3km in the two cases uses (with one exception) the same actions with the same utilities. It is not exactly the same actions, since it is a parallel scenario the bank angles are negative for one player and it is the absolute value for the bank angle which is used in Figure 18. After 75 time steps something happens which makes the two players chose different actions. Since it is a 3-step look-ahead game it is hard to say exactly where the utility values come from since it depends on the probability distribution three steps ahead. The result might have something to do with the fact that the players have to turn in different directions (right, left) to approach the other player since this is the only thing that separates the conditions of the players. There is no difference in the result when the players initially switch places with each other.

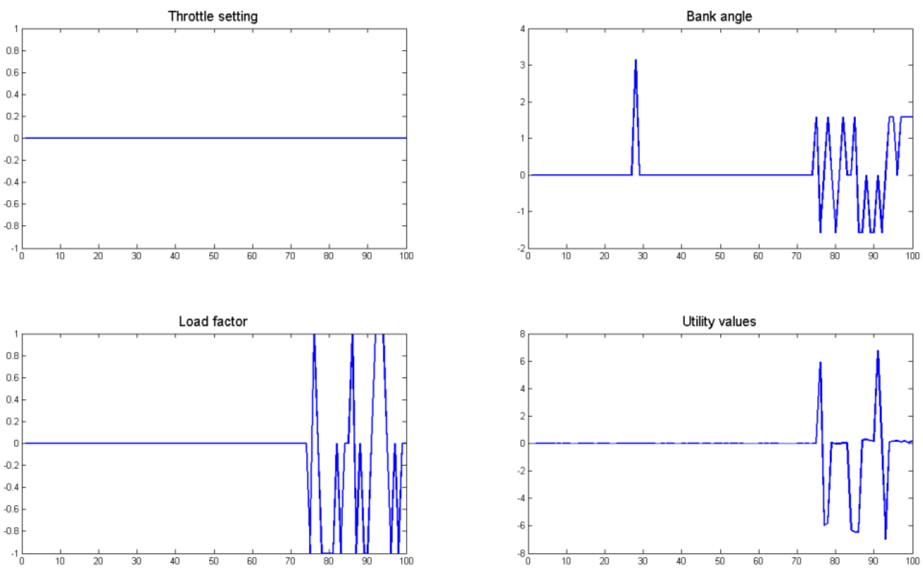

Figure 18 Difference between control- and utility values of black and white player when starting with symmetric initial states.

An initially higher velocity does not seem to be an advantage in this scenario since, in a majority of the cases, the plane with higher initial velocity ends up being hunted. This might have to do with the fact that an aircraft with lower velocity can turn faster and therefore can end up in an advantage state faster than an opponent with higher velocity.

An additional test where one player had higher initial velocity was conducted, this time with more feasible values for the control vector (8). Now the bank angle variable is allowed to assume the value $\pm\pi/4$ in addition to the previous feasible values, for both players. The maximum rate of change is still $\pm\pi/2$ however. This test resulted in a victory for the player with the lowest initial speed, none with a crash of either player. So this again shows that initial higher velocity in the parallel scenario is not preferable.

The result with a crashing player has much to do with the fact that one player does a controlled dive and the opponent wants to follow it but dives too steep and is not able to save itself in time due to the high speed obtained from the dive. A solution to this type of problem is shown in section 4.1.1.

4.1.3 Head-to-head

The initial states of the players in a head-to-head scenario are shown in Figure 8. From here, first the relative height is varied followed by variation of relative speed and distance respectively.

The initial default states of the players in this scenario are shown in Table 7. The altitude *h*, velocity *v* and distance *x* are varied the same way as in the chase scenario.

Table 7: Initial default states of the players in the head-to-head scenario.

	x_0, m	y_0, m	h_0, m	γ_0, rad.	χ_0, rad.	v_0, m/s
Black	0	5000	5000	0	0	240
White	8000	5000	5000	0	π	240

Results

In the cases when starting at the same altitude and with the same speed the players mirrors their movement completely as shown in Figure 19. There are two exceptions though, in the case when the altitude is initially at 6km for both players and the case when the velocity initially is 240m/s for both players one of them deviates from the symmetric path and begins to hunt the other as shown in Figure 20.

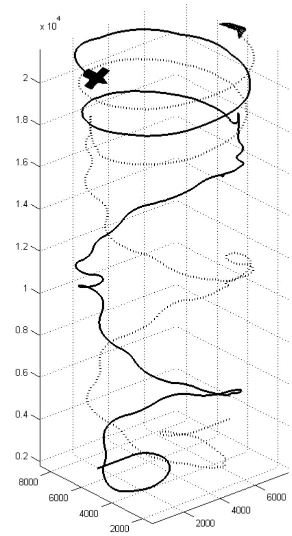

Figure 19 The players start with identical altitude and velocity which ends in symmetric trajectories.

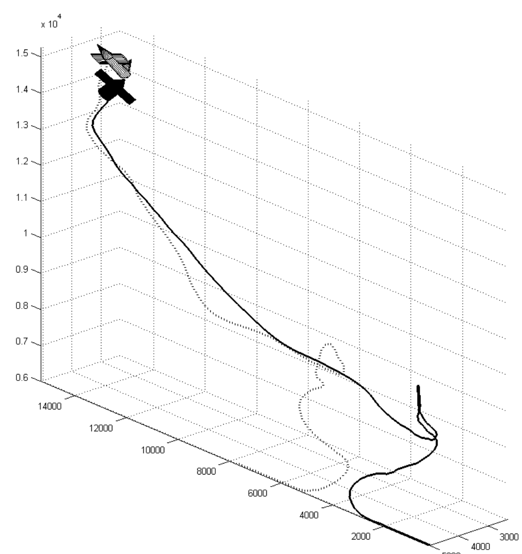

Figure 20 The players start with identical altitude and velocity but the trajectories are not symmetric.

In the cases with a distance variance of 2km and 5km, the players had little time to react before they could catch each other so the game ended in a joint capture after three and nine time steps respectively. The case with a distance of 8km between the players gives identical initial states as in the case when the initial velocity of both aircrafts are 240m/s when the test of varying speed is carried out. The result is also identical and is mentioned above.

When the altitude differs between the players the results are scattered. If the aircrafts start at 3km and 6km respectively, the player with the lowest altitude crashes into the ground, similar to one of the results in section 4.1.1.

If the aircrafts start with altitudes 3km and 9km respectively, the player with the highest altitude begins to hunt its opponent after a few time steps but the game ends in a draw, shown in Figure 21. Finally if the players start with an altitude of 6km and 9km respectively the player with the highest initial altitude ends up being chased but still manages to keep away from its opponent. The game ends with a draw similar to the result in Figure 16.

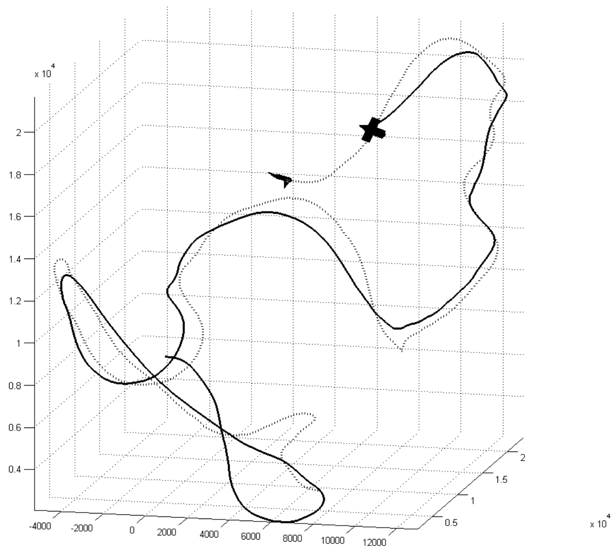

Figure 21 Black player starts with higher altitude and end up hunting white player.

The results are very scattered when the speed is varied as well. If a player with initial velocity 360m/s starts against an opponent with 300m/s it is able to begin hunting that opponent and eventually catches it. This result is presented in Figure 22.

When the initial velocities of the players had the largest difference, i.e. 360m/s and 240m/s respectively, the player with lowest velocity ended up hunting its opponent but was unable to catch it as shown in Figure 23.

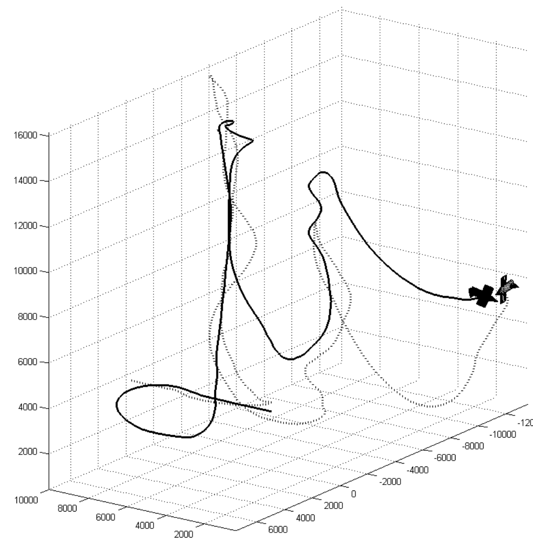

Figure 22 Black player starts with higher velocity and manages to catch white player.

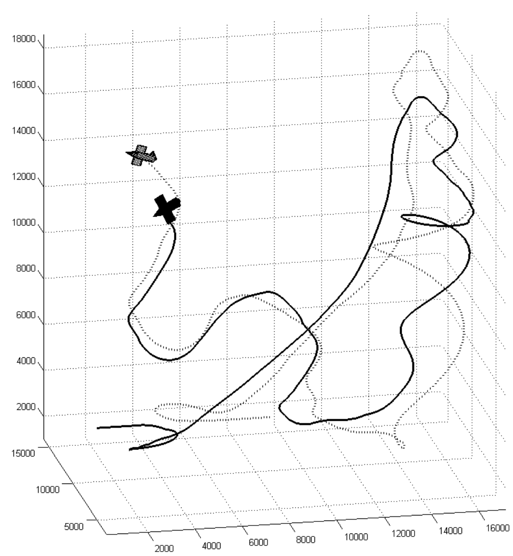

Figure 23 Black player starts with lower velocity and ends up hunting white player.

When the initial velocity of black and white player is 240m/s and 300m/s respectively white player ends up hunting black player who in turn manages to escape from white player making the distance between them exceed the maximum allowed distance between the players. This is shown in Figure 24. When the initial velocities where switched the result was different. Black player did end up chasing white player but in this case the chasing player manages to catch its opponent quite quickly as shown in Figure 25. So when the initial velocities of the players where symmetric the results where not symmetric.

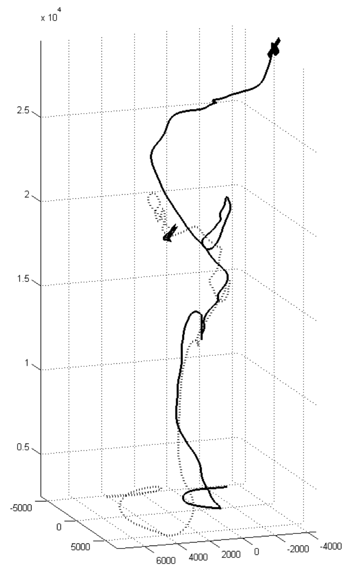

Figure 24 Black player starts with lower velocity but manages to get away from white player.

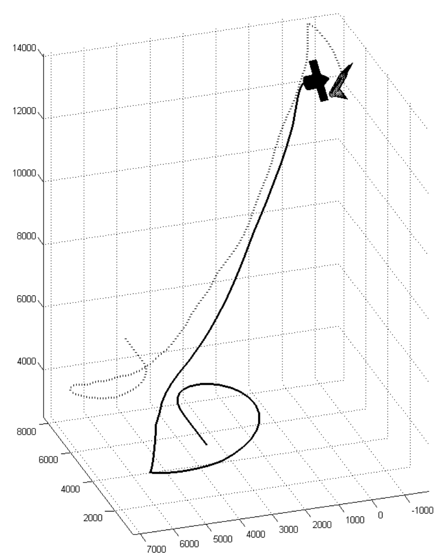

Figure 25 Black player starts with higher velocity and manages to catch white player.

The combined outcome of the head-to-head scenario is shown in Figure 26.

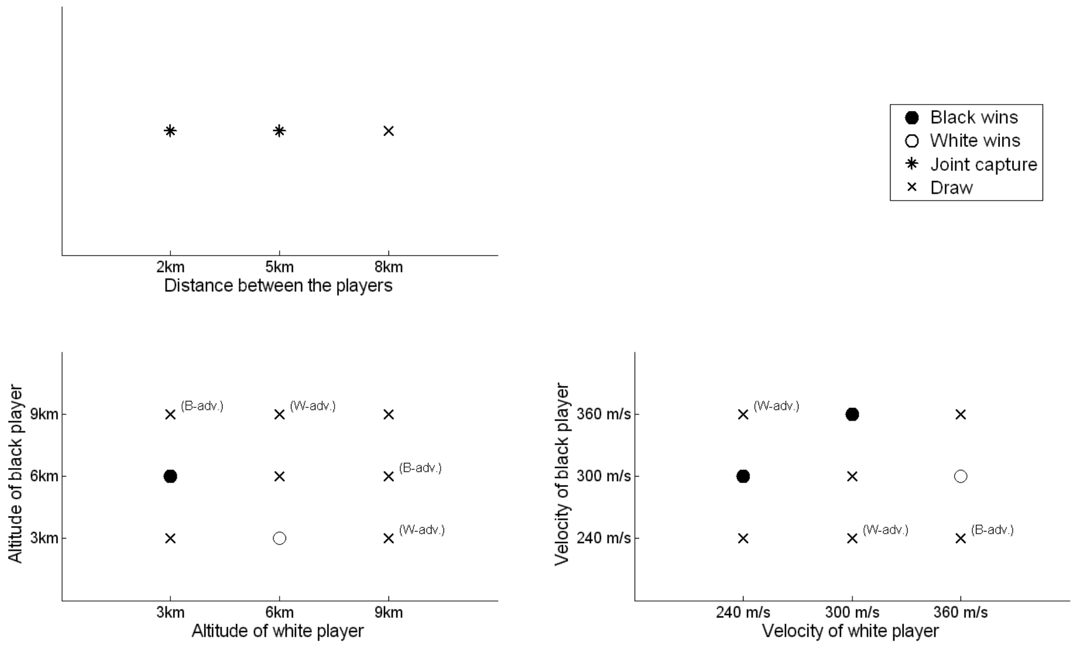

Figure 26 Combined outcome of the game in the head-to-head scenario. W/B-adv. means that black player has the advantage in the corresponding test case and vice versa.

Discussion

When both players initially have the same altitude and velocity, as in Figure 19, the players move identically, i.e. the control vector (8) is identical for both players during all time stages of the game. This seems reasonable for the model since if the players fly straight towards each other on equal conditions the utility values should be symmetric around zero when choosing the bank angle for the control vector. This means that turning right or left gives the same utility value for both players. This is the reason why sometimes the result is as in Figure 20, where one player deviates from the symmetric trajectory. Another reason for the result in Figure 20 is that if the players find themselves in states which violate one or several of the constraints (29)-(34), they focuses only on getting out of their illegal state and not on optimizing their path in regard to the opponent. This can sometimes result in different control values for the two players and in turn different results in the outcome.

It is hard to draw any conclusions from the results of varying the initial altitude of the players since they are so different from each other. The expected result would be if the player with the initial highest altitude could draw advantage of this. But this only happens in two of the cases and neither of those two end in a catch of the opponent.

Higher velocity should give an advantage in an air-combat and in this scenario it does so in a majority of the cases, some even ending in a catch of the opponent. This is a good rating of the model.

Just as in the parallel scenario there was a test case where symmetrical initial values did not result in symmetric trajectories of the players. A plot similar to the one done in Figure

18 shows that also in this case the difference between the control values as well as the utility values for each action are basically identical in the beginning for two players with symmetric initial states.

4.2 Number of look-ahead steps

The model is heavily dependent on the moving horizon technique where the next states and control vectors of a player are predicted for a certain number of look-ahead steps. This is described in more detail in section 3.1. In this section it is evaluated how the number of those steps in the look-ahead strategy affect the outcome of the game. Initially the scenarios from above are used: chase, parallel and head-to-head.

4.2.1 Chase

The basic idea behind the chase scenario is shown in Figure 6 and the initial states of both players are shown in Table 8.

Table 8: Initial default states of the players in the chase scenario.

	x_0, m	**y_0, m**	**h_0, m**	**γ_0 , rad.**	**χ_0, rad.**	**v_0, m/s**
Black	0	5000	5000	0	0	240
White	2500	5000	5000	0	0	240

Results

Figure 27 show the outcomes for each test run done in this scenario. Two of the test cases end in a draw; when black player has 1 and 3 number of look-ahead steps and white player has 2 and 3 number of look-ahead steps respectively. The game terminates because white player has managed to make the distance between the players exceed the maximum allowed distance between the players. In the rest of the test cases, black player wins.

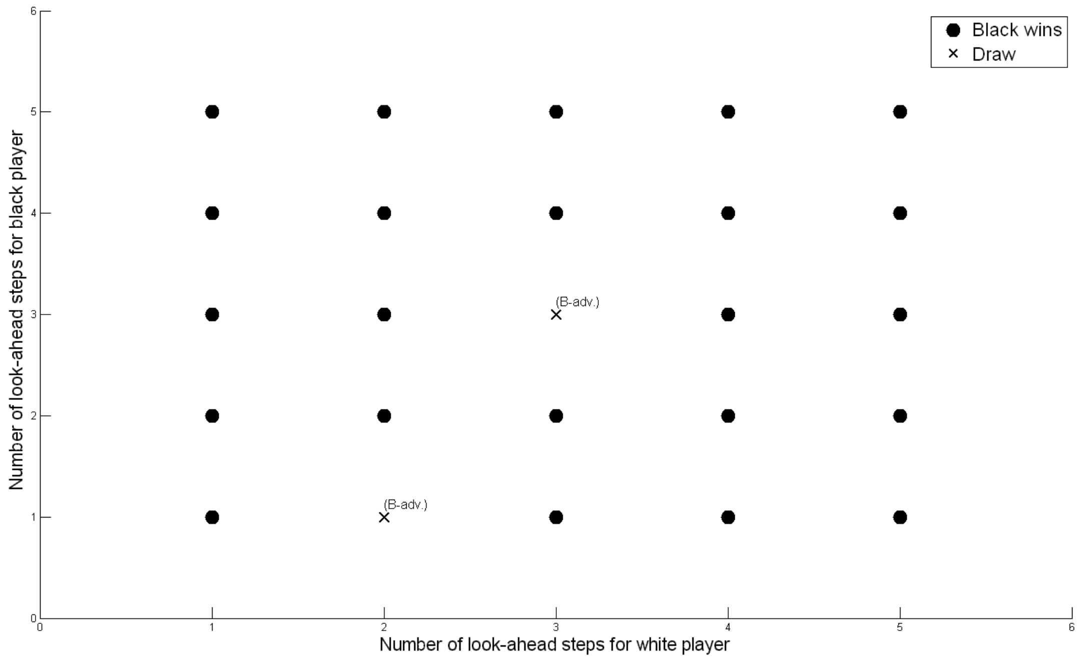

Figure 27 Outcomes of the game when different numbers of look-ahead steps are used in the chase scenario. B-adv. means that black player has the advantage in the corresponding test case.

Discussion

Figure 27 shows that when in a disadvantageous situation, as white player is in this scenario, the relation between the numbers of look-ahead steps of the players does not seem to make any difference. As mentioned in section 4.1, white player could be suffering from starting with the uniform distribution (28) set in section 3.2.

The reason why white player succeeded in the cases where the game ended in a draw with the players too far apart is that it used significantly more throttle than its opponent and thus gained more speed. A good question is then why black player does not use a higher value for its throttle variable, seeing as this probably would be in its best interest. In the case where black player only has one look-ahead step the reason is that in a majority of the time steps the utility values are identical for all throttle values, i.e. it is the player's belief that the throttle variable has no effect on the outcome. This is probably the result of using just a single look-ahead step, the result also points to this since black player manages better when that number is increased.

But black player also manages better when white players increases its number of look-ahead steps. Increasing the number of look-ahead steps should logically give the player additional information to make better decisions, which is not the case here. Perhaps the additional uncertainty that comes with increased number of look-ahead steps makes white player chose actions which are not as favorable as in the case when a 2-step look-ahead strategy was used. Additional analysis of the result is required to find out the exact reason why black player advantages from white player increasing its number of look-ahead steps. But taking all test cases in this scenario into account builds a strong suspicion that these two results are one-offs in the model.

Black players' throttle variable takes higher values in the case when both players have three look-ahead steps, but still white player uses a higher value more frequently which allows white player to climb away from black player. This time, the increased number of look-ahead steps makes it too difficult to analyze the reason behind black player's behavior.

4.2.2 Parallel

The basic idea behind the parallel scenario is shown in Figure 7 and the initial states of both players are shown in Table 9.

Table 9: Initial default states of the players in the parallel scenario.

	x_0, m	y_0, m	h_0, m	γ_0, rad.	χ_0, rad.	v_0, m/s
Blue	0	5000	5000	0	$\pi/2$	240
Red	6000	5000	5000	0	$\pi/2$	240

Results

The outcomes for the test runs done for the parallel scenario are presented in Figure 28. A clear pattern with a few exceptions can be identified, one where all test runs with equal number of look-ahead steps for the players (with the exception where they both have one) results in a joint capture, similar to the result in Figure 14. A further pattern is that whenever a player has five look-ahead steps (except when they both have it) it wins. The trajectories are often similar to the one in Figure 14, only that it does not end in a joint capture since the player with less number of look-ahead steps tries to turn away from its opponent during the last time steps but is still captured very fast.

All other test runs ended in a draw. One thing to say about the results in these cases is that the player with lower number of look-ahead steps is the one that ends up in an advantageous situation in the game. However it often ends with the distance between the players exceeding the maximum allowed distance between the players.

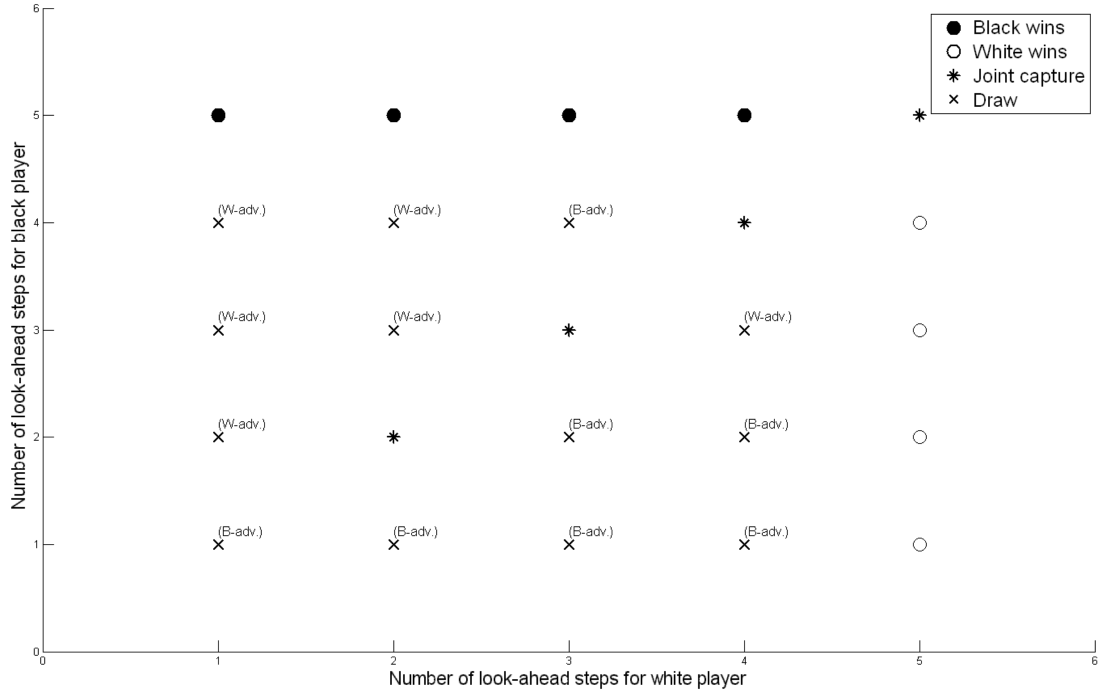

Figure 28 Outcomes of the game when different numbers of look-ahead steps are used in the parallel scenario. W/B-adv. means that black player has the advantage in the corresponding test case and vice versa.

Discussion

As shown in Figure 28, the outcomes for this scenario are symmetric, which is a good rating for the model since switching the number of look-ahead steps for the players should result in switched outcome as well. An interesting extension to this evaluation would be to continue increasing the number of look-ahead steps and see if the outcomes of the game keep the symmetric shape they have in Figure 28 or if they diverge from it.

4.2.3 Head-to-head

The basic idea behind the head-to-head scenario is shown in Figure 8 and the initial states of both players are shown in Table 10.

Table 10: Initial default states of the players in the head-to-head scenario.

	x_0, m	y_0, m	h_0, m	γ_0, rad.	χ_0, rad.	v_0, m/s
Blue	0	5000	5000	0	0	240
Red	8000	5000	5000	0	π	240

Results

The outcomes for the test runs for this scenario are shown in Figure 29. The trajectories of the players were symmetric, similar to the result presented in Figure 19, when the number of look-ahead steps where equal for the two players. One exception was when both players used a 3-step look-ahead strategy; the result was then similar to the one in Figure 20.

When the outcome was a draw, as in section 4.2.2, the player with lowest number of look-ahead steps was the one to end up in an advantageous situation.

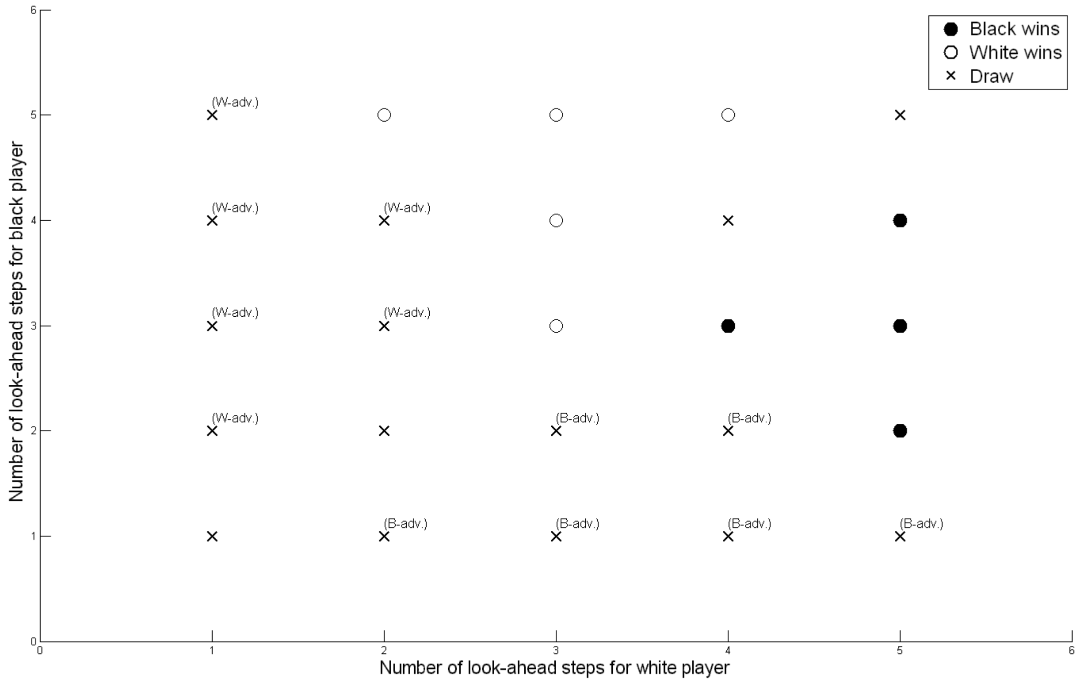

Figure 29 Outcomes of the game when different numbers of look-ahead steps are used in the head-to-head scenario. W/B-adv. means that black player has the advantage in the corresponding test case and vice versa.

Discussion

The results in the diagonal in Figure 29, where the trajectories are symmetric for almost all cases, are expected from the results in section 4.1.3. Even that there would be an exception was expected when taking the results from that section into consideration where also an explanation to why this behavior is possible is given.

In the other cases, there is clearly a form of symmetry in the results in this scenario as there were in section 4.2.2. If lines were drawn along the diagonal in Figure 29, the outcomes to the upper-left of the diagonal are advantageous for white player and the outcomes to the lower-right are advantageous for black player. Another resemblance to the results in section 4.2.2 is that when the numbers of look-ahead steps are high, there is a winner in the game. Though in this case, the winners are switched over the diagonal from results in the previous section.

Conclusions

Giving any general guidelines about how many number of look-ahead steps are best to use is difficult to do since the results between in particular the parallel and the head-to-head scenarios are so different. In the parallel case, five look-ahead steps seem to be a good choice while in the head-to-head scenario this almost guarantees a loss. If analyzing logically, an increased number of look-ahead steps should grant an advantage in the game if both players start on equal conditions as in the parallel and head-to-head scenarios. This would be results similar to the ones in the parallel scenario. But here it is more the situation that decides the number of look-ahead step which is best to use, something that is well illustrated in the chase scenario where the number of look-ahead steps not mattered at all.

4.3 Correctness when predicting next action

In this section the goal is to evaluate how accurate in its prediction the look-ahead strategy is. Let us say that a player, in time step k of a 3-step look-ahead game, is approximating the control vectors for the next three steps to $[\mathbf{u}_k, \mathbf{u}_{k+1}, \mathbf{u}_{k+2}]$. When the game later arrives at time step k+2, the evaluation is to check how far from the predicted control vector in time step k the control vector actually implemented in time step k+2 is. With a perfect system the predicted and the actually implemented control vectors are identical. However, as the number of steps *k* in the k-step look-ahead strategy is increased, the uncertainty of the system should increase and thus affecting how well the control vector can be predicted.

The chase- parallel- and head-to-head scenario presented in Figure 6, 7 and 8 respectively are used in this evaluation with the same initial values as in section 4.2. A 5-step look-ahead strategy is used for each scenario.

For all scenarios, control vectors used when either player is in an infeasible state according to the constraints in section 3.2 are removed from the following results. This is because the players are not planning when in such a state, their only goal is to get out of it.

4.3.1 Chase

As in the previous evaluations, it is white player that has the initial disadvantageous position. The game ended after 50 time steps and the outcome was a black player victory with a trajectory result similar to the one presented in Figure 9.

The mean value of the absolute difference for all three variables at the different predicted steps (for a certain prediction step, sum the absolute difference in all time steps together and divide by the total number of time steps in the game) are displayed in Figures 30-32. With this mean value it is easier to see the difference in prediction correctness between the different prediction steps. A mean value close to zero is of course preferable. The difference to the implemented control variables gradually increases in step two, three, four and is at it highest in the fifth approximation of the control vector.

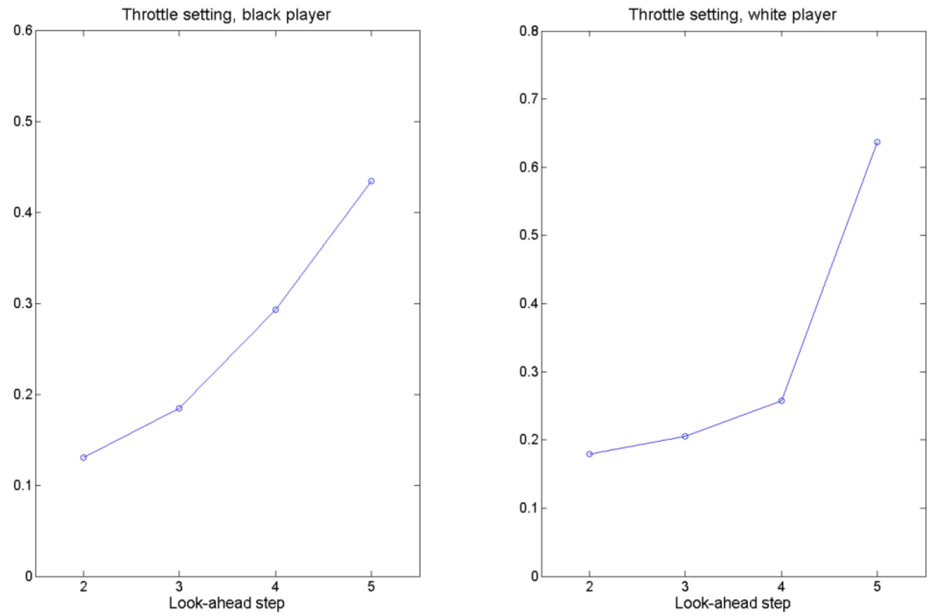

Figure 30 Mean values for the absolute difference of the throttle setting variable for both players in the chase scenario.

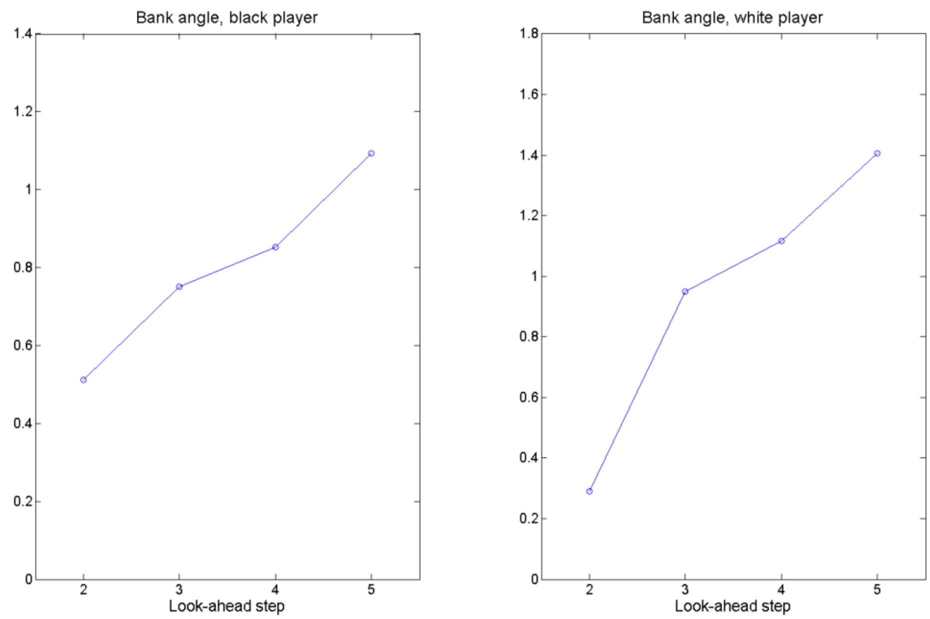

Figure 31 Mean values for the absolute difference of the bank angle variable for both players in the chase scenario.

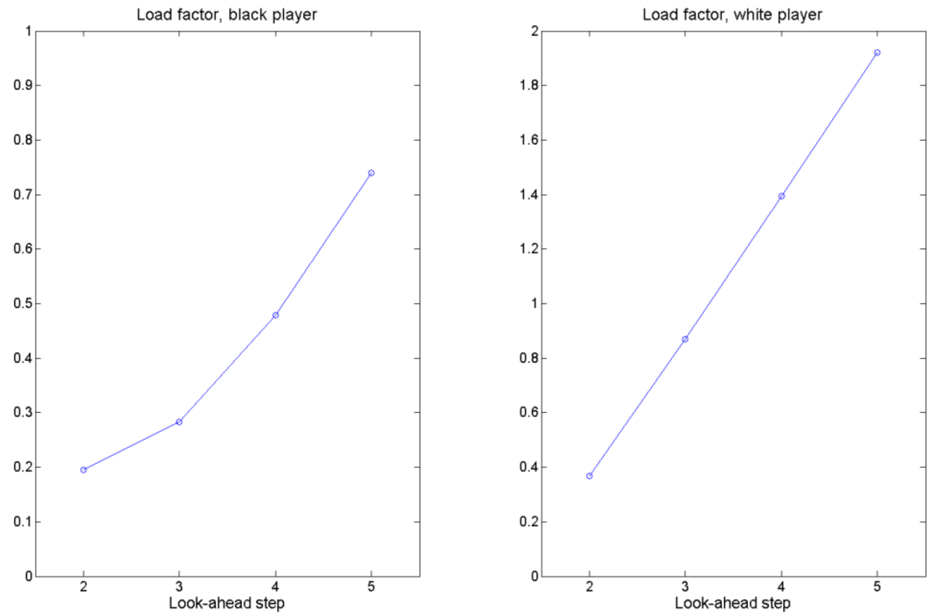

Figure 32 Mean values for the absolute difference of the load factor variable for both players in the chase scenario.

Discussion

As shown in Figures 30-32, black player approximates its control vector in general and its load factor in particular better than white player, even though it is only marginally better. This might have to do with the fact that it is easier to approximate your control vector when in an advantageous situation in the game which black player is in this scenario. However, to make that conclusion would require more than just one test case pointing to that result.

4.3.2 Parallel

The game in this scenario ended after 15 time steps with a joint capture and the result of the game was similar to the one presented in Figure 14.

The mean value of the difference for all three control variables at the different predicted steps are displayed in Figures 33-35. The difference to the implemented control variables does not gradually increase as in section 4.3.1, only in the case of the load factor variable is this true. Instead, the difference to the implemented bank angle gradually decreases and the difference to the implemented throttle setting both increase and decrease for each approximation step.

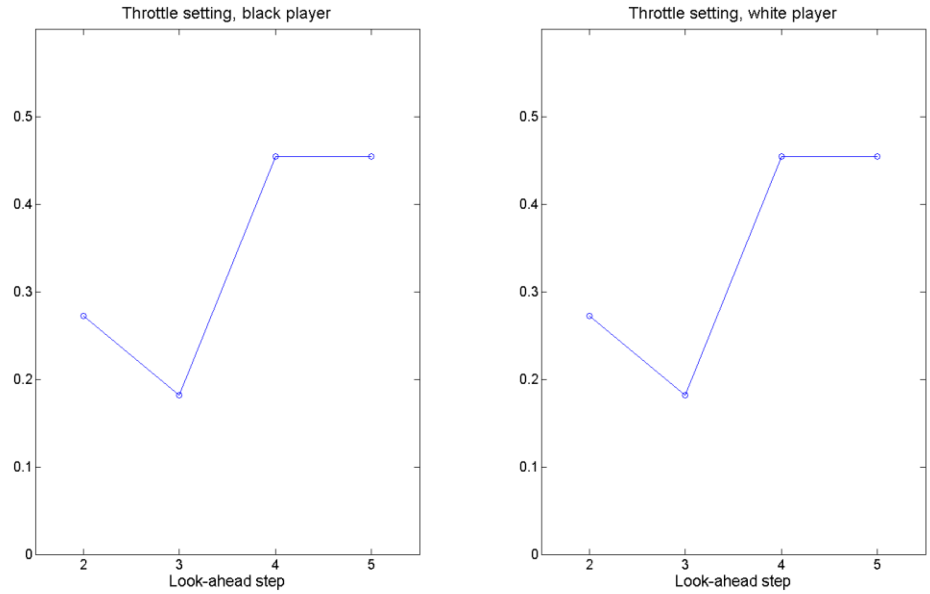

Figure 33 Mean values for the absolute difference of the throttle setting variable for both players in the parallel scenario.

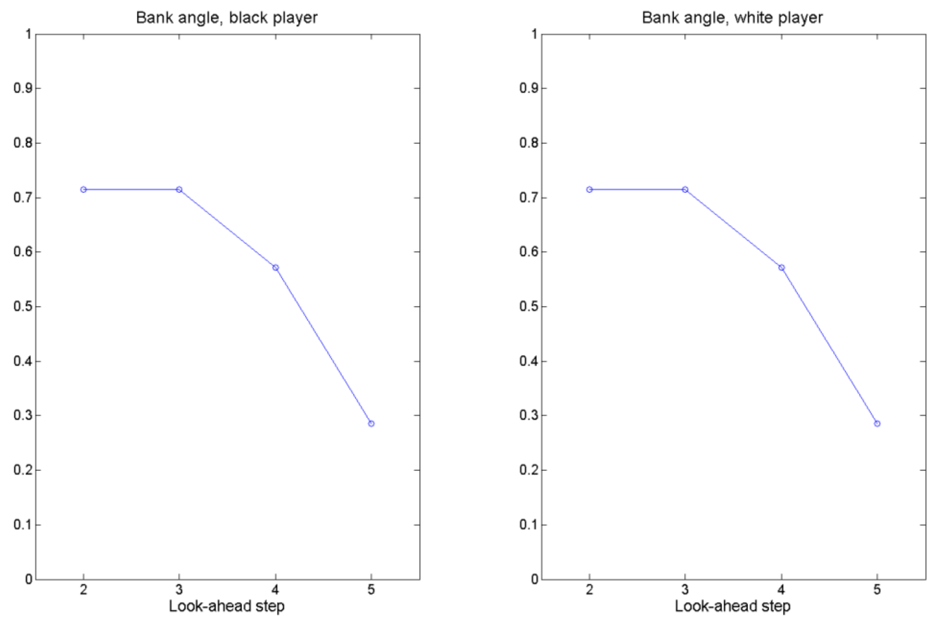

Figure 34 Mean values for the absolute difference of the bank angle variable for both players in the parallel scenario.

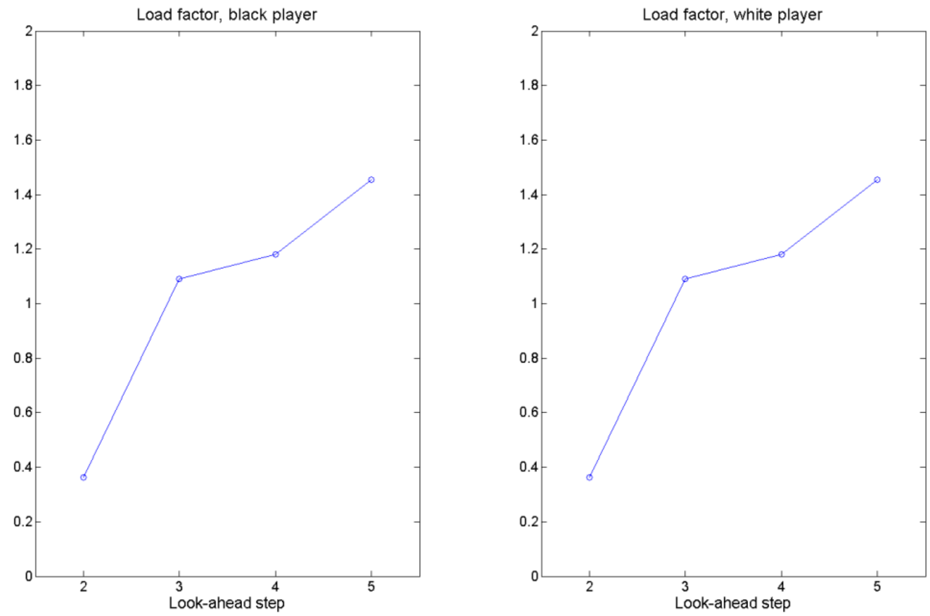

Figure 35 Mean values for the absolute difference of the load factor variable for both players in the parallel scenario.

Discussion

The mean values are identical for all control variables, this seems logical if the trajectory result is of the kind presented in Figure 14 where the players simply turn in the direction of the other. The bank angle should be negated for one player but it is the absolute mean value shown in Figure 34.

Still, the result does not appear as expected. For the throttle setting, the approximation done in the third step is the best one according to the mean values in Figure 33 where the second one is expected to be the best. The mean values for the difference of the bank angle variable even decreases as the approximation step increases. This should be the other way around, more like the results in section 4.3.1 and as the load factor in this scenario. However, both the throttle setting as well as the bank angle varies within their full interval, [0 1] and [$-\pi/2$ $\pi/2$] respectively and among only three numerical values in every time step. This means that it is easier for the approximation to get the values for these two variables right, since there are so few of them. It is easier to 'get lucky', which might be the case here. There are also very few time steps compared to the other scenarios in this evaluation which is something that also suggests that this result is a one-off or at least not something to draw conclusions out of.

4.3.3 Head-to-head

For this scenario, the test is done with different number of look-ahead steps between the players. White player uses a 4-step look-ahead strategy and black player uses a 5-step look-ahead strategy in the game. This is because, if both were to use a 5-step look-ahead strategy, the trajectory of the players would be symmetric (similar to the result in Figure 19 and also seen in section 4.2) which, in turn, would mean symmetric control vectors. Since it is the difference in control vectors that is examined in this section, it is more interesting to examine a result that is not as predictable as a result with symmetric trajectories would be.

The trajectory result of the game is similar to the one in Figure 25, only the roles are switched and it is white player that is chasing and catching black player.

The mean value of the absolute difference for all three variables at the different predicted steps are displayed in Figure 36-38. Here, as well as in section 4.3.1, the difference to the implemented control variables gradually increases for each approximation step.

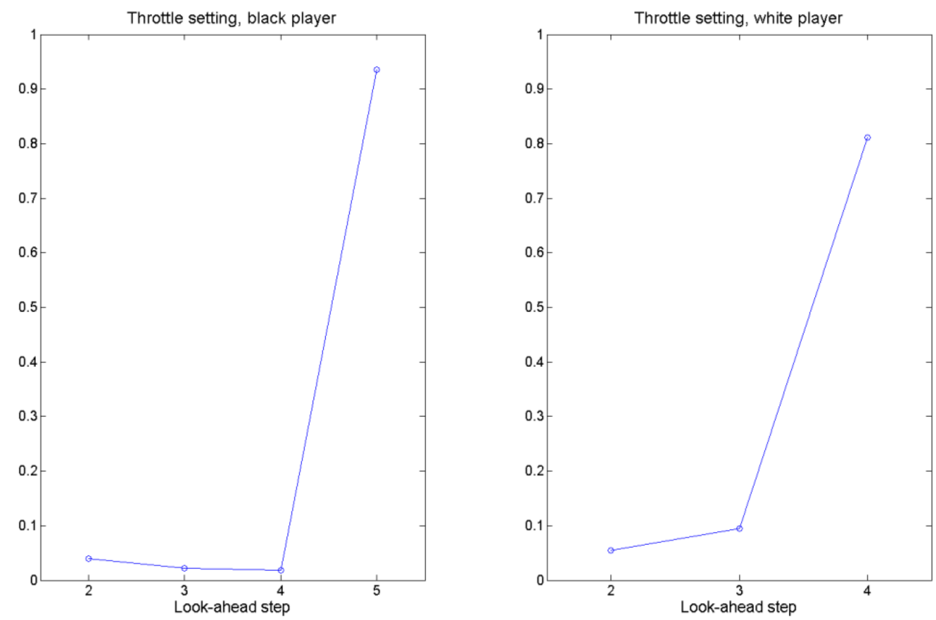

Figure 36 Mean values for the absolute difference of the throttle setting variable for both players in the head-to-head scenario.

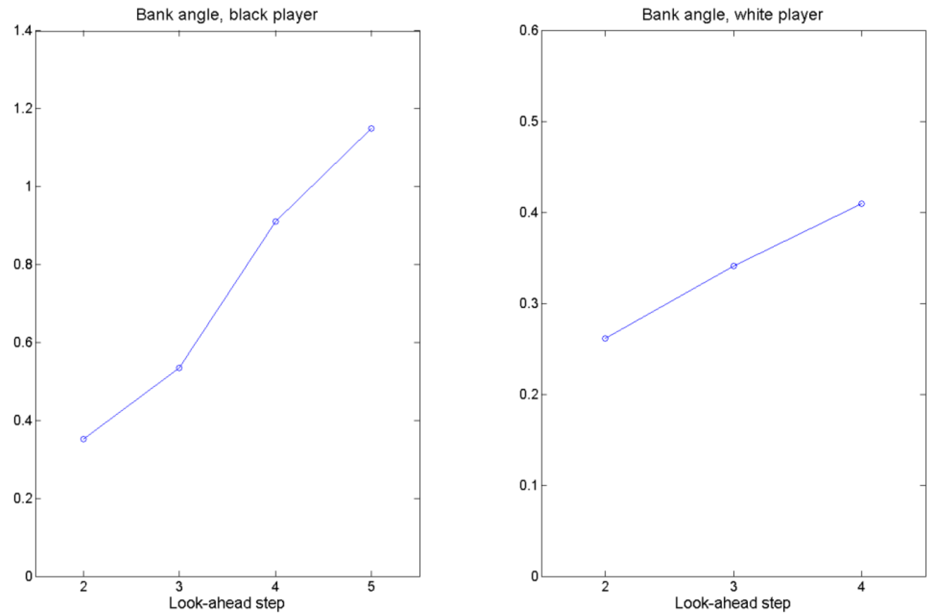

Figure 37 Mean values for the absolute difference of the bank angle variable for both players in the head-to-head scenario.

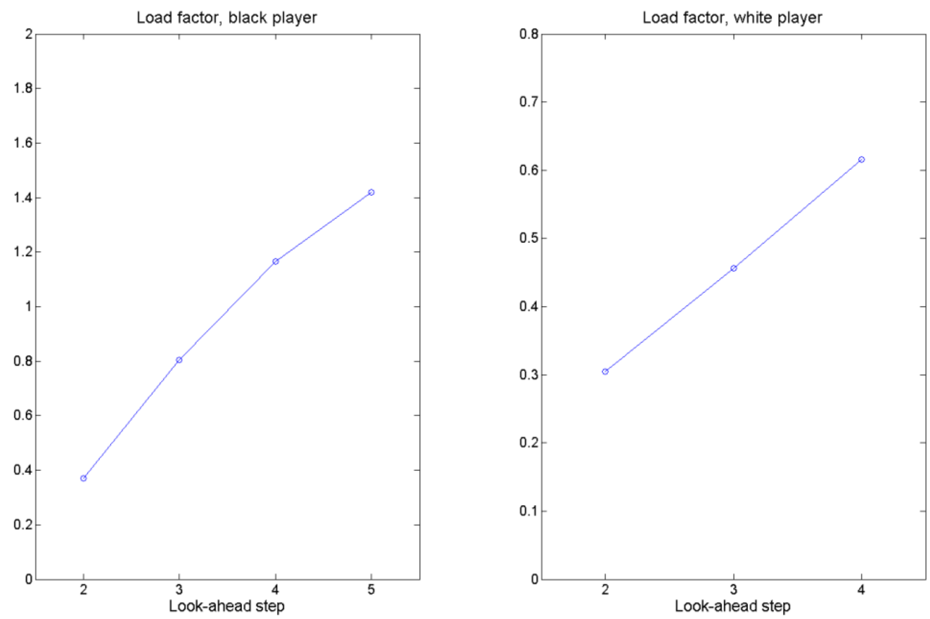

Figure 38 Mean values for the absolute difference of the load factor variable for both players in the head-to-head scenario.

Discussion

One odd thing about the results presented for the head-to-head scenario is that in the last prediction step the throttle setting is predicted very poorly compared to the previous done predictions for each player. In the other evaluations the mean value of the throttle setting difference has barely been over 0.5, now it approaches 1. Observe that this also happens in different approximation steps for the two players, four and five respectively, but still in the last step predicted. What this depends on would require additional testing to draw any accurate conclusions about.

However, the results in section 4.3.1 pointed towards that a player in an advantageous situation in the game was better at approximating the control vectors. This also seem to be true in this scenario as the chasing white player's mean values are considerably lower for the bank angle and load factor variables.

4.4 Different modeling of the opponent's decision strategy

When a player chooses its control vector, as described in section 3.1, it compares a possible control vector against all of the opponent's possible control vectors. That is, the player assumes that the opponent chooses its control vector with a uniform probability. This section introduces other ways of assuming how the opponent makes its decisions and compares those against each other.

Uniform probability

This is the technique used in the previous evaluations and the one explained in section 3.1. Basically, for a certain possible new state, a player uses the utility value from all possible states the opponent can reside in the next step and sums them together. This means that the players believe that it is equally possible for the opponent to take any of its possible actions, hence the name uniform probability strategy.

Min-max solution

In the game presented in chapter 3, the players always have full view of its opponent whereabouts, which means that they always know what the best action to take in concern to where the opponent is. So now when a player chooses its control vector, instead of taking every possible control vector of the opponent into account, the player assumes that the opponent optimizes its decision and chooses the action which maximizes the utility for itself and thus minimizing the utility for the player. The player then maximizes its own utility so to choose the action which is the least favorable for the opponent.

Weighted probability

This is a variant of the two methods above. The player still takes every possible control vector of the opponent into account, but those actions are weighted in the sense of which action a player beliefs its opponent is most likely to chose. In this case the actions which grant the opponent higher altitude and/or higher velocity are considered more likely for the opponent to chose and are therefore weighted more.

Both players use a 3-step look-ahead strategy in all cases during this evaluation. The scenarios and initial states for both players in all test cases are the same as in section 4.2. The results are presented also in a similar way as in section 4.2; the three different strategies for modeling the opponent's decision are all compared against each other giving a total number of nine test runs for each scenario.

Results

Figure 39 show the outcomes of the game for the chase scenario. In the cases when white player uses the uniform probability strategy of modeling black player's control decision the game always ends in a draw. In two of those cases the game terminates because the distance between the players exceeds the maximum allowed distance between players, this result is shown in Figure 10. In all the other cases black player wins in a similar manner to the one presented in Figure 9.

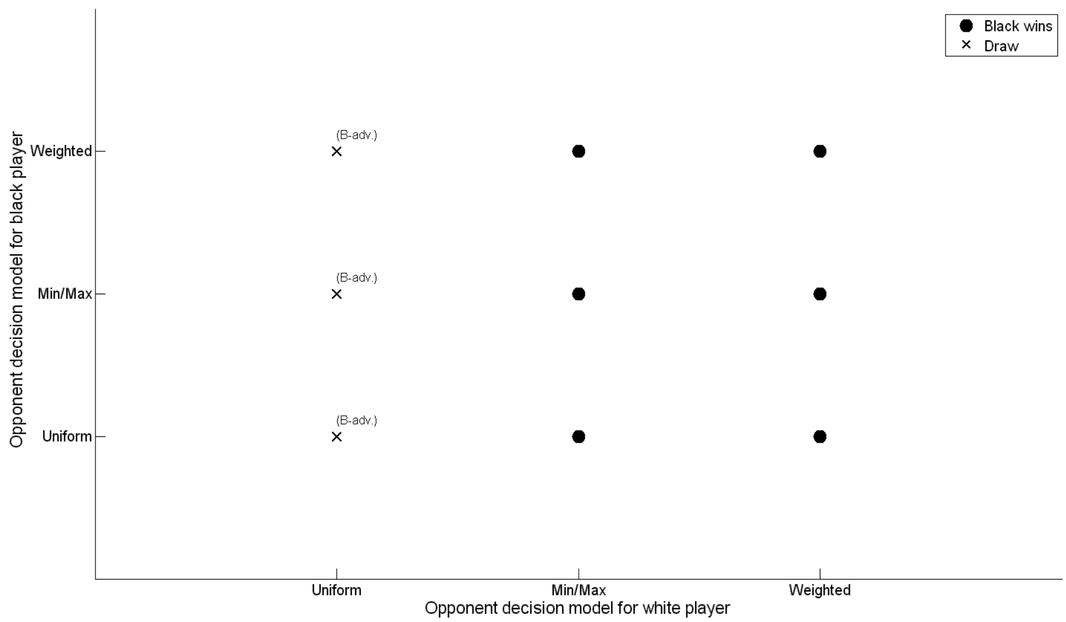

Figure 39 Outcomes of the game when varying the strategy for modeling the opponent's decision in the chase scenario. B-adv. means that black player has the advantage in the corresponding test case.

Figure 40 show the outcomes of the game for the parallel scenario. When either of the players uses the min-max strategy of modeling the opponent's decision they end up in an advantageous situation and in one case black player even manages to catch white player. The exception is when both players use the min-max strategy, then the game ends in a joint capture.

In all other cases the game ends in a joint capture similar to the result in Figure 14.

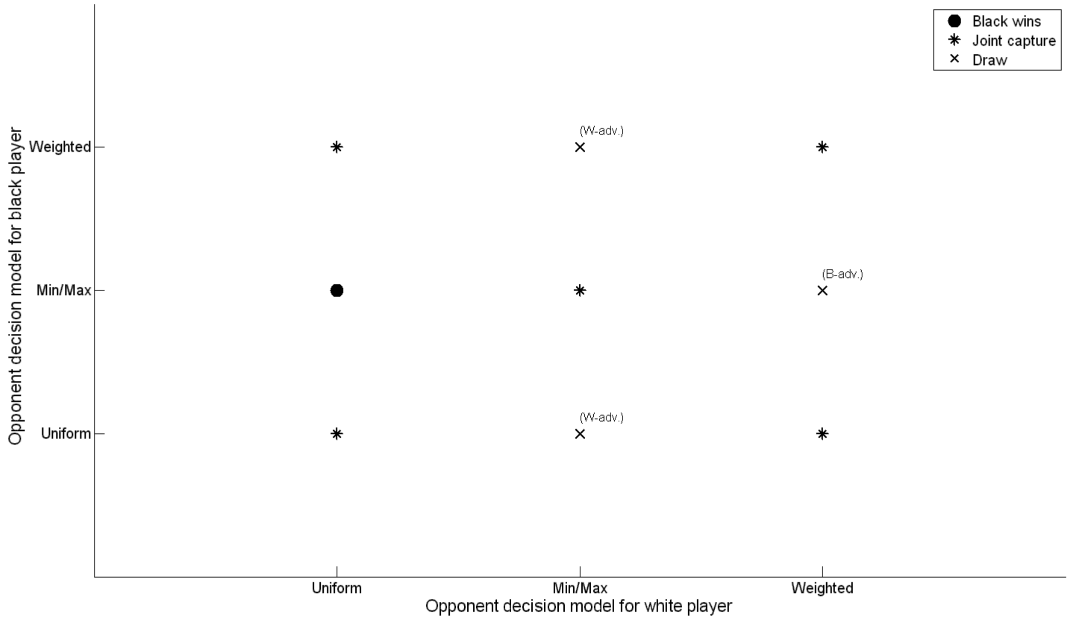

Figure 40 Outcomes of the game when varying the strategy for modeling the opponent's decision in the parallel scenario. W/B-adv. means that black player has the advantage in the corresponding test case and vice versa.

Figure 41 show the outcomes of the game for the head-to-head scenario. When both players use the uniform probability strategy the result is similar to the one in Figure 20 but with a white player victory. A reason why the result can be inconsistent with a white player victory instead of a black player victory as in Figure 20 in this case is discussed in section 4.1.3. In the other two cases when the players use the same strategy for modeling the opponent's decision the game end in a draw with symmetric trajectories for the players similar to the result in Figure 19.

When either of the players uses the min-max strategy against either of the other two strategies it ends up in a disadvantageous situation. The game still ends in a draw though. The last two cases end in a draw with the player using the uniform probability strategy ending up in an advantageous situation similar to the result in Figure 23.

One notable result for the head-to-head scenario is that a player using the min-max strategy for modeling the opponent's decision and who end up in an advantageous situation manages to follow its opponent very well even though it does not end in a catch of the opponent, like the result in Figure 21.

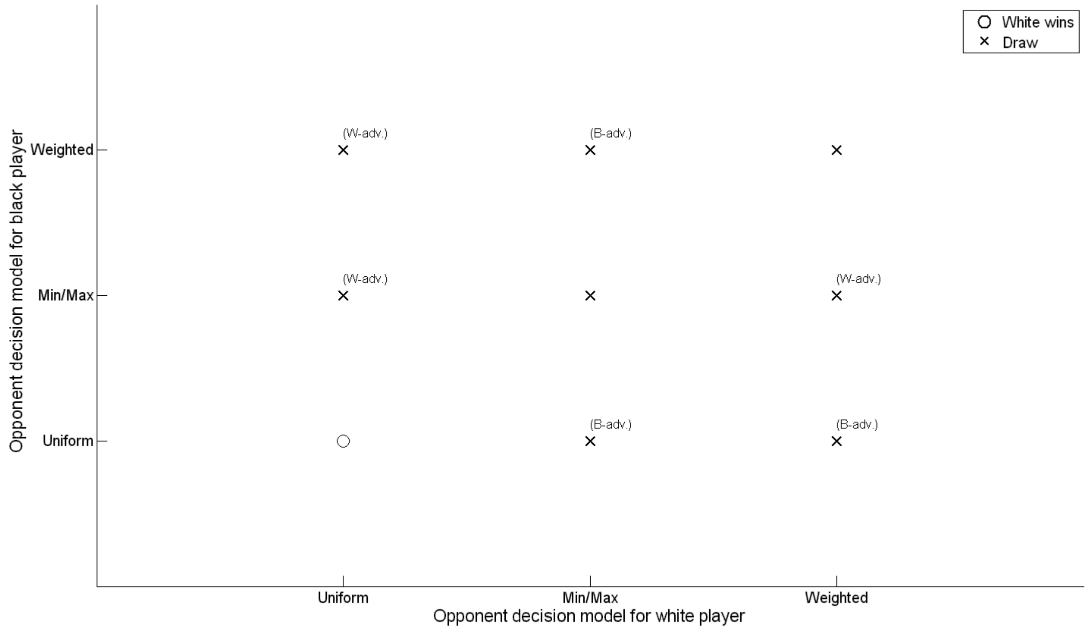

Figure 41 Outcomes of the game when varying the strategy for modeling the opponent's decision in the head-to-head scenario. W/B-adv. means that black player has the advantage in the corresponding test case and vice versa.

Discussion

From the chase scenario it is clear that a uniform probability modeling of the opponent's decision is the best strategy to use when being chased as white player is in that scenario. The results in section 4.2 stated that using different number of look-ahead steps did not help the chased player in any way. But here are some results that point towards, that by using the uniform probability way of modeling of the opponent's decision, the chased player can better manage to escape its opponent. However, tests done with varying the number of look-ahead steps for the two players while also varying black player's way of modeling of the opponent's decision gave results similar to the one shown in Figure 9 where black player catches white player after a few time steps. With this information it seems that the uniform probability strategy is not the optimal strategy for a chased player after all. Instead it appears as if it is the case when both players use a 3-step look-ahead strategy which is advantageous for the chased player, as it was in section 4.2.

The fact that the game ends in a joint capture when both players use the same way of modeling the opponent's decision is expected from the results in section 4.2 where equal conditions for the players made them turn towards each other in the parallel scenario, making the game end in a joint capture.

Using the min-max strategy for modeling the opponent's decision gives an advantage in the parallel scenario. But it is an inconsistency when the strategies min-max and uniform probability are switched between the players but the result is not, as shown in Figure 40. Inconsistencies are not good for the modeling but there have been such results in the previous evaluations. Examining the utility values for each action in a certain time step more thoroughly and comparing the two inconsistent test runs against each other showed a pattern. Namely that when the control vectors for two players using the uniform probability strategy differs, the players are in an infeasible state. It is the bank angle which differs, or rather it is identical in the comparison. But since it is a parallel scenario the bank angle should be negated when comparing it between the two players. This means that the function described in section 3.1 is used to find a suitable control vector so that the player can get out of its infeasible state. That function does not take any consideration to where the plane is or where its opponent is, only to get the player into a feasible state again. So the function gives identical bank angle to both players where, if the game should be symmetric, the function should give a negated bank angle to one of the players. This result has been seen in parallel scenarios in previous evaluations so perhaps using additional possible values for the bank angle or letting the function described in section 3.1 take more parameters into consideration could solve this problem.

As in section 4.1 and 4.2 the trajectory result of the players are symmetric when they are starting on equal conditions. Also as seen previously there is an exception and in this case it is when both players start with the uniform probability strategy, so this result was expected.

Once again there is an advantage/disadvantage pattern when the players use the min-max strategy. But this time, in the head-to-head scenario, it is reversed. This time it is the opponent which gets the advantageous situation when a player uses the min-max strategy. This is somewhat strange since logically the min-max strategy would be the best one, which it is in the parallel scenario. Examining the predicted control vector of the opponent and the actually implemented control vector gives a partial explanation. In a test with only one look-ahead step for both players, the min-max strategy manages perfectly when predicting the opponent’s control vector. But as the look-ahead steps increase, so does the uncertainty in predicting. In a test with a 2-step look-ahead strategy for both players are approximately 1/10 control vectors wrongly predicted. Increasing the number of look-ahead steps to three, the same number as in the test above, increases the error to approximately 2/10 control vectors.

Another way of looking at it is that by using the uniform probability strategy a player end up in an advantageous situation since also in the case of a uniform vs. weighted probability strategy, the uniform gives an advantage over the weighted. The weighted strategy does not grant any obvious advantages. Perhaps if the players actively tried to gain additional altitude or velocity this strategy would be more useful.

5. Conclusions and future work

This thesis introduces a model of the successive control decisions in one-on-one close range air combat using a multi-agent influence diagram. Due to the complexity of dynamic programming, the influence diagram is truncated by a moving horizon technique. The motion of the players is described by a set of differential equations and the relation between them by a combat state vector, consisting of the distance between the aircraft and the angles between the velocity vectors and the line of sight. A player estimates the probability that it resides in any of four possible states; neutral, advantageous, disadvantageous and mutual disadvantageous. The pilot's preferences are modeled by utility functions, one for each possible state, and the probability distribution for residing in each state works as a weight for those utility functions. In each time step, the players are maximizing the cumulative payoff (calculated by the sum of the utilities for each state, weighted with the corresponding probabilities) each possible action gives.

The evaluation was done in three different scenarios; chase, parallel and head-to-head which are shown in Figures 6, 7 and 8 respectively.

Section 4.1 shows that there are some inconsistencies when the initial states of the players are varied. The few cases with players crashing are disturbing and not a very good rating for the model. However, additional feasible control values are proposed as a solution in the discussion of that chapter and are also proven to help. The reason why these additional feasible control values are not used in the evaluations of this thesis is because of the increase in computational complexity they bring. This is one aspect where the computational complexity of the model becomes a hinder in the usability of it. The computation time of a modeling depends on how many look-ahead steps being used and of course how many time steps it takes before the game terminates. A game where both players use a 3-step look-ahead strategy and where the game terminates after 100 time steps has a computing time of 1 minute and 31 seconds to complete. Whereas if the players use a 5-step look-ahead strategy in the same number of time steps, the game has a computing time of 12 hours, 37 minutes and 31 seconds. This clearly demonstrates the computational explosion of dynamic programming.

Another example where a decrease in the computational complexity is wanted is in chapter 4.2 where additional number of look-ahead steps would be required to draw any real conclusions from the parallel and head-to-head scenarios. However, modeling the air combat with six look-ahead steps for both players takes over 24h to complete with the current implementation which is too much to be included in this work. Using five look-ahead steps are still more than the three used in [1] which this work is inspired by, which indicates that improvements are possible. The results from the chase scenario do however very convincingly show that an advantageous position is far more important than the number of look-ahead steps used. In the discussion of [1] it is suggested that a longer planning horizon benefits a less agile player. One of the examples used in [1] also shows that if the agility is restrained for the chasing player, its opponent is able to not just to get away, but even to swing around behind the initially chasing player and win the game.

The evaluation of the correctness when predicting the control variables in the future time steps gave reasonable results. Since the uncertainties about the game increase with every step, it is not possible to ask that the model predicts every control correct. Instead the result in section 4.3 shows that it becomes harder for each new step ahead to predict the control variables.

Section 4.4 introduced different strategies for modeling the opponent's control variables. The uniform probability strategy which is used in all other evaluations, a min-max strategy where assuming that the opponent always chooses the optimal action for itself and a weighted probability where actions leading to higher speed and/or altitude assumed more likely to be chosen were all tested against each other. The uniform probability strategy of modeling the opponent's decision seems to be a winning concept when being chased. But in the discussion in that same section it becomes clear that if the opponent chooses another number of look-ahead steps that advantage is gone. So, as in section 4.2, the conclusion is that an initial advantageous position is more important than any strategy (at least those tested in this work). For the parallel and head-to-head scenarios, using the min-max strategy is favorable in one scenario but unfavorable in the other. This makes it hard to draw any general conclusions.

The results from sections 4.1 and 4.3 shows that this model is robust and behaves reasonable from a real world air combat point of view, or at least can be made to behave reasonable if given additional feasible control values. Sections 4.2 and 4.4 are more about testing the capabilities of the model.

To summarize, influence diagrams grants an obvious way of taking decisions based on several different inputs in a world of uncertainty which is the case in air combat. The presented model and its results during the evaluation clearly shows that a working and reasonable model can be obtained from using influence diagrams to model close range air-combat. The model's usability areas are many; everything from analyzing different aircraft models or tactics against each other to support in pilot training, in for example air combat simulators.

For future work, here are some ideas on how the model could be improved. As mentioned above does the computational heaviness of the model perform as a hinder, both when having to truncate the influence diagram by the moving horizon technique but above all when having to limit the control variables for the players. But there are several ways of decreasing the runtime for the model. Parallelism has not been taking advantage of in the implementation done in this thesis and that would be the most obvious method of speeding up the modeling. Even moving some of the calculations to the GPU could be a valid solution which would grant even greater speedup.

Making the players act more as pilots in a real life air combat is another thing which would make the model more interesting. This could be done by extending the information included in the combat state vectors. In this thesis, the combat state vector consists of the distance between the aircraft and the angles between the velocity vectors and the line of sight. One example of an extension is features describing the energy levels between the players, which is an vital component in an air combat [13].

References

[1] - K. Virtanen, J. Karelahti, and T. Raivio, “Modeling Air Combat by a Moving Horizon Influence Diagram Game”, Journal of Guidance Control and Dynamics, vol. 29, no. 5, pp. 1080-1090, 2006.

[2] - Howard, R. A., and Matheson, J. E., “Influence Diagrams,” The Principles and Applications of Decision Analysis, edited by R. A. Howard and J. E. Matheson, Vol. 2, Strategic Decision Group, Palo Alto, CA, 1984, pp. 719–762.

[3] - Shachter, R. D., “Evaluating Influence Diagrams,” Operations Research, Vol. 34, No. 6, Stanford University, Stanford, California, 1986, pp. 871–882.

[4] - Koller, D., and Milch, B., “Multi-Agent Influence Diagrams for Representing and Solving Games,” Proceedings of the 17th International Joint Conference of Artificial Intelligence, Morgan Kaufmann Publishers, Seattle, WA, Aug. 2001.

[5] – MathWorks documentation center about the hgtransform object.

http://www.mathworks.se/help/matlab/ref/hgtransform.html (Accessed: 6 June 2013)

[6] - Virtanen, K., Raivio, T., and Hämäläinen, R. P. “Modeling pilot's sequential maneuvering decisions by a multistage influence diagram.” Journal of Guidance, Control, and Dynamics 27, number 4, pages 665-677. © 2004 American Institute of Aeronautics and Astronautics (AIAA). 2004.

[7] – Technical specification for JAS-39 Gripen provided by the Ministry of defence and armed forces of the Czech Republic.

http://www.army.cz/scripts/detail.php?id=38154 (Accessed: 6 June 2013)

[8] - Virtanen, K., Hämäläinen, R. P., and Mattila, V., Team optimal signaling strategies in air combat. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, accepted for publication. © 2005 by authors and © 2005 IEEE.

[9] - Ben-Gal I., Bayesian Networks, in Ruggeri F., Faltin F. & Kenett R., Encyclopedia of Statistics in Quality & Reliability, Wiley & Sons (2007).

[10] – H. J. Pesch. Real time computation of feedback controls for constrained optimal control problems – Part 2: A correction method based on multiple shooting. Department

of Mathematics, University of Technology, P.O.Box 202420, D-8000 Munich 2, Federal Republic of Germany.

[11] - Virtanen, K., Ehtamo, H., Raivio, T., and Hämäläinen, R. P., 1999. VIATO—Visual Interactive Aircraft Trajectory Optimization.IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews 29, number 3, pages 409-421. © 1999 IEEE.

[12] – The International Standard Atmosphere: http://home.anadolu.edu.tr/~mcavcar/common/ISAweb.pdf (Accessed: 6 June 2013)

[13] – Shaw, R. L., Fighter Combat, Tactics and maneuvering, Naval Institute Press, Annapolis, MD 1985.

[14] - Ross, S., A First Course in Probability, 7th ed., Pearson Prentice–Hall,Upper Saddle River, NJ, 2006. Proposition 3.1 pp. 74.

[15] - Jensen, F., F. V. Jensen, S. L. Dittmer. From influence diagrams to junction trees. Department of Mathematics and Computer Science, Aalborg University. 1994.

[16] – Bertsekas, D. P., Dynamic Programming and Optimal Control, Athena Scientific, Belmont, MA, 2001.

[17] - Cruz, Jr., J. B., M. A. Simaan, A. Gacic, Y. Liu. Moving horizon Nash strategies for a military air operation. IEEE Transactions on Aerospace and Electronic Systems. Vol. 38, No. 3, 2002, pp. 978–999.

[18] - Lakoba, Taras I. Simple Euler method and its modifications. Technical report, University of Vermont. 2012.

http://www.cems.uvm.edu/~tlakoba/math337/notes_1.pdf (Accessed: 7 June 2013)

[19] - Ascher, Uri M.; Petzold, Linda R. Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, Society for Industrial and Applied Mathematics, Philadelphia. 1998.

Appendix

This is a numerical example of how to obtain the optimal control vector for a player at a certain stage k in the game. A 2-step look-ahead strategy is used. Consider two players, black player and white player, with state vectors:

$$\mathbf{x}_\mathbf{k}^\mathbf{B} = [\, x_k^B \; y_k^B \; h_k^B \; \gamma_k^B \; \chi_k^B \; v_k^B \,] = [0 \; 5000 \; 5000 \; 0 \; 0 \; 240],$$

$$\mathbf{x}_\mathbf{k}^\mathbf{W} = [\, x_k^W \; y_k^W \; h_k^W \; \gamma_k^W \; \chi_k^W \; v_k^W \,] = [2500 \; 5000 \; 5000 \; 0 \; 0 \; 240],$$

and control vectors:

$$\mathbf{u}_\mathbf{k}^\mathbf{B} = [\, \eta_k^B \;\; \mu_k^B \;\; n_k^B \,]^\mathrm{T} = [0.5 \; 0 \; 1]^\mathrm{T}, \; \mathbf{u}_\mathbf{k}^\mathbf{W} = [\, \eta_k^W \;\; \mu_k^W \;\; n_k^W \,]^\mathrm{T} = [0.5 \; 0 \; 1]^\mathrm{T}$$

The state vectors show that black player is pursuing white player.

Now the control vector for black player in stage k+1 is wanted. First the possible control vectors in stage k+1 are listed for black player, for simplicity let's say that the number of possible control vectors for black player in the next time step, $\#\mathbf{u}^\mathbf{B}$, is three in this case:

$$\mathbf{u}_{possible,\mathbf{k}+1}^\mathbf{B} = [\, \mathbf{u}_{1,\mathbf{k}+1}^\mathbf{B} \; \mathbf{u}_{2,\mathbf{k}+1}^\mathbf{B} \; \mathbf{u}_{3,\mathbf{k}+1}^\mathbf{B} \,] = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 \\ 2 & 1 & 0 \end{bmatrix}$$

So $\#\mathbf{u}^\mathbf{B} = 3$.

Since each player is fully aware of where its opponent is and which actions it might take the possible control vectors for the opponent can be listed. In this example, white player assumed to be the opponent, who is also assumed to only be able to use three new control vectors:

$$\mathbf{u}_{possible,\mathbf{k}+1}^\mathbf{W} = [\, \mathbf{u}_{1,\mathbf{k}+1}^\mathbf{W} \; \mathbf{u}_{2,\mathbf{k}+1}^\mathbf{W} \; \mathbf{u}_{3,\mathbf{k}+1}^\mathbf{W} \,] = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0 & 0 & 0 \\ 2 & 1 & 0 \end{bmatrix}$$

So $\#\mathbf{u}^\mathbf{W} = 3$.

This means there are three possible states, calculated by (A7), for each player in the next time stage.

$$x' = v \cos\gamma \cos\chi, \tag{A1}$$

$$y' = v \cos\gamma \sin\chi, \tag{A2}$$

$$h' = v \sin\gamma, \tag{A3}$$

$$\gamma' = \frac{g}{v}(n \cos\mu - \cos\gamma), \tag{A4}$$

$$\chi' = \frac{g}{v}(n \sin\mu / \cos\gamma), \tag{A5}$$

$$v' = \frac{1}{m}(\eta\, T_{max} - D(M(v,h))) - g \sin\gamma =$$

$$\frac{1}{m}\left(\eta\, T_{max} - \frac{1}{2} C_D v^2 S \varsigma(h)\right) - g \sin\gamma, \tag{A6}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t\, \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k) \tag{A7}$$

The function **f** consist of [x' y' h' γ' χ' v'] calculated by equations (A1)-(A6) respectively, Δt is equal to one.

$$\mathbf{x}^{\mathbf{B}}_{possible,\mathbf{k}+1} = [\,\mathbf{x}^{\mathbf{B}}\left(\mathbf{u}^{\mathbf{B}}_{1,\mathbf{k}+1}\right),\ \mathbf{x}^{\mathbf{B}}\left(\mathbf{u}^{\mathbf{B}}_{2,\mathbf{k}+1}\right),\ \mathbf{x}^{\mathbf{B}}\left(\mathbf{u}^{\mathbf{B}}_{3,\mathbf{k}+1}\right)] = \begin{bmatrix} 240 & 240 & 240 \\ 5000 & 5000 & 5000 \\ 5000 & 5000 & 5000 \\ 0.0408 & 0 & -0.0408 \\ 0 & 0 & 0 \\ 242.1 & 242.1 & 242.1 \end{bmatrix}$$

$$\mathbf{x}^{\mathbf{W}}_{possible,\mathbf{k}+1} = [\,\mathbf{x}^{\mathbf{W}}\left(\mathbf{u}^{\mathbf{W}}_{1,\mathbf{k}+1}\right), \mathbf{x}^{\mathbf{W}}\left(\mathbf{u}^{\mathbf{W}}_{2,\mathbf{k}+1}\right), \mathbf{x}^{\mathbf{W}}\left(\mathbf{u}^{\mathbf{W}}_{3,\mathbf{k}+1}\right)] = \begin{bmatrix} 2740 & 2740 & 2740 \\ 5000 & 5000 & 5000 \\ 5000 & 5000 & 5000 \\ 0.0408 & 0 & -0.0408 \\ 0 & 0 & 0 \\ 242.1 & 242.1 & 242.1 \end{bmatrix}$$

For each combination of these possible states of black and white player, a combat state vector is calculated, giving a total of $\#\mathbf{u}^{\mathbf{B}} * \#\mathbf{u}^{\mathbf{W}}$ possible combinations. In this case 3*3=9 combat state vectors for each player.

$$\omega_k^B = \arccos\left\{\frac{\left(x_k^W - x_k^B\right)\cos\gamma_k^B \cos\chi_k^B + \left(y_k^W - y_k^B\right)\cos\gamma_k^B \sin\chi_k^B + \left(h_k^W - h_k^B\right)\sin\gamma_k^B}{d_k^B}\right\} \qquad \text{(A8)}$$

$$\theta_k^B = \arccos\left\{\frac{\left(x_k^W - x_k^B\right)\cos\gamma_k^W \cos\chi_k^W + \left(y_k^W - y_k^B\right)\cos\gamma_k^W \sin\chi_k^W + \left(h_k^W - h_k^B\right)\sin\gamma_k^W}{d_k^B}\right\} \qquad \text{(A9)}$$

$$d_k^B = \sqrt{(x_k^W - x_k^B)^2 + (y_k^W - y_k^B)^2 + (h_k^W - h_k^B)^2} \qquad \text{(A10)}$$

$$\mathbf{c}_{\mathbf{k}+1}^{\mathbf{B}} = \begin{bmatrix} \mathbf{c}_{1,1}^{\mathbf{B}} & \mathbf{c}_{1,2}^{\mathbf{B}} & \mathbf{c}_{1,3}^{\mathbf{B}} \\ \mathbf{c}_{2,1}^{\mathbf{B}} & \mathbf{c}_{2,2}^{\mathbf{B}} & \mathbf{c}_{2,3}^{\mathbf{B}} \\ \mathbf{c}_{3,1}^{\mathbf{B}} & \mathbf{c}_{3,2}^{\mathbf{B}} & \mathbf{c}_{3,3}^{\mathbf{B}} \end{bmatrix}, \ \mathbf{c}_{\mathbf{k}+1}^{\mathbf{W}} = \begin{bmatrix} \mathbf{c}_{1,1}^{\mathbf{W}} & \mathbf{c}_{1,2}^{\mathbf{W}} & \mathbf{c}_{1,3}^{\mathbf{W}} \\ \mathbf{c}_{2,1}^{\mathbf{W}} & \mathbf{c}_{2,2}^{\mathbf{W}} & \mathbf{c}_{2,3}^{\mathbf{W}} \\ \mathbf{c}_{3,1}^{\mathbf{W}} & \mathbf{c}_{3,2}^{\mathbf{W}} & \mathbf{c}_{3,3}^{\mathbf{W}} \end{bmatrix}$$

Each combat vector $\mathbf{c}_{\mathbf{n},\mathbf{m}}^{\mathbf{i}}$ (n stands for the player's control vector and m for the opponent's control vector, i.e. $\mathbf{c}_{1,3}^{\mathbf{B}}$ is the combat state vector obtained from $\mathbf{u}_1^{\mathbf{B}}$ and $\mathbf{u}_3^{\mathbf{W}}$) consists of the bearing angle ω, the angle-off θ, and the distance *d* and they are calculated according to (A8)-(A10).

Since this matrix of combat vectors is a tensor (dimensions #$\mathbf{u}^{\mathbf{B}}$*#$\mathbf{u}^{\mathbf{W}}$*3, 3 since it is the length of $\mathbf{c}_{\mathbf{n},\mathbf{m}}^{\mathbf{i}}$) it would be too messy to give all numerical values here, but for an example are the values of $\mathbf{c}_{1,1}^{\mathbf{B}}$ and $\mathbf{c}_{1,1}^{\mathbf{W}}$ shown below and the rest are left for the interested reader:

$$\mathbf{c}_{1,1}^{\mathbf{B}} = [\,\omega_{1,1}^{\mathbf{B}}, \theta_{11}^{B},\ d_{11}^{B}] = [0.0408,\ 0.0408,\ 2500]$$

$$\mathbf{c}_{1,1}^{\mathbf{W}} = [\,\omega_{1,1}^{\mathbf{B}}, \theta_{11}^{B},\ d_{11}^{B}] = [3.1008,\ 3.1008,\ 2500]$$

Given a combat state vector we can calculate probability and utility values to see how "good" it is. The probabilities consist of a four value vector which says with what probability the player is in any of the four possible states of the game (shown in Table 2). The vector is shown in (A11) and how to calculate each element is shown in (A12).

$$\mathbf{p}_{\mathbf{k}}^{\mathbf{i}}(\mathbf{c}_{\mathbf{k}}^{\mathbf{i}}) = [\mathrm{P}(\Theta_k^i = 1 \mid \mathbf{C} = \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \mathrm{P}(\Theta_k^i = 2 \mid \mathbf{C} = \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \mathrm{P}(\Theta_k^i = 3 \mid \mathbf{C} = \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \mathrm{P}(\Theta_k^i = 4 \mid \mathbf{C} = \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})]^{\mathrm{T}} \qquad \text{(A11)}$$

$$\mathrm{P}(\Theta_{k+1}^{i} = \mathrm{j} \mid \mathbf{C} = \mathbf{c}_{k+1}{}^{\mathrm{i}}) = \frac{\mathrm{P}\left(\Theta_{\mathrm{k}+1}^{\mathrm{i}} = j\right)\mathrm{P}\left(\mathbf{c}_{\mathbf{k}+1}^{\mathbf{i}} \mid \Theta_{k+1}^{i} = j\right)}{\sum_{n=1}^{4}\mathrm{P}\left(\Theta_{\mathrm{k}+1}^{\mathrm{i}} = n\right)\mathrm{P}\left(\mathbf{c}_{\mathbf{k}+1}^{\mathbf{i}} \mid \Theta_{k+1}^{i} = n\right)} =$$

$$\frac{\mathrm{P}\left(\Theta_{\mathrm{k}}^{\mathrm{i}} = j \mid C = \mathbf{c}_{\mathbf{k}}^{\mathbf{i}}\right)\mathrm{P}\left(\mathbf{c}_{\mathbf{k}+1}^{\mathbf{i}} \mid \Theta_{k+1}^{i} = j\right)}{\sum_{n=1}^{4}\mathrm{P}\left(\Theta_{\mathrm{k}}^{\mathrm{i}} = n \mid C = \mathbf{c}_{\mathbf{k}}^{\mathbf{i}}\right)\mathrm{P}\left(\mathbf{c}_{\mathbf{k}+1}^{\mathbf{i}} \mid \Theta_{k+1}^{i} = n\right)} \tag{A12}$$

where $\mathrm{P}(\mathbf{c}_{\mathbf{k}}^{\mathbf{i}} \mid \Theta_{k}^{i} = \mathrm{j}) = \mathrm{p}^{\omega,\mathrm{i}}(\omega_{k}^{i} \mid \Theta_{k}^{i} = \mathrm{j})\mathrm{p}^{\theta,\mathrm{i}}(\theta_{k}^{i} \mid \Theta_{k}^{i} = \mathrm{j})\mathrm{p}^{\mathrm{d},\mathrm{i}}(d_{k}^{i} \mid \Theta_{k}^{i} = \mathrm{j})$. The utility for each combat state vector also consist of a four value vector where each value is a utility according to which threat assessment situation the player is assumed to be in. The vector is shown in (A13) and each element is calculated according to (A14).

$$\mathbf{U}_{\mathbf{k}}^{\mathbf{i}}(\mathbf{c}_{\mathbf{k}}^{\mathbf{i}}) = [\mathrm{U}^{\mathrm{i}}(1, \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \ \mathrm{U}^{\mathrm{i}}(2, \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \ \mathrm{U}^{\mathrm{i}}(3, \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})\ \ \mathrm{U}^{\mathrm{i}}(4, \mathbf{c}_{\mathbf{k}}^{\mathbf{i}})]^{\mathrm{T}}, \tag{A13}$$

$$\mathrm{U}^{\mathrm{i}}(\mathrm{j}, \mathbf{c}_{\mathbf{k}}^{\mathbf{i}}) = w_{j}^{\omega,i}\, u_{j}^{\omega,i}(\omega_{k}^{i}) + w_{j}^{\theta,i}\, u_{j}^{\theta,i}(\theta_{k}^{i}) + w_{j}^{d,i}\, u_{j}^{d,i}(d_{k}^{i}),\ \mathrm{j}=1..4 \tag{A14}$$

After calculating probabilities and utilities for each combat state vector, two tensors are obtained (dimensions $\#\mathbf{u}^{\mathbf{B}} * \#\mathbf{u}^{\mathbf{W}} * 4$, 4 since it is the length of both $\mathbf{p}_{\mathbf{k}}^{\mathbf{i}}$ and $\mathbf{U}_{\mathbf{k}}^{\mathbf{i}}$) for each player:

$$\textbf{BProbabilities} = \begin{bmatrix} \mathrm{P}_{1,1}^{\mathrm{B}} & \mathrm{P}_{1,2}^{\mathrm{B}} & \mathrm{P}_{1,3}^{\mathrm{B}} \\ \mathrm{P}_{2,1}^{\mathrm{B}} & \mathrm{P}_{2,2}^{\mathrm{B}} & \mathrm{P}_{2,3}^{\mathrm{B}} \\ \mathrm{P}_{3,1}^{\mathrm{B}} & \mathrm{P}_{3,2}^{\mathrm{B}} & \mathrm{P}_{3,3}^{\mathrm{B}} \end{bmatrix}, \textbf{WProbabilities} = \begin{bmatrix} \mathrm{P}_{1,1}^{\mathrm{W}} & \mathrm{P}_{1,2}^{\mathrm{W}} & \mathrm{P}_{1,3}^{\mathrm{W}} \\ \mathrm{P}_{2,1}^{\mathrm{W}} & \mathrm{P}_{2,2}^{\mathrm{W}} & \mathrm{P}_{2,3}^{\mathrm{W}} \\ \mathrm{P}_{3,1}^{\mathrm{W}} & \mathrm{P}_{3,2}^{\mathrm{W}} & \mathrm{P}_{3,3}^{\mathrm{W}} \end{bmatrix}$$

$$\textbf{BUtilites} = \begin{bmatrix} \mathrm{U}_{1,1}^{\mathrm{B}} & \mathrm{U}_{1,2}^{\mathrm{B}} & \mathrm{U}_{1,3}^{\mathrm{B}} \\ \mathrm{U}_{2,1}^{\mathrm{B}} & \mathrm{U}_{2,2}^{\mathrm{B}} & \mathrm{U}_{2,3}^{\mathrm{B}} \\ \mathrm{U}_{3,1}^{\mathrm{B}} & \mathrm{U}_{3,2}^{\mathrm{B}} & \mathrm{U}_{3,3}^{\mathrm{B}} \end{bmatrix}, \textbf{WUtilites} = \begin{bmatrix} \mathrm{U}_{1,1}^{\mathrm{W}} & \mathrm{U}_{1,2}^{\mathrm{W}} & \mathrm{U}_{1,3}^{\mathrm{W}} \\ \mathrm{U}_{2,1}^{\mathrm{W}} & \mathrm{U}_{2,2}^{\mathrm{W}} & \mathrm{U}_{2,3}^{\mathrm{W}} \\ \mathrm{U}_{3,1}^{\mathrm{W}} & \mathrm{U}_{3,2}^{\mathrm{W}} & \mathrm{U}_{3,3}^{\mathrm{W}} \end{bmatrix}$$

Once again it gets too messy go give all numerical values here. But the values of $\mathrm{P}_{1,1}^{\mathrm{B}}$ and $\mathrm{U}_{1,1}^{\mathrm{B}}$ are shown below and the rest are left for the interested reader.

$$\mathrm{P}_{1,1}^{\mathrm{B}} = [0.2453, 0.2714, 0.2322, 0.2510]^{\mathrm{T}}$$

$$\mathrm{U}_{1,1}^{\mathrm{B}} = [0.8503, 0.8503, 0.7534, 0.2471]^{\mathrm{T}}$$

Now the payoffs for each of these combat vectors can be calculated by (A15) and all payoffs for each state (each control vector) of the player and opponent are summed together giving one total payoff value for each control vector.

$$\mathbf{J}^{\mathbf{i}}_{\mathbf{k},\mathbf{k}+1}\left(\mathbf{u}^{\mathbf{B}}_{\mathbf{k}},\mathbf{u}^{\mathbf{W}}_{\mathbf{k}}\right)=\left[\mathbf{p}^{\mathbf{i}}_{\mathbf{k}+1}\left(\mathbf{c}^{\mathbf{i}}_{\mathbf{k}+1}\right)\right]^{T}\mathbf{U}^{\mathbf{i}}_{\mathbf{k}+1}\left(\mathbf{c}^{\mathbf{i}}_{\mathbf{k}+1}\right) \tag{A15}$$

$$\mathbf{BPayoff}=\begin{bmatrix} J^{B}_{1,1} & J^{B}_{1,2} & J^{B}_{1,3} \\ J^{B}_{2,1} & J^{B}_{2,2} & J^{B}_{2,3} \\ J^{B}_{3,1} & J^{B}_{3,2} & J^{B}_{3,3} \end{bmatrix}=\begin{bmatrix} 0.6764 & 0.6793 & 0.6764 \\ 0.6773 & 0.6802 & 0.6773 \\ 0.6764 & 0.6793 & 0.6764 \end{bmatrix}$$

$$\mathbf{WPayoff}=\begin{bmatrix} J^{W}_{1,1} & J^{W}_{1,2} & J^{W}_{1,3} \\ J^{W}_{2,1} & J^{W}_{2,2} & J^{W}_{2,3} \\ J^{W}_{3,1} & J^{W}_{3,2} & J^{W}_{3,3} \end{bmatrix}=\begin{bmatrix} 0.3724 & 0.3691 & 0.3724 \\ 0.3715 & 0.3682 & 0.3715 \\ 0.3724 & 0.3691 & 0.3824 \end{bmatrix}$$

$$\mathbf{BTotalPayoff}=\begin{bmatrix} J^{B}_{1,1}+J^{B}_{1,2}+J^{B}_{1,3} \\ J^{B}_{2,1}+J^{B}_{2,2}+J^{B}_{2,3} \\ J^{B}_{3,1}+J^{B}_{3,2}+J^{B}_{3,3} \end{bmatrix}=\begin{bmatrix} 2.0320 \\ 2.0348 \\ 2.0320 \end{bmatrix}$$

$$\mathbf{WTotalPayoff}=\begin{bmatrix} J^{W}_{1,1}+J^{W}_{1,2}+J^{W}_{1,3} \\ J^{W}_{2,1}+J^{W}_{2,2}+J^{W}_{2,3} \\ J^{W}_{3,1}+J^{W}_{3,2}+J^{W}_{3,3} \end{bmatrix}=\begin{bmatrix} 1.1140 \\ 1.1111 \\ 1.1140 \end{bmatrix}$$

Now there is a total payoff value associated with each possible control vector for both players. One thing to observe is that two of the control vectors for both players give the same payoff value.

Since a 2-step look-ahead strategy is used, there now is a recursive call with every possible control vector and corresponding new state for black player. The optimal control vector for white player, i.e. the control vector with index equal the index of max(WPayTot), is used in the recursive call together with its corresponding new state. Since there are two control vectors corresponding to max(WPayTot), the vector with lowest index is chosen, i.e. $\mathbf{u}^{\mathbf{W}}_{1}$.

The process is then repeated from the beginning three times, one for every possible control vector of black player. Each recursive call returns the optimal control vector for black player in the next step (k+2) and its corresponding payoff value. These three payoff values are then added to the corresponding value in **BPayTot**. Then the control vector with highest payoff value is chosen for implementation.

The algorithm is described in pseudo code below.

Pseudo code

```
function[controlVectors, utility]=GetHighestUtilityVectors(NB,BS,WS,BC,WC,BP,WP)
    // NB = number of look-ahead steps
    // BS/WS = current state of black/white player
    // BC/WC = current control vector for black/white player
    // BP/WP = current probability distribution for black/white player
    if(NB = 0)
        controlVectors = [];
        utility = 0;
    else
        BC_next = ProducePossibleControlVectors(BC);
        WC_next = ProducePossibleControlVectors(WC);

        BS_next = ProducePossibleStates(BC_next,BS);                (A7)
        WS_next = ProducePossibleStates(WC_next,WS);                (A7)

        BCo = ProduceCombatVectors(BS_next, WS_next);               (A8)-(A10)
        WCo = ProduceCombatVectors (WS_next,BS_next);               (A8)-(A10)

        BPayOff = sum(Probabilities(BCo,BP)*Utilities(BCo));        (A14)
        WPayOff = sum(Probabilities(WCo,WP)*Utilities(WCo));        (A14)

        WMax = max(WPayOff);

        for i=1 to length(BC_next) do
            if(ViolatesContraints(BS_next(i))
                BPayOff(i) -= 1000;
            else
                [C,U]=GetHighestUtilityVectors(NB-1,BS_next(i),
                WS_next(WMax), BC_next(i), WC_next(WMax),
                Probabilities(BCo,BP)(i), Probabilities(WCo,WP)(WMax))
                BPayOff(i) += U;
        controlVectors = BC_next(BPayOff == max(BPayOff));
```