DISSERTATION
submitted
to the
Combined Faculty for the Natural Sciences and Mathematics
of
Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by

Diplom-Informatiker:	Alexander Alexopoulos
Born in:	Heidelberg
Oral examination:	.....

Multi-Agent Pursuit of a Faster Evader with Application to Unmanned Aerial Vehicles

Advisor: Prof. Dr. sc. techn. Essameddin Badreddin

Για τους γονείς μου.

For my parents.

Abstract

Robotic applications like search and rescue missions, surveillance, police missions, patrolling, and warfare can all be modeled as a Pursuit-Evasion Game (PEG). Most of these tasks are multi-agent problems, often including a cooperation between team members and a conflict between adversarial teams. In order to realize such a situation with robots, two major problems have to be solved. Initially, a decomposition of the PEG has to be performed for getting results in reasonable time. Present embedded computers lack the computational power enabling them to process the highly complex solution algorithm of the non-decomposed game fast enough. Secondly, a framework has to be defined, enabling the computation of optimal actions for both the pursuers and the evaders when a cooperation within the teams is possible. It is intended to develop strategies, that allow the team of pursuers to capture a faster evader in a visibility-based PEG setup due to cooperation.

For tackling the first problem a game structure is sought, aiming to considerably reduce the time complexity of the solution process. The first step is the decomposition of the action space, and the second is the change of the game structure itself. The latter is reached by defining a two-pursuer one-evader PEG with three different game structures, which are the Non-Decomposed Game, the Multiple Two-Player Game Decomposition (MTPGD) game, and the Team-Subsumption Two-Player Game (TSTPG). Several simulation results demonstrate, that both methods yield close results in respect to the full game. With increasing cardinality of each player's strategy space, the MTPGD yields a relevant decrease of the run-time. Otherwise, the TSTPG does not minimize the time complexity, but enables the use of more sophisticated algorithms for two-player games, resulting in a decreased runtime.

The cooperation within a team is enabled by introducing a hierarchical decomposition of the game. On a superordinate collaboration level, the pursuers choose their optimal behavioral strategy (e.g. pursuit and battue) resulting in the case of a two-pursuer one-evader PEG in a three-player non-cooperative dynamic game, which is solved in a subordinate level of the overall game. This structure enables an intelligent behavior change for the

pursuers based on game-theoretical solution methods. Depending on the state of the game, which behavioral strategy yields the best results for the pursuers within a predefined time horizon has to be evaluated. It is shown that the pursuer's outcome can be improved by using a superordinate cooperation. Moreover, conditions are presented under which a capture of a faster evader by a group of two pursuers is possible in a visibility-based PEG with imperfect information.

Since Unmanned Aerial Vehicles (UAVs) are increasingly a common platform used in the aforementioned applications, this work focuses only on PEGs with multi-rotor UAVs. Furthermore, the realization of the concepts in this thesis are applied on a real hex rotor. The feasibility of the approach is proven in experiments, while all implementations on the UAV are running in real-time.

This framework provides a solution concept for all types of dynamic games with an 1-M or N-1 setup, that have a non-cooperative and cooperative nature. At this stage a N-M dynamic game is not applicable. Nevertheless, an approach to extend this framework to the N-M case is proposed in the last chapter of this work.

Zusammenfassung

Robotikanwendungen wie Such- und Rettungseinsätze, Überwachung, Polizeieinsätze, Patrouille, und Kriegsführung können alle als PEG modelliert werden. Die meisten dieser Aufgabenbereiche sind Multi-Agenten-Probleme, welche oftmals sowohl eine Kooperation zwischen Teammitgliedern, als auch eine Konfliktsituation mit einem gegnerischen Team beinhalten. Um eine solche Situation mit echten Robotern realisieren zu können, müssen zwei wesentliche Probleme gelöst werden. Zuerst muss eine Zerlegung des PEG durchgeführt werden, um Ergebnisse in akzeptabler Zeit bekommen zu können. Die heutigen Embedded Computer wiesen nicht die nötige Rechenleistung auf, um den höchst komplexen Lösungsalgorithmus des nichtzerlegten Spiels schnell bearbeiten zu können. Zweitens muss ein Framework definiert werden, das die Berechnung von optimalen Aktionen sowohl für die Verfolger als auch für die Verfolgten ermöglicht, wobei eine Kooperation innerhalb der Teams möglich ist. Es ist beabsichtigt, Strategien zu entwickeln, mit denen es einem Team von Verfolgern ermöglicht wird, einen schnelleren Verfolgten in sichtbasierten PEGs durch Kooperation einzufangen.

Um das erste Problem anzugehen, werden Spielstrukturen gesucht, die die Zeitkomplexität des Lösungsprozesses erheblich reduzieren. Der erste Schritt ist die Zerlegung des Aktionsraumes, der zweite die Veränderung der Spielstruktur selbst. Für letzteres wird ein zwei-Verfolger-ein-Verfolgter PEG in drei verschiedenen Spielstrukturen definiert: das nichtzerlegte Spiel, das MTPGD Spiel und das TSTPG. Durch eine Simulationsreihe wird gezeigt, dass beide Methoden Lösungen nahe den Lösungen des nichtzerlegten Spiels liefern. Mit steigender Kardinalität des Strategieraums der Spieler, liefert die MTPGD eine erhebliche Minderung der Laufzeit. Außerdem wird durch die TSTPG die Zeitkomplexität zwar nicht vermindert, jedoch können nun ausgeklügelte Algorithmen für Zwei-Spieler-Spiele benutzt werden, die wiederum die Laufzeit erheblich reduzieren.

Die Kooperation innerhalb eines Teams wird durch die Einführung einer hierarchischen Zerlegung des Spiels ermöglicht. Auf einer übergeordneten Kollaborationsstufe können Verfolger ihre optimalen Verhaltensstrategien

wählen (z.B. Verfolgung und Treibjagd), die im Falle eines zwei-Verfolger-ein-Verfolgter PEGs zu einem Drei-Spieler-, nicht-kooperativen dynamischen Spiel führen, das auf der darunterliegenden Stufe des Spiels gelöst wird. Diese Struktur ermöglicht den Verfolgern eine intelligente, auf Spieltheorie basierende Verhaltensumschaltung. Abhängig vom aktuellen Spielzustand kann evaluiert werden, welche Verhaltensstrategiekombination das beste Resultat über einen vordefinierten Zeithorizont für die Verfolger erzielt. Es wird gezeigt, dass das Ergebnis der Verfolger durch eine übergeordnete Kooperation verbessert werden kann. Außerdem wird gezeigt unter welchen Bedingungen der Fang eines schnelleren Verfolgten durch eine Gruppe von zwei Verfolgern in sichtbasierten PEGs mit imperfekter Information möglich ist.

Da unbemannte Luftfahrzeuge heutzutage häufig als Platform für die oben genannten Anwendungen eingesetzt werden, wird in dieser Arbeit der Fokus auf PEG mit mehrrotorigen UAVs gelegt. Außerdem werden alle Konzepte, welche in dieser Arbeit entwickelt werden, auf einem echten Hexakopter realisiert. In Experimenten wird die Umsetzbarkeit der Methoden bewiesen, wobei alle Implementierungen auf dem UAV in Echtzeit laufen.

Das Framework bietet ein Lösungskonzept für jegliche Art von dynamischen Spielen mit einem 1-M oder N-1 Setup, die einen nicht-kooperativen und einen kooperativen Anteil haben. In diesem Stadium ist ein dynamisches N-M Spiel nicht durchführbar. Nichtsdestotrotz wird im letzten Kapitel dieser Arbeit ein Lösungsvorschlag beschrieben, wie das beschriebene Framework auf den N-M Fall erweitert werden kann.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr. sc. techn. Essameddin Badreddin for the continuous support, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I am especially grateful to M.Sc. Tobias Schmidt, M.Sc. Benjamin Kirsch and M.Sc. Tim Cottin for the support in overcoming numerous obstacles I have been facing through my research.

I would like to thank all my fellow workers at the Automation Laboratory of the Institute for Computer Engineering (ZITI) of the Heidelberg University for their feedback, cooperation and of course friendship.

Last but not the least, I would like to thank my friends and my family: my parents, my fiancée and my sisters for supporting me throughout writing this thesis and my life in general.

Contents

List of Figures

List of Tables

Nomenclature

Variables

$\boldsymbol{\Upsilon}$	Multi-Rotor Input Space
$\boldsymbol{\upsilon}$	Multi-Rotor Inputs
Δt	Time Step
Γ	Strategy Space of the Game
Γ_i^k	Set of Strategies of $\mathbf{p}_i$ in Stage k
γ_i^k	Strategy of $\mathbf{p}_i$ in Stage k
Γ_i	Strategy Space of $\mathbf{p}_i$
$\hat{X}$	Observation Set of the Game
$\hat{X}_i^k$	Observation Set of $\mathbf{p}_i$ in Stage k
$\hat{\mathbf{x}}_i^k$	Observation of $\mathbf{p}_i$ in Stage k
ι	State Information of the Game
κ_i^k	State-Observation Equation of $\mathbf{p}_i$ in Stage k
$\mathbf{b}$	Behavioral Strategy
$\mathbf{E}$	Evading Team
$\mathbf{K}$	Set of Stages
$\mathbf{P}$	Pursuing Team
$\mathbf{p}_j$	Pursuing Player j
Φ	Set of Roll Angles
$\Phi \times \Theta \times \Psi$	Attitude Space
Π	N-team discrete-time deterministic dynamic game with non-fix terminal time
π_i^k	Attribute Information of $\mathbf{p}_i$ in Stage k
Ψ	Set of Yaw Angles
τ	Set of all Players of the Game
Θ	Set of Pitch Angles

$\mathcal{X}$	State Space of the Game
Ξ	Terminal Set of the Game
${}^{\tau}\iota_{\mathbf{J}}^{k}$	State Information of Team **J** in Stage k
${}^{\tau}\pi_{\mathbf{J}}^{k}$	Attribute Information of Team **J** in Stage k
d_{ϵ}	Capture Distance
f^{k}	Difference Equation of the Game
g	Gravitation Constant
g^{k}	Cost of $\mathbf{p}_i$ in stage k
I_r	Rotor Moment of Inertia
I_x	Inertia around x-axis
I_y	Inertia around y-axis
I_z	Inertia around z-axis
J	Cost Functional with Mixed Strategies
K	Terminal Stage of the Game
k	Stage
L	Cost Functional of the game
l	Cantilever Length
$P \times Q \times R$	Angular Velocity Space
P	Angular Velocity around x-Axis
Q	Angular Velocity around y-Axis
R	Angular Velocity around z-Axis
r_S	Sensing Range
U	Action Set of the Game
U_i^k	Action Space of $\mathbf{p}_i$ in Stage k
V	Value Function
$V_x \times V_y \times V_z$	Linear Velocity Space
V_x	Linear Velocity Set in x-direction
V_y	Linear Velocity Set in y-direction
V_z	Linear Velocity Set in z-direction
$X \times Y \times Z$	Position Space
X	Position Set of the x-Axis
Y	Position Set of the y-Axis
Z	Position Set of the z-Axis
$\mathbf{e}_o$	Evading Player o
$\mathbf{q}^{k*}$	Optimal Probability Distribution over the Action Space
$\mathbf{u}_i^k$	Actions of $\mathbf{p}_i$ in Stage k

$\mathbf{x}^k$	State of the Game in Stage k

Abbreviations

LoS	Line of Sight
MTPGD	Multiple Two-Player Game Decomposition
PEG	Pursuit-Evasion Games
PMA	Performance Map Approach
RNBC	Recursive Nested Behavioral Control
TSTPG	Team-Subsumption Two-Player Game
UAVs	Unmanned Aerial Vehicles
VTOL	Vertical Take-Off and Landing

1 Introduction

1.1 Motivation

Game theory is the study of non-cooperative and cooperative decision situations. It is a framework for the modeling of situations where two or more players are faced with choices of action. These choices affect the outcome of every player. With other words, the final outcome of a game depends on the strategies chosen by all participants. Hence, game theory is the generalization of single-agent optimization problems, like optimal control or linear programming. A wide class of multi-player games are called Pursuit-Evasion Games (PEGs). PEGs describe a class of optimization problems, in which an agent or a team of agents aim to catch an adversarial agent or team of agents, while the latter try to avoid capture. In general, capture is equal to the fulfillment of one or more conditions, which depend on the specific problem being considered. Ever since carnivorous creatures have existed, pursuit-evasion games have taken place in the natural world. The predator hunting the prey, or in other words the pursuer chasing the evader. Especially for predators hunting in packs, e.g. lions, intelligent hunting behaviors have been observed. Regarding a lion versus a gazelle scenario, the possibility for the lion to capture the gazelle through an ordinary pursuit (one on one) is not as big as one might assume. Although the lion has a higher maximal speed, his limited stamina demands that capture occurs in the first few seconds of the chase. Especially for feral feline predators, intelligent pursuing strategies are essential for their survival. In order to maximize their chances of getting a meal, lions often try to ambush their prey by applying a battue; one or more lions hiding in the bushes, while another lion drives

the prey towards this direction. Once the prey spots the hidden lions, it is often too late to escape.

Some conceivable applications for PEGs are search and rescue missions [CHI11], police missions (search and capture) [CHI11; Mor+05], patrolling [AGI08], surveillance [BBH14b; ZL11], and warfare [SS95; Pon11; TS03]. As depicted on Figure 1.1, each application incorporates several topics. Although capture-and-escape problems are a sub-class of PEGs, battue games [She02] have fuzzy borders to PEGs. Search games are a distinct class of problems and are not regarded in this work. The visibility-based PEG are often treated in literature as a combination of search and PEG [GTG06; SO14a; SO14b]. One or more pursuers, having a limited visibility, try to find one or more evaders, and depending on the problem formulation, try to catch them. Regarding most applications in which PEGs could be utilized, a search of the targets must be performed at first, e.g. cops and robbers games or search and capture games. This work focuses on the search-free game classes, while regarding mostly the class of multiple-pursuer single-evader PEGs providing a game-theoretical formulation for such problems. Game theory is used to solve these type of games, as it represents the most general framework for multi-player optimization problems. Game theory is characterized by the following three properties:

- The solution of the problem depends on the actions of all involved players.
- Everybody considers the actions of all involved players.
- Everybody is acting rationally.

Generally, such multi-player games are problems with more than one involved player or party. The solution process of such problems is often very complex and depends on the number of players and the number of available strategies for each player. One may distinguish between non-cooperative and cooperative games. In non-cooperative games or conflict games, all involved parties have differing goals, meaning that every party wants to maximize its outcome and at the same time maximize the losses of the adversarial agents. In cooperative games usually all involved agents have the same overall goal and try to find optimal strategies that maximize the outcome of the team.

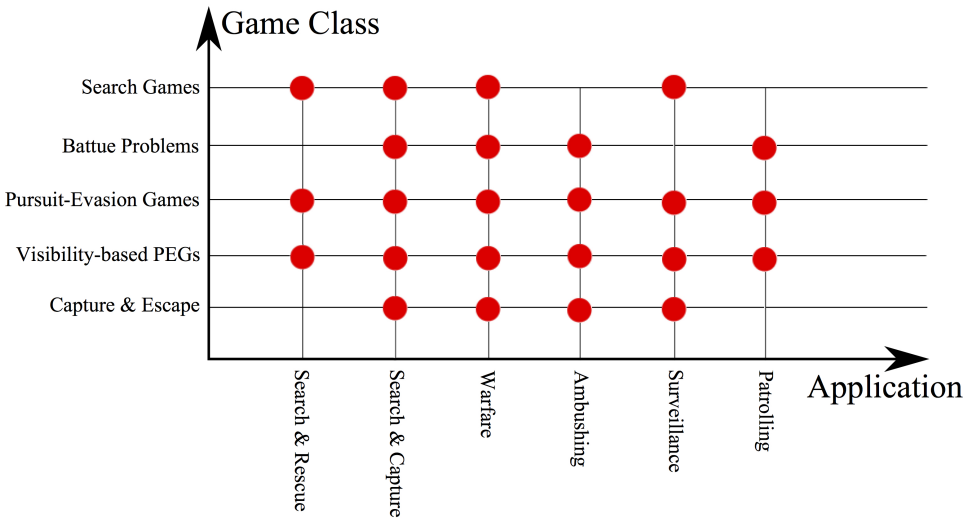

Figure 1.1: *Classification of pursuit-evasion games.*

No matter which applications are modeled as PEG, the essence of the following questions can be projected to any of them. *"Is the pursuer able to catch the evader?"* or in other words *"Does the PEG converge to a solution?"*

This thesis proposes a framework enabling the computation of such hybrid game-theoretical problems. The multi-player PEG is divided into two levels. The lower level describes the non-cooperative part of the game and the upper level describes the cooperative part of the game. In the upper level, a team has the opportunity to negotiate about the ideal role for each team-mate to maximize the outcome. In the lower layer, this behavior is applied against the opponents in the non-cooperative game. Regarding the PEG example with the hunting lions, a hard-running and more maneuverable animal like a gazelle can be caught much more easily through cooperation and outnumbering. This approach is formulated in this work for a team of autonomous pursuers trying to catch a faster and more maneuverable autonomous evader. In chapter 5, an approach is proposed imitating this behavior.

As stated above, one disadvantage of a game-theoretical solution process lies in the computational complexity. Regarding the utilization of PEGs in real applications, the following question arises: *"Can I obtain a solution within a reasonable time period?"* With respect to the above mentioned applications, it is more than likely that a team of pursuers and a team of evaders are opposing each other. Utilization demands a short reaction time and thus

the answer to this questions will mostly be “no”. This work emphasizes the reduction of time complexity by means of game decompositions for which computationally efficient algorithms exist.

Due to the recent development status of embedded computers and batteries, more and more battery-driven unmanned aerial vehicles (UAVs) are used for the aforementioned applications. Especially rotary-wing or vertical take-off and landing (VTOL) UAVs are a common platform in these applications. Even though the mission time of such UAV agents is still limited, the trend in battery research points to a rapid increase of the energy density of batteries, making the usage of battery-driven UAVs even more attractive. This work focuses on the application of UAVs in PEGs. In particular, the realization of a hex rotor test platform will be described, which is used to show the feasibility of the presented approaches in this work by experiments.

1.2 State of the Art

The very first step for a general definition of game theory were made by Emile Borel in 1921 [Bor21], when he proved a minimax theorem for two-player zero-sum games with symmetric pay-off matrix. He presumed, that there are cases in two-player zero-sum games where a mixed strategy is non-existent. The work of John von Neumann published in 1928 [Neu28] demonstrated that Borel’s presumption was wrong. Von Neumann proved that each two-player zero-sum game has a solution in mixed-strategies. This work and his book published with Oskar Morgenstern in 1944 [NM07], which focused on game theory and its applications to economics including concepts for co-operative games, initiated the birth of modern game theory. John Nash was the one to prove the existence of an equilibrium for non-cooperative nonzero-sum n-player games [Nas50a]. Even today, the Nash equilibrium is still the most common method for solving non-cooperative games. John Nash also published [Nas50b] and [Nas53], where he proved the existence of the Nash bargaining solution for cooperative games. In addition to von Neumann and Nash is Shapley, who also contributed to solution methods for cooperative games, such as the Core [Sha52] and the Shapley Value [Sha53]. Another key figure in this space is Rufus Isaacs, the father of differential games, was a member of the RAND Corporation (Research and Development) where he

conducted his studies in this topic. RAND was founded to act as a "think tank" financed, among others, by the U.S. government, which loomed large especially in the cold war.

One of the first, if not the first, formulated and most famous PEGs is the *lion-and-man* problem, which originated in the 1920s when R. Rado posed the following question: *"A lion and a man are both inside a circular arena, each running at the same constant speed. Can the lion catch the man?"* [Nah12]. However, Rufus Isaacs was the first to mathematically formulate PEGs in [Isa51] and his book "Differential Games" [Isa65]. He addressed PEG problems of the two-player zero-sum type, while the kinematics of the players are described by differential equations. He defined a framework for solving such games, by solving partial differential equations, later called the Hamilton-Jacobi-Isaacs equation, and also known as the Hamilton-Jacobi-Bellman-Issacs equation, because of the affinity with dynamic programming [Bel57] introduced by Bellman at about the same time (1957). Furthermore, Isaacs introduced discrete differential games, providing a dynamic programming like solution. Many famous two-player PEG were introduced in Isaacs book. For example, the "homicidal chauffeur game", which is, alongside "the game of two identical cars", one of the most investigated differential games. Extensive research has been done on both problems, resulting in many relevant variations [Mer72; PT00; Mit01; PT11]. Later, differential games were generalized to N-Player games, while dynamic programming for N-player games was introduced as a solution method in [BO99] yielding a Nash equilibrium, when a solution is existent. As, Bellman's dynamic programming was introduced to be applied on discrete-time optimal control problems, and as game theory is the generalization of optimal control, [BO99] introduced a framework to solve N-player discrete-time dynamic games with dynamic programming. Dynamic programming suffers from the "curse of dimensionality". This means that the computation of an optimization problem, like a dynamic game, becomes very complex, as the time-complexity is increasing exponentially with the number of players, the number of strategies and the number of state variables.

In recent years, the multi-player (mostly 1-N or N-1 games) PEG have been investigated more and more, especially for the utilization in robotic applications. Two problems arise when regarding such games. On the one hand, the

cooperation between team-mates and on the other hand the increase in time-complexity. While in literature some solution methods for 1-N or N-1 PEGs can be found [KR05; BBH07; Ls10; LQT12; Liu+13; WF13], these methods either neglect an explicit cooperation within teams or provide a limited cooperation capability. Another method that was utilized in [CSG09], [RT07], and [Pan+12] were two solution approaches for n-pursuer single-evader PEGs enabling a dynamic role assignment, e.g. pursue or contain target. In both approaches, the player's role assignment depends on specific conditions on the player's states. In terms of game theory, a negotiation and agreement between team-mates on which role to be assigned has to take place, which is not the case.

When it comes to multi-pursuer multi-evader PEGs, little work has been done mainly due to the time-complexity of present solution algorithms. [LU06] and [LCS08] for instance tackled this problem by trying to extend the differential game theory for multi-pursuer multi-evader PEGs. A suboptimal solution was iteratively improved based on limited look-ahead methods that approach an optimal solution. Nevertheless, the authors stated that no practical algorithm exists for such a problem. Another work dealing with this problem is [Wei+07]. Decomposition of the game is used to reduce the time-complexity of the solution process. However, as in the above mentioned literature tackling 1-N and N-1 games, no explicit cooperation takes place regarding the assignment of a pursuer to an evader.

While currently the computation of two-player dynamic games with a reasonable set of strategies and states is utilizable for most real-time applications, the time-complexity of the solution method for multi-player games is not feasible. There are two ways to tackle this problem. The first one is to determine the open-loop solution of the game, as presented in [Liu+13]. An open-loop formulation of a single-pursuer multiple-evader PEG was presented there. The open-loop approach was chosen here, to avoid the computational burden, given by the Hamilton-Jacobi-Isaacs equation, giving a closed-loop solution. Within an actual application, the open-loop method would fail. Since in open-loop games the opponents play cannot be observed, even a slight disturbance or sensing errors lead to a non-predictable deviation of the opponents state. The second way is to decompose the game into multiple, easier to handle, two-player games, or even into multiple optimal control

problems as proposed in [Ge+06; RT07; FV13]. In [RT07], for instance, two different approaches were discussed for time-complexity reduction. In one approach, the game is decomposed into multiple two-player PEGs, while the Hamilton-Jacobi-Bellman-Isaacs equation is solved using "fast marching methods" [SV03], as a conditional pursuer-evader assignment is performed. The other proposed approach is a method for solving multi-pursuer and faster single-evader games. The "performance map approach (PMA)" divides the game into multiple minimum-time optimal control problems solved with the "fast marching method". Having the minimum-time maps, the pursuers are able to compute possible interception strategies, while the evader can determine the best escape path. In [Ge+06] a hierarchical decomposition approach for pursuit-evasion differential games was proposed. They aimed to decompose the game, such that it can be solved in a distributed fashion. After a prediction step, all evaders are assigned to a pursuer, resulting in multiple two-player games. Whereas a reassignment of an evader to a pursuer is possible, implying some kind of cooperation between pursuers, an evader is always only playing one two-player game against one single evader, while pursuers without an assigned evader try to remain in "potential future capture regions", which are seen as obstacles the evader will have to avoid. A sophisticated approach for the pursuers is given, yet a game-theoretical solution for the evaders against all other pursuers is neglected. [FV13] provides an analytical solution of a decomposed multi-pursuer single-evader PEG. For certain specially structured differential games, it is shown that a decomposition of the original problem into a family of simpler differential games is possible. These games can be computed by solving the Hamilton-Jacobi-Isaacs equations for each sub-game and constructing optimal strategies for the overall game from those for the simpler problems. At this point this solution approach is applicable on a class of multi-pursuer single-evader games.

Besides the performance map approach presented in [RT07] (stated above) more effort was done in the field of multi-pursuer pursuit of a faster evader. [Wei+07], for instance, states that capturing of a faster evader in a game with perfect state information cannot be guaranteed, while giving a necessary condition for capturing on the plane. In [AS16] a decentralized learning method based on fuzzy reinforcement for pursuing teams in PEGs with a faster evader on the plane is proposed. The learning algorithm is used to tune

the parameters of a formation controller. The pursuer surround the evader, trying to overlap their capture regions. Thereby the evader's escape strategy is to maximize the distance to only the nearest pursuer. This fact restricts the evader strategy considerably. It was not analyzed whether the evader can be captured, while escaping from each pursuer in equal measure. Another multi-pursuer faster-single-evader situation, where the pursuers and the evader form an Apollonius circle was regarded in [JQ10] and [RK15]. Under specific conditions, the pursuers are able to maintain a shrinking Apollonius circle, making an escape of the evader impossible. Both propose evading strategies, aiming to prolong the capture, or even leading to a successful escape, in the case of a gap appearing in the Apollonius circle, caused by a limited number of pursuers. Nevertheless, such a formation is very hard to achieve in real applications, due to dependencies of the initial position of each player.

Generally, PEGs may include static and moving obstacles of all shapes and sizes. Furthermore, all involved players have to be seen as obstacles. Therefore, a collision avoidance capability is indispensable for the players in such PEGs. Two different approaches to this can be found in the literature. Firstly, the obstacles are included in the optimization process, and secondly the obstacle avoidance task is treated as a distinct behavior of the autonomous agent. Regarding the first approach, [GRH10] established a sufficient condition for capture of an evader with a single pursuer in a visibility-based pursuit and an environment with unknown obstacles. Pursuing strategies were developed for a circular and a single unknown convex obstacle. Moreover, [BHB09] presents a solution for a differential game for a visibility-based two-player PEG with holonomic agents in a known bounded environment, showing optimal strategies for a point and hexagonal obstacle and a corner. Following this work, [BBH14a] proposed a solution for a visibility-based PEG in presence of one circular obstacle, while in [BBF14] a numerical approximation for that solution was developed. Later, an extension of that approach to polygonal obstacles for agents with no motion constraints was proposed in [ZB16]. Since those approaches include the obstacle in their optimization process, and in addition can only apply to specific shapes, neglecting the dynamical constraints of the agents, they do not yet seem to be applicable to real applications. The second approach was addressed by [AX09] and [DZJ12]. In [AX09], the two-dimensional environ-

ment was represented by a graph. For each robot an evaluation function determines the vertices, that should be visited to fulfill its mission (i.e., pursue or evade). Furthermore, the collision avoidance is implemented on the wheel-axis-level control. Three sensors cause the wheel motors to speed up or slow down proportionally with the distance to the detected obstacle. [DZJ12] proposes an hybrid algorithm for PEGs in a two-dimensional space, based on improved potential field and differential games. According to the distances to obstacles and the adversarial agent, the algorithm determines either to avoid collisions or to play the PEG.

In summary, the current methods for solving PEG are either extensively time-complex, or the decomposition of the game leads to results far away from those of the full game. Mostly, those approaches make a one-to-one assignment, neglecting the presence of all other players participating in the full game. Moreover, a cooperation in multi-player games within a team is either neglected or performed limitedly. Furthermore, there are no approaches for PEGs, where the obstacle avoidance runs simultaneously to the PEG, but those that include them in their optimization problem as additional constraints. Lastly, all present approaches for PEG are difficult to extend to the N-pursuer-M-evader case. In the following, those issues will be addressed.

1.3 Main Contributions

The overall goal of this thesis is to demonstrate a framework for multi-pursuer single-faster-evader PEGs, which is utilizable in real-time applications with UAVs. Therefore, this work focuses on the following major topics:

- Reducing the time-complexity of a game.
- Introducing the cooperation and the behavior assignment within a team.
- Incorporating the collision avoidance as an integral part of the solution approach.
- Facilitating the extensibility to multi-pursuer multi-evader games.

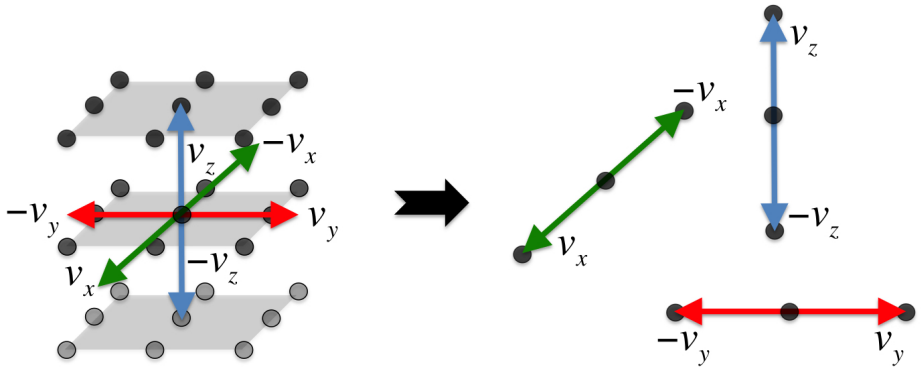

Figure 1.2: *Decomposition of the action space.*

The first part deals with the time-complexity reduction of multi-player games. Generally, this is achieved by two concepts:

i. Action space decomposition

ii. Game decomposition

In PEG all players have to decide whether, and if so, in what amount to move to a specific direction in x, y, and z. As depicted on figure 1.2, the action vector is a combination of inputs manipulating the motion in all directions for an agent. For example, a reference velocity vector for x, y, and z, a decomposition of the action space to motion commands in x, y, and z direction, respectively, is possible. Therefore, e.g., the optimal x-action for each player is determined independently to the optimal action for the y and z direction. This results in three distinct games. It is shown that the time complexity is reduced by an exponent of 3, but at the same time increased by a factor of 3. Moreover, the approach provides close results to the full game. The average terminal stage difference of all considered simulations with full and the decomposed action space, is given by 0.039 stages with a standard deviation of 0.37 stages ($\equiv 0.037s$ capture time).

Next, two different game structures, which are utilizable for multi-player games are presented:

i. The first approach, the Multiple Two-Player Game Decomposition (MTPGD), decomposes the game into multiple two-player games, while

the results in respect to the full game deviate with an average of 0.66 stages with a standard deviation of 6.81 stages ($\equiv 0.681s$ capture time). The cost functionals of each resulting two-player game, are such that the actual position of all other involved players are also taken into account. After the computation of all distinct solutions, they are combined to one overall solution of the game. This approach reduces the time-complexity and enables the usage of more sophisticated solution algorithms, existing only for two-player games, like the Lemke-Howson algorithm [LJTH64]. Using the latter reduces the run-time by a factor of ≈ 70 for $S = 9$, while S is the number of strategies per axis and per player.

ii. The second approach, the Team-Subsumption Two-Player Game (TSTPG), encapsulates each team to one super player with extended action set. While the solution of this type of game is closer to the solution of the full game (average deviation of 0.23 and standard deviation of 3.53 stages ($\equiv 0.353s$ capture time)), there is no improvement in the time complexity. Nevertheless, the usage of the Lemke-Howson algorithm is possible, providing a run-time improvement by a factor of ≈ 21 for $S = 9$, while S is the number of strategies per axis and per player.

In the sequel, a framework is presented, enabling a superordinate cooperation within a team. Due to a negotiation process, team-mates are able to assess, which behavioral strategy maximizes the outcome of the team for a given time horizon. While this approach is generally applicable, it is used here, to introduce the two pursuing behavioral strategies *pursuit* and *battue*. The behavior of a player is switched by altering its cost functional, whereby a higher outcome for the team can be achieved. However, in a PEG with perfect information structure, the gain in outcome is limited (up to 13.25%). The benefit in visibility-based PEG with imperfect information under realistic circumstances is even bigger. This set-up enables the lion-like hunting behavior; one or more pursuers in the visibility of the evader try to drive it towards one or more pursuers outside its visibility. The results against an slower evader are much better under this set-up (up to 73.89%), as the capture of a faster evader is possible. With the results of two-pursuer-one-evader PEG simulations with autonomous agents, sets of initial positions for

which the evader can be captured, for different pursuer-evader speed ratios, have been determined. It is also assumed that members of a team are able to share their observations. Under realistic conditions, such a communication results in a delayed observation sharing. It is found out that the usage of a Kalman Filter makes a capture of an equally fast evader, even with an observation delay of 6 stages, possible.

Besides the above mentioned capabilities of the autonomous agents in the game, a collision avoidance behavior is implemented. The obstacle avoidance is separated from the optimization problem, being a behavior on a lower level of complexity than the non-cooperative and the cooperative game. The repulsion force approach is used to generate a reference velocity vector, which is added to the reference provided by the PEG behavior, resulting in a pursuing/evasion behavior with obstacle avoidance. The simulation results reveal, that a one-pursuer-one-evader game with a faster pursuer still converges in presence of obstacles. Moreover, it can be seen that the capture time increases for initial conditions in the vicinity of obstacles.

Lastly, several practical experiments with an actual multi-rotor UAV prove the real-time feasibility of these approaches.

1.4 Problem Statement and Solution Approach

1.4.1 Problem Statement

A team of UAV pursuers and a single UAV evader with identical dynamic constraints are facing a conflict situation called PEG. The pursuing agents try to capture the evader, while the evader tries to avoid capture. The game takes place in a three-dimensional unbounded and unknown environment. The UAVs are capable of avoiding collisions with static and dynamic obstacles. Moreover, an attitude and linear velocity controller is implemented on each UAV agent.

Since, the situation is considered as a N-player discrete-time deterministic dynamic game, a solution must fulfill the following requirements:

- All agents have to consider that the solution of the problem depends on the decisions of each participant.
- The players must be able to react to unexpected behavior (closed-loop solution).
- The pursuers have to cooperate, allowing the capture of a faster evader.

In general, the solution process of such a game is very complex. It is required, that the time and space complexity of the solution algorithm allow an application in real-time. Regarding present computers, the time-complexity reduction of the solution methods for dynamic games is necessary to deal with multi-pursuer-single-evader PEGs.

1.4.2 Solution Approach

First of all, a two-player PEG with UAVs is defined. This example enables the demonstration of the action space decomposition. This decomposition method is applicable to any type of game, provided the players have more than one controllable degree of freedom.

Based on the action space decomposition, alterations of the game structure itself are regarded. Without loss of generality, a two-pursuer-one-evader PEG with UAVs is chosen for the definition of the approaches. Besides the full game, the MTPGD and the TSTPG approaches are introduced and defined.

The MTPGD, partitions the regarded game in two independent two-player PEGs. It is expected that the time-complexity of the solution algorithm can be reduced by its multiple with increasing strategy space. Moreover, a combination method of the distinct two-player game results has to be found, leading to a solution as close as possible to that of the full game. The MTPGD is supposed to be a decomposition which enables a real-time computation in games with many players and/or strategies demanding a short reaction time. Since it decomposes the game into many independent two-player games, a parallel computation is possible. Consequently, a processor with N computational units, is able to compute the solution of a $2N$-player game without any time loss.

In the TSTPG the two pursuers are combined to one super-pursuer. Obviously, the combination of the pursuers yields in an expanded pursuer's strategy space. Due to this transformation of the game structure, the time complexity does not change at all for the utilized algorithm (npg). Nevertheless, it is expected, that the solution is much closer to the solution of the full game, than it is with the MTPGD. The advantage of this decomposition is that more sophisticated solution approaches, only available for two-player games, can be utilized. The TSTPG is supposed to be used in games with a relatively small number of players and strategies, demanding a more exact solution in respect to the full game.

In order to enable an intelligent team-based pursuit, a novel cooperative approach on top of the non-cooperative game is introduced. A superordinate cooperative team-behavior game is defined, while the solution of this game yields an optimal behavior for every player within a team. For the multi-pursuer-single-evader game, two behavioral strategies are defined: On the one hand, the conventional pursuit and on the other hand the battue behavior. Based on game-theoretical methods for cooperative games, the pursuers are able to negotiate from time to time, which pursuing behavior maximizes their outcome. Therefore, a comparison of three game-theoretical solution methods (Pareto Efficiency, Nash Bargaining Solution, Nash Equilibrium) with the original game is done. While the improvement in games with perfect state information is expected to be limited, games with imperfect information, in particular visibility-based PEGs with UAVs are analyzed, too. It is given, that the pursuers are able communicate with each other in such a set-up. This assumption is used, to analyze whether an equally fast or faster evader can be caught with and without an information-exchange delay of α-stages ($\alpha \in \mathbb{N}$).

In a next step, the cooperative and the non-cooperative game are included, together with all other UAV behaviors, into the Recursive Nested Behavioral Control (RNBC) structure. Using this implementation, a comparison to another approach [RT07] is done, pursuing similar goals, which are: (i) the time-complexity reduction of game and (ii) the capture of a faster UAV evader with a team of UAV pursuers. A performance measure is defined enabling the comparison of the solutions of the two approaches. A three-pursuer-one-evader game is regarded with three different configurations: (i)

slower evader, (ii) equally fast evader, and (iii) faster evader. The Monte-Carlo method is used to perform 1000 simulations per configuration.

The applicability of practical real-time experiments is shown in five different two-pursuer-one-evader PEG. For that, a real hex rotor UAV agent for PEGs is realized. In the first three experiments a perfect state information structure is assumed, while the evader is slower than the two pursuers. The last two experiments describe visibility-based PEG with imperfect information and zero-delay observation sharing between the pursuers. In Experiment 4 the pursuers and the evader have an equal maximal speed, while in experiment 5 a faster evader is assumed. A total of 15 experimental runs are accomplished, while the UAV agent takes every player role in each of the five experiments.

1.5 Outline of the Thesis

In chapter 2 some general assumptions, formulations and definitions are given which are important throughout this thesis.

Chapter 3 presents all game-theoretical solution approaches used in this work. This chapter includes, on the one hand, solution methods for static non-cooperative and cooperative games and, on the other hand, solution methods for dynamic games in discrete-time.

Then, solution concepts, reducing the time-complexity for multi-player games are provided in chapter 4. It is shown, how the time-complexity is reduced by the decomposition of the action space of the game or due to the transformation of the game structure.

Chapter 5 demonstrates how the outcome of a team can be enhanced by using the proposed framework for cooperation and behavior assignment for teams in multi-player games. Moreover, it is analyzed if a faster evader can be caught by a team of pursuers in a visibility-based PEG with imperfect information.

In chapter 6, it is shown how the approaches can be applied to UAV agents. Moreover, a comparison to the “Performance Map Approach” from [RT07]

is performed, being a work pursuing similar goals, namely the cooperative pursue of a faster UAV evader and complexity reduction of the algorithm.

In chapter 7 the realization of a hex rotor test platform is provided, while all relevant parts for the PEG are described in detail. Moreover, a collision avoidance implementation for the PEG-UAV agents is provided.

Next to the last, chapter 8 presents the experimental set-up and the results accomplished with the test platform described in chapter 7.

Finally, a conclusion and a look ahead are stated in chapter 9.

2 Assumptions, Formulations, and Definitions

This chapter describes all fundamental formulations, definitions, and assumptions required across this work. A precise mathematical formulation of pursuit-evasion games lays the foundations for the definition of a solution approach. Further, several assumptions have to be made to specify the regarded problems.

2.1 Assumptions

2.1.1 Unknown Environment with Obstacles

One goal is to create a scenery for pursuit-evasion simulations, which is a tradeoff between generality and implementability. Naturally a three-dimensional environment with randomly arranged obstacles is assumed. Though, an approach is sought for solving PEGs in an infinite general environment, the decisions the players are going to make in the PEG do not depend on the environment. Nevertheless, in a general environment, all players can face moving or static obstacles, being a threat. Therefore, each system could be equipped with a collision avoidance. In chapter 7.6 the collision avoidance extension of the UAV agents is provided. It will be analyzed how the PEG is affected by the collision avoidance.

2.1.2 Discretization of Time and Euclidean Space

The real-world PEGs take place in continuous time and space, but to be able to simulate those scenarios and to calculate solutions numerically on digital computers, it is suitable to discretize the mathematical evolution models and equations, thus the game can be divided into stages of constant time steps Δt with each stage having a distinct solution. The fragmentation of the overall problem into sub-problems enables the usage of several solution methods, like e.g. dynamic programming [Bel57]. The three-dimensional space in which the players are moving is considered to be continuous during a PEG, but for simulation purposes all initial positions a player can take in the environment are represented by a two-dimensional grid with integer values.

2.1.3 Sensing within a Bounded Line of Sight

Realistic scenarios modeled as PEGs are mostly visibility-based PEGs. This leads to at least two key questions in each pursuit-evasion situation a player has to answer: "*Where are my foes?*" and "*Where are my allies?*". Assuming that each player has the ability of self localization and assuming a permanent intra-team communication, the answer to the latter seems trivial. For the former, the following assumption is set. If there is a line of sight (LoS) between two adversarial players and if this LoS is smaller or equal to a pre-specified sensing range r_S, both can observe each other. In other words, the states (e.g. position, linear velocities) of a player can be estimated by the other one and vice versa. Taken together, the intra-team communication and the ability of estimating the states of a foe under specific conditions, enables the observation sharing within a team. Certainly, several sensing devices and filters need to be implemented, to be able to detect and to estimate the states of an adversarial player, but this goes beyond the scope of this work and thus the state estimation capability is assumed to be given.

2.1.4 Multi-Pursuer-Single-Evader PEGs

In this work, only multi-pursuer-single-evader PEGs are regarded. Since, PEGs with an arbitrary number of teams and players are conceivable, an

important question, especially for the pursuers, arises; *Which evader am I going to pursue?* For the main part, this question will not be answered in this thesis. This problem goes beyond the scope of this work. Nevertheless, a solution concept will be introduced in chapter 9. Above all, this work tackles the problem of catching a faster evader due to cooperation.

2.2 The Properties of Pursuit-Evasion Games

As described in chapter 1, PEGs describe a wide class of problems (see figure 1.1). For a mathematical description a unified structure is defined, enabling a precise characterization of such a problem. A *N-team discrete-time deterministic dynamic game with non-fix terminal time* can be defined by a decuplet $\Pi = \{\tau, \mathbf{K}, \mathcal{X}, U, f, \hat{X}, \kappa, \iota, \Gamma, L\}$, with τ being the set of all players, $\mathbf{K}$ being the set of stages, $\mathcal{X}$ being the state space of the game, U being the action space of the game, f being the difference function of the game, $\hat{X}$ being the observation space, κ being the observation function, ι being the state information, Γ being the strategy space, and L being the cost functional of the game. All characteristics and attributes of a game are described in the following sections.

2.2.1 Teams and Players

In terms of game theory, a game is an optimization problem with at least two parties involved. Regardless whether the parties represent one or more players, PEGs describe, in the first instance, a conflicting no-compromise problem. Thus, on the one hand there is a pursuing team consisting of a finite non-empty set of pursuing players $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_j\}, j \in \mathbb{N}$. On the other hand there is a an evading team $\mathbf{E}$ consisting of a finite non-empty set of evading players $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_o\}, o \in \mathbb{N}$. The set $\tau = \mathbf{P} \cup \mathbf{E}$ contains all teams and players involved in a game.

2.2.2 Stages

Since only discrete-time dynamic games are considered in this work, they are divided into stages with a fixed duration Δt. Thus, a set $\mathbf{K} = \{1, 2, \ldots, K\}$

is introduced, denoting the stages of the game, while K is non-fix and is the stage where the game terminates.

2.2.3 Attribute Information

One of the most relevant aspects is the information available to each arbitrary team $\mathbf{J} \subset \tau$ and in particular to each arbitrary player $i \in \tau$. In other words, what does a player know about the other players at a specific time. This includes the state variables, strategies, cost functional, but also the number of involved players and teams in the game. The *attribute information* π_i^k summarizes all relevant characteristics of a game at stage k. Thus, it is a subset of the complete game definition Π. If a player has *complete information*, its *attribute information* π_i^k is given by Π^k, meaning overall knowledge of the game in stage k. Otherwise, *incomplete information* means for a player a partial knowledge about Π at a given stage k, and thus its *attribute information* $\pi_i^k \subset \Pi^k$. In consequence of the cooperation between team members, it is assumed that a team's *attribute information* ${}^{\tau}\pi_{\mathbf{J}}^k$, with $\mathbf{J} \subset \tau$ being an arbitrary team, is the union of all *attribute information* π_j^k, with $j \in \mathbf{J}$ in stage k.

2.2.4 Dynamics

Since the PEG takes place in a region of the three-dimensional Euclidean space, each player $i \in \tau$ can be represented by a set of states. A dynamical model describes the constraints of the state change. The collection of all player's dynamical equations constitutes the dynamical model of the game. Given an infinite set $\mathcal{X}$, being the state space with the states $\mathbf{x}^k \in \mathcal{X}, \forall\ k \in \mathbf{K} \cup \{K+1\}$ and a finite set U with $U = U_1^k \times U_2^k \times \cdots \times U_N^k$, while an element U_i^k, with $k \in \mathbf{K}$ and $i \in \tau$ is the action space of player i in stage k, whereas the elements $\mathbf{u}_i^k$ are all admissible actions of player i in stage k, the difference equation $f^k : \mathcal{X} \times U_1^k \times U_2^k \times \cdots \times U_N^k \to \mathcal{X}$, defined for each $k \in \mathbf{K}$, is

$$\mathbf{x}^{k+1} = f^k(\mathbf{x}^k, \mathbf{u}_1^k, \ldots, \mathbf{u}_N^k), k \in \mathbf{K} \tag{2.1}$$

with $\mathbf{x}^1 \in \mathcal{X}$ being the initial state, describing the evolution of the game state.

2.2.5 Observations

The set $\hat{X} \subseteq \mathcal{X}$ is called the observation set of the game, while $\hat{X}_i^k \subset \hat{X}$ defined $\forall k \in \mathbf{K}$ and $\forall i \in \tau$ is called the observation set of player i in stage k. The elements of these sets $\hat{\mathbf{x}}_i^k$ are the observations of player i in stage k. Moreover, $\exists \kappa_i^k(\mathbf{x}^k) : X \to \hat{X}_i^k, \forall k \in \mathbf{K}$ and $\forall i \in \tau, \kappa_i^k \in \kappa$, being the state-observation equation of player i, while $\kappa = \bigcup_{j \in \tau} \kappa_i^k$.

2.2.6 State Information

As a strict subset of π_i^k the finite set ι_i^k, defined $\forall k \in \mathbf{K}$ and $\forall i \in \tau$, is the *state information* of each distinct player $i \in \tau$, while the collection of all ι_j^k of players j within an arbitrary team $\mathbf{J} \subset \tau$, with $j \in \mathbf{J}$, is the *state information* ${}^{\tau}\iota_{\mathbf{J}}^k$ of the team $\mathbf{J}$ in stage k. The collection of all player's *state information* ι_i^k is the *state information* ι of the game. The topology of the state information ι is given by the state-observation equation κ. Amongst others, the following types of *state information* for a player $i \in \tau$ can be distinguished:

Open-Loop State Information

An open-loop state information means that for all stages k, player $i \in \tau$ knows only the initial state of a game:

$$\iota_i^k = \{\mathbf{x}^1\}, k \in \mathbf{K} \tag{2.2}$$

Closed-Loop Perfect State Information

A player $i \in \tau$ with a closed-loop perfect state information, has access to all previous states of a game in stage k:

$$\iota_i^k = \{\mathbf{x}^1, \ldots, \mathbf{x}^k\}, k \in \mathbf{K} \tag{2.3}$$

Closed-Loop Imperfect State Information

With this type of information a player $i \in \tau$ is able to observe or to measure the states $\hat{x}^k$ of a game entirely or partially in each stage k, depending on the observation function κ_i^k:

$$\iota_i^k = \{\hat{\mathbf{x}}^1, \ldots, \hat{\mathbf{x}}^k\}, k \in \mathbf{K} \tag{2.4}$$

Memoryless Perfect State Information

Contrary to the closed-loop perfect state information, a player $i \in \tau$ with memoryless perfect state information, has access to the initial and to the actual state of a game in stage k:

$$\iota_i^k = \{\mathbf{x}^1, \mathbf{x}^k\}, k \in \mathbf{K} \tag{2.5}$$

Feedback (Perfect) State Information

If a player $i \in \tau$ has only access to the actual state of a game at stage k, a feedback information is present:

$$\iota_i^k = \{\mathbf{x}^k\}, k \in \mathbf{K} \tag{2.6}$$

Feedback Imperfect State Information

Player $i \in \tau$ is able to observe or to measure the state vector of a game $\hat{\mathbf{x}}^k$ in stage k entirely or partially, depending on the observation function κ_i^k:

$$\iota_i^k = \{\hat{\mathbf{x}}^k\}, k \in \mathbf{K} \tag{2.7}$$

Delayed Feedback State Information

If for player $i \in \tau$ the actual state of a game in stage k is only available after a delay α, an α-step delayed feedback state information is given:

$$\iota_i^k = \{\mathbf{x}^{k-\alpha}\}, k \in \mathbf{K}, k > \alpha \tag{2.8}$$

Delayed Imperfect State Information

An α-step delayed imperfect state information is present, if the observations or measurements of player $i \in \tau$ regarding the states of a game at time k are available after a delay α:

$$\iota_i^k = \{\hat{\mathbf{x}}^{k-\alpha}\}, k \in \mathbf{K}, k > \alpha \tag{2.9}$$

Observation Sharing

For a multi-player game it is advantageous for team members to share their information. In other words, a fixed set of players $\mathbf{J} = \{j, \ldots, l\} \subset \tau$ should have access to all states of the game that are available to each player $i \in \mathbf{J}$. Thus, this type of state information is called observation sharing and is defined by the collection of all individual state information of players $i \in \mathbf{J}$ in stage k:

$${}^{os}\iota_i^k = \{\iota_i^k, \iota_j^k, \ldots, \iota_l^k\}, k \in \mathbf{K}, \forall m \in \mathbf{J} \mid m = j, \ldots, l \wedge i \in \mathbf{J} \tag{2.10}$$

Delayed Observation Sharing

Another state information, being relevant for this work is the delayed observation sharing. Due to a required communication channel between a fixed set of players $\mathbf{J} = \{j, \ldots, l\} \subset \tau$ sharing their state information, a transmission delay of α stages could exist, leading to an $\alpha-$step delayed observation sharing in stage k:

$${}^{os}\iota_i^k = \{\iota_i^k, \iota_j^{k-\alpha}, \ldots, \iota_l^{k-\alpha}\}, k \in \mathbf{K}, k > \alpha, \forall m \in \mathbf{J} \mid m = j, \ldots, l \wedge i \in \mathbf{J} \tag{2.11}$$

2.2.7 Strategies

The strategies a player can choose during a game are defined by a set Γ_i^k of mappings γ_i^k: $\mathcal{X} \rightarrow U_i^k$, defined $\forall k \in \mathbf{K}$, and $\forall i \in \tau$, while Γ_i is the strategy space of player i. The strategy space Γ of the game is defined as $\Gamma = \Gamma_1 \times \cdots \times \Gamma_{|\tau|}$.

2.2.8 Cost Functional

The cost functional $L : (\mathcal{X} \times U_1^1 \times \cdots \times U_{|\tau|}^1) \times \cdots \times (\mathcal{X} \times U_1^K \times \cdots \times U_{|\tau|}^K) \to \Re$ for a *discrete-time dynamic game* is formulated as follows:

$$L(\mathbf{u}_1, \ldots, \mathbf{u}_{|\tau|}) = \sum_{k=1}^{K} g^k(\mathbf{x}^{k+1}, \mathbf{u}_1^k, \ldots, \mathbf{u}_{|\tau|}^k, \mathbf{x}^k), \tag{2.12}$$

with $\mathbf{u}_j = (\mathbf{u}_j^{1\prime}, \ldots, \mathbf{u}_j^{K\prime})', j \in \{1, \ldots, |\tau|\}$. This cost functional L is called *stage-additive* and implies the existence of a g^k: $\mathcal{X} \times U_1^k \times \cdots \times U_{|\tau|}^k \times \mathcal{X} \to \Re, k \in \mathbf{K}$.

2.2.9 Termination

The game stops as soon as the terminal set $\Xi \subset \mathcal{X} \times \{1, 2, ...\}$ is reached, meaning for a given $|\tau|$-tuple of actions in stage k, k is the smallest integer with $(\mathbf{x}^k, k) \in \Xi$. With this definition it is possible to describe the dynamic game in normal form (matrix form). Each fixed initial state $\mathbf{x}^1$ and each fixed $|\tau|$-tuple of admissible strategies $\{\gamma_i \in \Gamma_i; i \in \tau\}$ yield a unique set of vectors $\{\mathbf{u}_i^k \triangleq \gamma_i^k(\iota_i^k), \mathbf{x}^{k+1}; k \in \mathbf{K}, i \in \tau\}$, due to the causality of the information and the evolution of the states according to a difference equation. Inserting this vector in L_i $(i \in \tau)$ yields a unique $|\tau|$-tuple of numbers, reflecting the costs of each player. This implicates the existence of the mapping $J_i : \Gamma_1 \times \cdots \times \Gamma_{|\tau|} \to \Re$ for all $i \in \tau$, being also the cost functional of player i with $i \in \tau$. According to that, the spaces $(\Gamma_1, \ldots, \Gamma_{|\tau|})$ and the cost functional $(J_1, \ldots, J_{|\tau|})$ built the normal-form description of the dynamic game with a fixed initial state $\mathbf{x}^1$.

3 Game Theory Preliminary

Game theory is an approach for strategical decision-making, considering that the solution depends on the decision of other agents, while everybody is aware of that. This makes the solution process, especially if the number of players increases, very complex. One can distinguish between non-cooperative and cooperative game theory. Non-cooperative games treat a conflict situation where increasing the pay-off of one player results in decreasing that of another. In cooperative games, however, all players contribute to the benefit of all. In this chapter, all basic concepts for solving static and dynamic games in discrete-time are presented, which are to be used throughout this work. In the first part of this chapter, the non-cooperative game solution conditions are presented. Relevant for this work are the static and dynamic two-player zero-sum games and the static and dynamic N-player non-zero-sum games. Moreover, in the second part, solution conditions for static cooperative games which are used in this work are discussed. All concepts and definitions in this chapter can be found in [BO99; Neu28; Nas50a; Nas50b; Nas53].

3.1 Non-Cooperative Games

3.1.1 Two-Player Zero-Sum Games

Matrix Games

The most common type of zero-sum games are the matrix games. There are two players P_1 and P_2 and an $(n \times m)$ matrix $A = \{a_{ij}\}$ with real values. Each row of matrix A is assigned an admissible strategy γ_i of P_1 and each

column is assigned an admissible strategy γ_j of P_2 (tab. 3.1). That means when the strategy pair (γ_i, γ_j) is played, the outcome of the game is a_{ij}. In a zero-sum game, there is always a minimizer (P_1) and a maximizer (P_2). The outcome of such a game has to be interpreted as follows: Is the outcome positive, P_2 has to pay a_{ij} to P_1, and if its negative, P_1 has to pay to P_2 the amount of $|a_{ij}|$.

$P_1 \vert P_2$	γ_1	γ_2
γ_1	a_{11}	a_{12}
γ_2	a_{21}	a_{22}
γ_3	a_{31}	a_{32}

Table 3.1: *Zero-sum matrix game.*

Assuming that this game has only one stage, it can be said that P_1 is trying to secure a possible loss against P_2 by choosing the strategy which gives the minimum loss, no matter what P_2 does. In other words, P_1 chooses γ_{i*} where the inequality

$$\max_j a_{i*j} \leq \max_j a_{ij} \tag{3.1}$$

with $i = 1, \ldots, m$ holds. The other way around, P_2 will try to secure his loss against P_1 by playing a strategy γ_{j*} while the inequality

$$\min_i a_{ij*} \geq \min_i a_{ij} \tag{3.2}$$

with $j = 1, \ldots, n$ holds.

It is defined that a strategy pair $(\gamma_{i*}, \gamma_{j*})$ constitutes a saddle-point equilibrium if the inequalities

$$a_{i*j} \leq a_{i*j*} \leq a_{ij*}, \tag{3.3}$$

$\forall i = 1, \ldots, n$ and $\forall j = 1, \ldots, m$ are satisfied. If such a strategy pair $(\gamma_{i*}, \gamma_{j*})$ exists, a_{i*j*} is called the value of the game. The general definition for the saddle-point equilibrium in zero-sum games is given in the following section.

Saddle-Point Equilibrium

A tuple of action variables $(\mathbf{u}_1^*, \mathbf{u}_2^*) \in U, U = U_1 \times U_2$ in a two-player game with cost functional L is in saddle-point equilibrium, if

$$L(\mathbf{u}_1^*, \mathbf{u}_2) \leq L(\mathbf{u}_1^*, \mathbf{u}_2^*) \leq L(\mathbf{u}_1, \mathbf{u}_2^*), \forall (\mathbf{u}_1, \mathbf{u}_2) \in U. \tag{3.4}$$

This means that the order of the maximization and minimization done is irrelevant:

$$\min_{\mathbf{u}_1 \in U_1} \max_{\mathbf{u}_2 \in U_2} L(\mathbf{u}_1, \mathbf{u}_2) = \max_{\mathbf{u}_2 \in U_2} \min_{\mathbf{u}_1 \in U_1} L(\mathbf{u}_1, \mathbf{u}_2) = L(\mathbf{u}_1^*, \mathbf{u}_2^*) =: L^* \tag{3.5}$$

Note that if a value exists (a saddle-point exists), it is unique, meaning if another saddle-point $(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2)$ exists, $L(\hat{\mathbf{u}}_1, \hat{\mathbf{u}}_2) = L^*$ applies. Moreover $(\mathbf{u}_1^*, \hat{\mathbf{u}}_2)$ and $(\hat{\mathbf{u}}_1, \mathbf{u}_2^*)$ constitute also a saddle-point. This feature does not hold for Nash equilibria (*non-zero-sum games*). If there is no value in a *zero-sum game*,

$$\min_{\mathbf{u}_1 \in U_1} \max_{\mathbf{u}_2 \in U_2} L(\mathbf{u}_1, \mathbf{u}_2) > \max_{\mathbf{u}_2 \in U_2} \min_{\mathbf{u}_1 \in U_1} L(\mathbf{u}_1, \mathbf{u}_2) \tag{3.6}$$

holds. Hence, there is no saddle-point solution. Therefore, we consider the saddle-point equilibrium in mixed strategies with the following property:

Minimax-Theorem 1 *Each finite two-player zero-sum game has a saddle-point equilibrium in mixed strategies [NM07].*

Saddle-Point Solution in Mixed Strategies

If there is no saddle-point solution in pure strategies, the strategy space is extended, thus, the players can choose their strategies based on random events, leading to the so called mixed strategies. That means, a mixed strategy for a player i is a probability distribution $\mathbf{p}_i$ over the action space U_i. This holds also for general games having no Nash equilibrium. To get a solution in mixed strategies, L_i is replaced by its expected value, according to the chosen mixed strategies, denoted by $J_i(\mathbf{p}_1, \mathbf{p}_2)$. A 2-tuple $(\mathbf{p}_1^*, \mathbf{p}_2^*)$ is a saddle-point equilibrium in mixed strategies of a two-player game, if

$$J(\mathbf{p}_1^*, \mathbf{p}_2) \leq J(\mathbf{p}_1^*, \mathbf{p}_2^*) \leq J(\mathbf{p}_1, \mathbf{p}_2^*), \forall (\mathbf{p}_1, \mathbf{p}_2) \in P,\ P = P_1 \times P_2 \tag{3.7}$$

holds, with $J(\mathbf{p}_1, \mathbf{p}_2) = E_{\mathbf{p}_1,\mathbf{p}_2}[L(\mathbf{u}_1, \mathbf{u}_2)]$. Thus, $J^* = J(\mathbf{p}_1^*, \mathbf{p}_2^*)$ is called the value of the *zero-sum game* in mixed strategies.

3.1.2 Discrete-Time Dynamic Zero-Sum Games

Dynamic Programming for Discrete-Time Dynamic Zero-Sum Games

Stage-Additive Cost Functional

The cost functional for the *discrete-time dynamic game* is formulated as follows:

$$L(\mathbf{u}^k{}_1, \mathbf{u}^k{}_2) = \sum_{k=1}^{K} g_i^k(\mathbf{x}^{k+1}, \mathbf{u}_1^k, \mathbf{u}_2^k, \mathbf{x}^k), \tag{3.8}$$

with $\mathbf{u}_j = (\mathbf{u}_j^{1\prime}, \ldots, \mathbf{u}_j^{K\prime})'$. This cost functional for player i is called "stage-additive" and implies the existence of a $g_i^k : \mathcal{X} \times \mathcal{X} \times U_1^k \times U_2^k \to \Re, k \in \mathbf{K}$.

Information Structure

It is assumed that a feedback information structure is available to all agents during the game $\iota_i^k = \{\mathbf{x}^k\}, k \in \mathbf{K}, i \in \mathbf{N}$.

Since a stage-additive cost functional and a feedback information structure is assumed, dynamic programming and the *Principle of Optimality* [Bel57] can be applied. The solution of the following value function V provides the value of the game. It is defined as:

$$V(k, \mathbf{x}^k) = \min_{\mathbf{u}_1^k \in U_1^k} \max_{\mathbf{u}_2^k \in U_2^k} L(\mathbf{u}_1, \mathbf{u}_2), \tag{3.9}$$

with $k \in \mathbf{K}$, and $\mathbf{x} \in X$. The set of strategies $\{\gamma_i^{k*}(\mathbf{x}^k); k \in \mathbf{K}, i = 1, 2\}$ is for a *two-player discrete-time dynamic zero-sum game* a feedback-saddle-point

solution if, and only if a function $V(k,\cdot) : \Re^n \to \Re, k \in \mathbf{K}$ exists, thus the following recursion is satisfied:

$$\begin{aligned} V_i(k,\mathbf{x}) &= \min_{\mathbf{u}_1^k \in U_1^k} \max_{\mathbf{u}_2^k \in U_2^k} \left[g_i^k \left(f^k(\mathbf{x},\mathbf{u}_1^k,\mathbf{u}_2^k), \mathbf{u}_1^k, \mathbf{u}_2^k, \mathbf{x} \right) \right. \\ &\qquad \left. + V\left(k+1, f^k(\mathbf{x},\mathbf{u}_1^k,\mathbf{u}_2^k)\right) \right] \\ &= \max_{\mathbf{u}_2^k \in U_2^k} \min_{\mathbf{u}_1^k \in U_1^k} \left[g_i^k \left(f^k(\mathbf{x},\mathbf{u}_1^k,\mathbf{u}_2^k), \mathbf{u}_1^k, \mathbf{u}_2^k, \mathbf{x} \right) \right. \\ &\qquad \left. + V\left(k+1, f^k(\mathbf{x},\mathbf{u}_1^k,\mathbf{u}_2^k)\right) \right] \\ &= g_i^k \left(f^k(\mathbf{x},\gamma_1^{k*}(\mathbf{x}),\gamma_2^{k*}(\mathbf{x})), \gamma_1^{k*}(\mathbf{x}), \gamma_2^{k*}(\mathbf{x}), \mathbf{x} \right) \\ &\qquad + V\left(k+1, f^k(\mathbf{x},\gamma_1^{k*}(\mathbf{x}),\gamma_2^{k*}(\mathbf{x}))\right); \end{aligned}$$

$$V(K{+}1,\mathbf{x}) = 0. \tag{3.10}$$

3.1.3 N-Player Non-Zero-Sum Games

N-player non-zero-sum games are a much more general class of games than the two-player zero-sum games regarded before. On the one hand, the game could have more than two participants, and on the other hand, the outcome of the game cannot be described by one single value. In other words, the outcome of the game is an N-tuple of outcome values. Such decision making problems, can be solved by finding a Nash equilibrium as described in the following section.

Nash Equilibrium

An N-tuple of strategies $\{\gamma_{1*}, \gamma_{2*}, \ldots, \gamma_{N*}\}$, $\gamma_{i*} \in \Gamma_i$, $i \in \tau$ constitutes a Nash equilibrium solution for an static finite N-player nonzero-sum game in pure strategies if the inequalities

$$\begin{aligned} a^{1*} &\widehat{=} a^1_{\gamma_{1*},\gamma_{2*},\ldots,\gamma_{N*}} \leq a^1_{\gamma_1,\gamma_{2*},\ldots,\gamma_{N*}} \\ a^{2*} &\widehat{=} a^2_{\gamma_{1*},\gamma_{2*},\ldots,\gamma_{N*}} \leq a^2_{\gamma_{1*},\gamma_2,\ldots,\gamma_{N*}} \\ &\ldots \\ a^{N*} &\widehat{=} a^N_{\gamma_{1*},\gamma_{2*},\ldots,\gamma_{N*}} \leq a^N_{\gamma_{1*},\gamma_{2*},\ldots,\gamma_N} \end{aligned} \tag{3.11}$$

are satisfied $\forall \gamma_i \in \Gamma_i$ and $\forall i \in \tau$. The N-tuple $(a^{1*}, a^{2*}, \ldots, a^{N*})$ is the Nash equilibrium outcome of the N-player game in pure strategies [BO99]. As discussed above for the zero-sum games, it is not guaranteed, that a

Nash equilibrium solution in pure strategies exists. If this is the case, the Nash equilibrium in mixed strategies has to be considered while having the following property:

Nash's Existence Theorem 1 *Every N-person static finite game in normal form admits a non-cooperative (Nash) equilibrium solution in mixed strategies. [BO99]*

Nash Equilibrium in Mixed Strategies

The Nash equilibrium solution in mixed strategies is an optimal probability distribution over the strategy space Γ_i of each player $i \in \tau$. To define the set of inequalities which have to be satisfied to have a Nash equilibrium solution in mixed strategies, the mixed strategy spaces Y^i, $\forall i \in \tau$ with elements y^i are introduced. Moreover, it is defined that its kth component is denoted by y^i_k.

An N-tuple $\{y^{i*}; i \in \tau\}$ constitutes a Nash equilibrium solution in mixed strategies for an N-person finite static game in normal form if the following inequalities

$$\begin{aligned}
J^{1*} &\widehat{=} \sum_{\Gamma_1} \cdots \sum_{\Gamma_N} y^{1*}_{\gamma_1} y^{2*}_{\gamma_2} \dots y^{N*}_{\gamma_N} a^1_{\gamma_1,\gamma_2,\dots,\gamma_N} \\
&\leq \sum_{\Gamma_1} \cdots \sum_{\Gamma_N} y^{1}_{\gamma_1} y^{2*}_{\gamma_2} \dots y^{N*}_{\gamma_N} a^1_{\gamma_1,\gamma_2,\dots,\gamma_N} \\
J^{2*} &\widehat{=} \sum_{\Gamma_1} \cdots \sum_{\Gamma_N} y^{1*}_{\gamma_1} y^{2*}_{\gamma_2} \dots y^{N*}_{\gamma_N} a^2_{\gamma_1,\gamma_2,\dots,\gamma_N} \\
&\leq \sum_{\Gamma_1} \cdots \sum_{\Gamma_N} y^{1}_{\gamma_1} y^{2}_{\gamma_2} \dots y^{N*}_{\gamma_N} a^2_{\gamma_1,\gamma_2,\dots,\gamma_N} \\
&\dots \\
J^{N*} &\widehat{=} \sum_{\Gamma_1} \cdots \sum_{\Gamma_N} y^{1*}_{\gamma_1} y^{2*}_{\gamma_2} \dots y^{N*}_{\gamma_N} a^N_{\gamma_1,\gamma_2,\dots,\gamma_N} \\
&\leq \sum_{\Gamma_1} \cdots \sum_{\Gamma_N} y^{1}_{\gamma_1} y^{2*}_{\gamma_2} \dots y^{N}_{\gamma_N} a^N_{\gamma_1,\gamma_2,\dots,\gamma_N}
\end{aligned} \tag{3.12}$$

$\forall y^j \in Y^i$, $j \in \tau$, hold. The N-tuple $(J^{1*}, \dots, J^{N*})$ is the noncooperative Nash equilibrium outcome of the N-player game in mixed strategies. [BO99]

Each type of game presented in this chapter has a Nash equilibrium in mixed strategies. The Nash equilibriums in mixed strategies include also those in

pure strategies, since they are a generalization of the latter. Note, that the Nash equilibrium solution of a zero-sum game is identical to the saddle-point equilibrium solution.

3.1.4 Discrete-Time Dynamic N-Player Non-Zero-Sum Games

Dynamic Programming for Discrete-Time Dynamic N-Player Games

Stage-Additive Cost Functional

The cost functional for the *discrete-time dynamic game* is formulated as follows:

$$L(\mathbf{u}^k{}_1, \ldots, \mathbf{u}^k{}_{|\tau|}) = \sum_{k=1}^{K} g_i^k(\mathbf{x}^{k+1}, \mathbf{u}_1^k, \ldots, \mathbf{u}_{|\tau|}^k, \mathbf{x}^k), \tag{3.13}$$

with $\mathbf{u}_j = (\mathbf{u}_j^{1\prime}, \ldots, \mathbf{u}_j^{K\prime})'$. This cost functional for player i is called "stage-additive" and implies the existence of a $g_i^k : \mathcal{X} \times \mathcal{X} \times U_1^k \times \cdots \times U_{|\tau|}^k \to \Re, k \in \mathbf{K}$.

Information Structure

It is assumed that a feedback information structure is available to all agents during the game $\iota_i^k = \{\mathbf{x}^k\}, k \in \mathbf{K}, i \in \mathbf{N}$.

The solution of the following value function provides the value of the game. It is defined as:

$$V(k, \mathbf{x}^k) = \min_{\mathbf{u}_i^k \in \mathbf{U}_i^k} L(\mathbf{u}_1, \ldots, \mathbf{u}_{|\tau|}), \tag{3.14}$$

with $k \in \mathbf{K}$, $i \in \{1, \ldots, |\tau|\}$, and $\mathbf{x} \in X$.

The set of strategies $\{\gamma_i^{k*}(\mathbf{x}^k); k \in \mathbf{K}, i = 1, \ldots, |\tau|\}$ is for a *N-player discrete-time dynamic zero-sum game* a feedback-nash-equilibrium solution if, and only if a function $V(k, \cdot) : \Re^n \to \Re, k \in \mathbf{K}$ exists, thus the following recursion is satisfied:

$$
\begin{aligned}
V_i(k,\chi^k) = \min_{\mathbf{u}_i^k \in \mathbf{U}_i^k} \Big[& \mathfrak{g}_i^k \left(\tilde{\mathfrak{f}}_i^{k*}(\chi^k, \mathbf{u}_i^k), \gamma_1^{k*}(\chi^k), \ldots, \gamma_{i-1}^{k*}(\chi^k), \mathbf{u}_i^k, \right. \\
& \left. \gamma_{i+1}^{k*}(\chi^k), \ldots, \gamma_N^{k*}(\chi^k), \chi^k \right) \\
& + V_i(k+1, \tilde{\mathfrak{f}}_i^{k*}(\chi^k, \mathbf{u}_i^k)) \Big] \\
= \ & \mathfrak{g}_i^k \left(\tilde{\mathfrak{f}}_i^{k*}(\chi^k, \gamma_i^{k*}(\chi^k)), \gamma_1^{k*}(\chi^k), \ldots, \gamma_N^{k*}(\chi^k), \chi^k \right) \\
& + V_i \left(k+1, \tilde{\mathfrak{f}}_i^{k*}(\chi^k, \gamma_i^{k*}(\chi^k)) \right); \\
V_i(K+1, \chi^{K+1}) = 0, & i \in \mathbf{N}, \qquad (3.15)
\end{aligned}
$$

with $\tilde{\mathfrak{f}}_i^{k*}(\chi^k, \mathbf{u}_i^k) \widehat{=}$
$\mathfrak{f}^k \left(\chi^k, \gamma_1^{k*}(\chi^k), \ldots, \gamma_{i-1}^{k*}(\chi^k), \mathbf{u}_i^k, \gamma_{i+1}^{k*}(\chi^k), \ldots, \gamma_N^{k*}(\chi^k) \right)$.

3.2 Cooperative Games

In this work, bargaining problems are considered. A bargaining problem is a situation in which:

- N-players with specific objectives seek a mutual agreement, with $N > 1$.
- Each player has to approve the agreement, meaning that a disagreement is possible.
- A mutual agreement is reachable.
- Presence of a conflict of interests among players about the agreements.

That means, there are two possibilities when solving such a game: The players agree or disagree. In the following section, solution methods for the latter problem are presented.

3.2.1 Nash Bargaining Solution

In [Nas50b] a solution concept for a two player bargaining problem is presented. Nash's solution method is an axiomatic approach which abstracts the

details of a bargaining process and considering only the outcomes and agreements that satisfy "reasonable" properties. Nash claimed, that this solution approach yields a fair bargaining solution, which will be accepted by each rational player. The Nash bargaining solution is calculated as follows: The result $u*$ of a two-player bargaining game (P, c), with P being the outcome matrix and c being the conflict outcome when the player disagree, is given by the outcome vector $u \in P$, for which $u_i > c_i$, $\forall i \in \tau$, and which maximizes the Nash product $NP = (u_1 - c_1)(u_2 - c_2)$. Then, for the maximum Nash product NP, one has

$$NP^* = (u_1^* - c_1)(u_2^* - c_2) \tag{3.16}$$

such that $u* = (u_1^*, u_2^*) \in P$ and $u_i^* > c_i$, $\forall i \in \tau$. According to Nash, this bargaining solution is unique and satisfies the following four axioms:

- Pareto optimality : No player can be better off without making at least one player worse off.
- Symmetry: If the players are indistinguishable, the solution should not discriminate between them.
- Invariant to affine transformations: An affine transformation of the payoff and disagreement point should not alter the outcome of the bargaining process.
- Independence of irrelevant alternatives: If the solution (u_1, u_2) chosen from a feasible set P is an element of a subset $B \subseteq P$, then (u_1, u_2) must be chosen from B.

Extension to N-Players

The result $u*$ of an N-player bargaining game (P, c), with P being the outcome matrix and c being the conflict outcome when the player disagree, is given by the outcome vector $u \in P$, for which $u_i > c_i$, $\forall i \in \tau$, and which maximizes the Nash product $NP = (u_1 - c_1)(u_2 - c_2)\dots(u_N - c_N)$. Then, for the maximum Nash product NP, one has

$$NP^* = (u_1^* - c_1)(u_2^* - c_2)\dots(u_N^* - c_N^*) \tag{3.17}$$

such that $u* = (u_1^*, u_2^*, \ldots, u_N^*) \in P$ and $u_i^* > c_i$, $\forall i \in \tau$.

Conflict Outcome

In order to calculate the Nash bargaining solution, a conflict outcome has to be defined. When no agreement is achieved, the players get a conflict outcome. Thus it is favorable to choose a non-cooperative Nash equilibrium as conflict outcome, which is followed throughout this work.

3.2.2 Pareto Efficiency

As stated before, no player can be better off without making at least one player worse off. Regarding a bargaining game P, with P being the outcome matrix, a set $U \subseteq P$ with elements $u_i^* \in U$, $\forall i \in \tau$ is called the set of Pareto optimal outcomes, if there exist no other elements $u_i \in P$ so that for all $i = 1, \ldots, |\tau|$

$$u_i \geq u_i^* \tag{3.18}$$

and for at least one player

$$u_i > u_i^* \tag{3.19}$$

holds.

Solution Choice

Since the Pareto Efficiency Method can yield more than one solution (Pareto optimal set U), a superordinate selection has to be made, regarding the application in this work. Therefore, the solution $u^* \in U$ is chosen, with the property:

$$\min u^* \leq \min u, \tag{3.20}$$

$\forall u \in U$, for minimizing players, and

$$\max u^* \leq \max u, \tag{3.21}$$

$\forall u \in U$, for maximizing players.

4 Time-Complexity Reduction for Multi-Player Games

The purpose of this chapter is to introduce a methodology for the reduction of the complexity of the full PEG's solution process. This is based on two different decompositions. The decomposition of the action space and the decomposition of the original game.

4.1 Full-Dimensional Action Space vs Decomposed Action Space

In some applications it is possible to decompose the problem at a low level, like for example the decomposition of the single-action game through decomposition of the action space. The single-action game is defined as the game which is played in one stage k. Regarding a full single-action PEG, the action space is for example given by $\mathbf{U}_i \subset V_x \times V_y \times V_z$ for each player $i \in \mathbf{N}$, where V_x, V_y, and V_z are linear velocity sets in x, y, and z direction. It can be seen that the actions are a combination of admissible actions $u_x \in V_x$, $u_y \in V_y$, and $u_z \in V_z$. This circumstance enables the decomposition of the single-action game, into three distinct games where the optimal actions $u_x^* \in V_x$, $u_y^* \in V_y$, and $u_z^* \in V_z$ are sought (see figure 1.2). In other words, the optimal linear velocities for x, y, and z are independently calculated and combined afterwards to a single optimal velocity vector $\mathbf{u}^* = (u_x^*, u_y^*, u_z^*)$. First of all, for being able to verify the decomposition approach, an example problem has to be defined. Without loss of generality, a two-player PEG with UAVs is regarded.

4.1.1 Two-Player PEG with UAVs (Full Game)

The PEG is described by following characteristics:

- A set of two players $\{e, p\}$.
- A set $\mathbf{K} = \{1, \dots, K\}$ with variable number of stages K. K is the time p needs to capture e, i.e., to minimize the distance d_ϵ to player e (e reaches the terminal set Ξ). Thus, K depends on the initial states of e and p.
- A set $\mathbf{X} = X \times Y \times Z \times V_x \times V_y \times V_z \times \Phi \times \Theta \times \Psi \times P \times Q \times R$ being the state space, while the sets X, Y and Z span the position space, V_x, V_y, and V_z span the linear velocity space, Φ, Θ, and Ψ span the attitude space, and P, Q, and R span the angular velocity space.
- The terminal set $\Xi \subset X \times Y \times Z \times \{1, 2, \dots\}$ is the set of all elements $\xi \in \Xi$ of a sphere around the pursuer's position (x_p, y_p, z_p) with radius d_ϵ in stage k.
- Two finite discrete action spaces $U_p = U_e \subset V_x \times V_y \times V_z$. U_p and U_e are steady during each stage k of the game. They are defined as $U_p = \Bigg\{ \mathbf{u}_{pu,1} + i\frac{\mathbf{u}_{pu,2} - \mathbf{u}_{pu,1}}{s}, \mathbf{u}_{pv,1} + j\frac{\mathbf{u}_{pv,2} - \mathbf{u}_{pv,1}}{s}, \mathbf{u}_{pw,1} + l\frac{\mathbf{u}_{pw,2} - \mathbf{u}_{pw,1}}{s} \Bigg\}$, with $i = 0, \dots, s$; $j = 0, \dots, s$; $l = 0, \dots, s$ and $U_e = U_p$, while $(s+1)^3$ is the number of strategies available for each player and $[\mathbf{u}_{pu,1}, \mathbf{u}_{pu,2}] = [\mathbf{u}_{pv,1}, \mathbf{u}_{pv,2}] = [\mathbf{u}_{pw,1}, \mathbf{u}_{pw,2}] = [\mathbf{u}_{eu,1}, \mathbf{u}_{eu,2}] = [\mathbf{u}_{ev,1}, \mathbf{u}_{ev,2}] = [\mathbf{u}_{ew,1}, \mathbf{u}_{ew,2}] = [-1, 1]$ are the continuous action spaces. $\mathbf{u}_p$ and $\mathbf{u}_e$ are elements of the sets U_p and U_e, while $\mathbf{u} \in U_p \times U_e$.

- The state of the PEG between two UAVs in the pursuers reference frame is defined as

$$\mathbf{x}^k = \mathbf{x}_e^k - \mathbf{x}_p^k = \begin{bmatrix} x^k \\ y^k \\ z^k \\ v_x^k \\ v_y^k \\ v_z^k \\ \phi^k \\ \theta^k \\ \psi^k \\ p^k \\ q^k \\ r^k \end{bmatrix} = \begin{bmatrix} x_e^k - x_p^k \\ y_e^k - y_p^k \\ z_e^k - z_p^k \\ v_{xe}^k - v_{xp}^k \\ v_{ye}^k - v_{yp}^k \\ v_{ze}^k - v_{zp}^k \\ \phi_e^k - \phi_p^k \\ \theta_e^k - \theta_p^k \\ \psi_e^k - \psi_p^k \\ p_e^k - p_p^k \\ q_e^k - q_p^k \\ r_e^k - r_p^k \end{bmatrix} = \begin{bmatrix} x_1^k \\ x_2^k \\ x_3^k \\ x_4^k \\ x_5^k \\ x_6^k \\ x_7^k \\ x_8^k \\ x_9^k \\ x_{10}^k \\ x_{11}^k \\ x_{12}^k \end{bmatrix} \tag{4.1}$$

with the difference function $\mathbf{x}^{k+1} = f\left(\mathbf{x}^k, h(\mathbf{w}^k)\right)$ given by:

$$\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \\ x_4^{k+1} \\ x_5^{k+1} \\ x_6^{k+1} \\ x_7^{k+1} \\ x_8^{k+1} \\ x_9^{k+1} \\ x_{10}^{k+1} \\ x_{11}^{k+1} \\ x_{12}^{k+1} \end{bmatrix} = \begin{bmatrix} x_1^k \\ x_2^k \\ x_3^k \\ x_4^k \\ x_5^k \\ x_6^k \\ x_7^k \\ x_8^k \\ x_9^k \\ x_{10}^k \\ x_{11}^k \\ x_{12}^k \end{bmatrix} + \begin{bmatrix} x_4^k \\ x_5^k \\ x_6^k \\ -(cx_7^k sx_8^k cx_9^k + sx_7^k sx_9^k)\frac{v_1^k}{m} \\ -(cx_7^k sx_8^k sx_9^k + sx_7^k cx_9^k)\frac{v_1^k}{m} \\ g - cx_7^k cx_8^k \frac{v_1^k}{m} \\ x_{10}^k \\ x_{11}^k \\ x_{12}^k \\ \frac{I_y - I_z}{I_x} x_{11}^k x_{12}^k + \frac{L}{I_x} v_2^k - \frac{I_R}{I_x} x_8^k g(\boldsymbol{v}) \\ \frac{I_z - I_x}{I_y} x_{10}^k x_{12}^k + \frac{L}{I_y} v_3^k - \frac{I_R}{I_y} x_7^k g(\boldsymbol{v}) \\ \frac{I_x - I_y}{I_z} x_{10}^k x_{11}^k + \frac{1}{I_z} v_4^k \end{bmatrix} \Delta t, \tag{4.2}$$

with $\boldsymbol{v} = (v_1, v_2, v_3, v_4)^T, \boldsymbol{v} \in \boldsymbol{\Upsilon}$ being the inputs for altitude, roll, pitch and yaw, I_x, I_y, I_z are the inertia around x, y, z-axes, I_r is the rotor moment of inertia, l is the length between the center of gravity of the UAV and the center of one rotor, g is the gravitation constant, $g(\boldsymbol{v})$ is a function of $\boldsymbol{v}$ depending on the rotor's angular velocities, and

Δt is the sampling time, while s and c are abbreviations for $\sin(\cdot)$ and $\cos(\cdot)$, respectively. The function

$$h : V_x \times V_y \times V_z \rightarrow \Upsilon, \tag{4.3}$$

yields an input vector υ given an admissible action $\mathbf{u}^k_{i\in\mathbf{N}}$. A more detailed description of the utilized UAV system and its controllers, i.e. attitude and velocity control, can be found in chapter 7.

- A feedback perfect state information structure $\iota^k_e = \iota^k_p = \{\mathbf{x}^k\}, \forall k \in \mathbf{K}$.
- The strategy spaces $\Gamma_p = U_p$ and $\Gamma_e = U_e$.
- A cost functional J

$$J(\mathbf{q}^k_p, \mathbf{q}^k_e) = E\left[\sum_{k=1}^{K} \mathrm{D}(f\left(\mathbf{x}^k, h(\mathbf{w}^k)\right), \mathbf{x}^k)\right], \tag{4.4}$$

with $\mathrm{D}(\cdot)$ being a function describing the change in distance between p and e in one stage k, playing the control action $(\mathbf{u}^k_p, \mathbf{u}^k_e)$.

- The value function

$$V(k, \mathbf{x}^k) = \min_{\mathbf{q}^k_p} \max_{\mathbf{q}^k_e} J(\mathbf{q}^k_p, \mathbf{q}^k_e). \tag{4.5}$$

- $\mathbf{q}^{k*} = (\mathbf{q}^{k*}_p, \mathbf{q}^{k*}_e)$ is the optimal solution of the game in stage k. It is calculated by solving the closed-loop saddle-point equilibrium in mixed strategies. The optimal probability distributions $\mathbf{q}^{k*} = (\mathbf{q}^{k*}_p, \mathbf{q}^{k*}_e)$ over the action space $U_p \times U_e$ in stage k is given by

$$\mathbf{q}^{k*} = arg V(k, \mathbf{x}^k), \forall k \in \mathbf{K}. \tag{4.6}$$

- The optimal control actions $\mathbf{u}^{k*} = (\mathbf{u}^{k*}_p, \mathbf{u}^{k*}_e)$ are those where the probabilities $\mathbf{q}^{k*}_p$ and $\mathbf{q}^{k*}_e$ are maximal. The reference velocities for the pursuers' and evaders' velocity controller are given by

$$\mathbf{v}^{r,k}_p = (v^k_{xp}, v^k_{yp}, v^k_{zp})^T + (\mathbf{u}^{k*}_p)^T \tag{4.7}$$

and

$$\mathbf{v}_e^{r,k} = (v_{xe}^k, v_{ye}^k, v_{ze}^k)^T + (\mathbf{u}_e^{k*})^T. \tag{4.8}$$

4.1.2 Decomposition of the Action Set

While all other properties from above stay the same, the action spaces $\mathbf{U}_i \subset V_x \times V_y \times V_z$ are divided into three distinct action spaces for each player $i \in \mathbf{N}$. As a consequence, the three action spaces are defined as: $\mathbf{U}_{ix} = \left(u_{i,u,1} + a\frac{u_{i,u,2} - u_{i,u,1}}{s_{i,u}}, 0, 0 \right)$, $\mathbf{U}_{iy} = \left(0, u_{i,v,1} + b\frac{u_{i,v,2} - u_{i,v,1}}{s_{i,v}}, 0 \right)$, and $\mathbf{U}_{iz} = \left(0, 0, u_{i,w,1} + c\frac{u_{i,w,2} - u_{i,w,1}}{s_{i,w}} \right)$ with $a \in [0, s_{i,u}]_{\mathbb{N}}$, $b \in [0, s_{i,v}]_{\mathbb{N}}$ and $c \in [0, s_{i,w}]_{\mathbb{N}}$ while $(s_{i,u} + 1)$, $(s_{i,v} + 1)$, and $(s_{i,w} + 1)$ are the number of available actions of player i for each decomposed single-action game. Given the maximal velocity vector $\mathbf{v}_i^{max} = (v_{i,x}^{max}, v_{i,y}^{max}, v_{i,z}^{max})^T \in V_x \times V_y \times V_z$ of a player i, the continuous action spaces are given by $[u_{i,u,1}, u_{i,u,2}] = [-\frac{v_{i,x}^{max}}{2}, \frac{v_{i,x}^{max}}{2}]$, $[u_{i,v,1}, u_{i,v,2}] = [-\frac{v_{i,y}^{max}}{2}, \frac{v_{i,y}^{max}}{2}]$ and $[u_{i,w,1}, u_{i,w,2}] = [-\frac{v_{i,z}^{max}}{2}, \frac{v_{i,z}^{max}}{2}]$. Solving the above defined game for each of the three action sets in a stage k, three distinct solutions u_x^{k*}, u_y^{k*}, and u_z^{k*} are obtained. The distinct solutions are combined such that an optimal linear velocity vector for all three directions $\mathbf{u}^{\mathbf{k}*} = (u_x^{k*}, u_y^{k*}, u_z^{k*})$ is constituted.

Solving the full single-action game, means finding an optimal strategy combination while each player has a number of $(s_{i,u}+1)\cdot(s_{i,v}+1)\cdot(s_{i,w}+1)$ admissible actions. Due to the decomposition, three distinct games have to be solved while each player has a number of $(s_{i,u}+1)$, $(s_{i,u}+1)$, or $(s_{i,u}+1)$ admissible actions, respectively. Assuming, that $(s_{i,u}+1) = (s_{i,v}+1) = (s_{i,w}+1) = S$, each player $i \in \mathbf{N}$ has a number of S^3 admissible actions in one stage k when playing the full game and S admissible actions when playing one of the three sub-games resulting from the decomposition. Regarding a N-player game, the optimal solution is sought in an action space with cardinality S^{3N} for the full game and $3 \cdot S^N$ for the decomposed one. It can be seen that the cardinality of the action space is always greater for $S > 1$ by an exponent

of 3 in the full game, and thus the time complexity of finding a solution is obviously larger.

4.1.3 Time-Complexity Analysis

The resource-intensive part of the algorithm solving the PEG is the function NPG which computes the Nash equilibrium in mixed strategies in one stage of the game. This function transforms the game to a non-linear optimization problem with non-linear constraints and solves it using the *SLSQP* algorithm. According to the documentation of the NLOpt package [Joh13], which is used by the authors to implement this algorithm, the *SLSQP* algorithm requires $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^3)$ time in n dimensions. The dimension of the optimization problem depends, according to [Cha09], on the number of strategies of each player and on the number of involved players, i. e. on all possible strategy combinations. For instance, a game with three players and two strategies per player gives an overall number of eight strategy combinations, and thus an optimization problem with eighth dimension has to be solved. Regarding the two-player PEG with full dimensional action space, each player has $S = s^3$ admissible actions per stage, while $s \in \mathbb{N}$ is the number of options for each degree of freedom. The solution of this game needs $\mathcal{O}(s^6)$ time. With a decomposed action space, the time complexity for the calculation is given by $3 \cdot \mathcal{O}(s^3)$. For $s > 1$ the time complexity of the solution process is always for the decomposed game.

4.1.4 Comparison

The comparison of the game with full-dimensional action space and the game with decomposed action space is accomplished through several simulations. Following assumptions are set:

- Since the chosen optimal control actions represent a velocity change in three linear directions of p and e, a maximum velocity $\mathbf{v}_{max}$ with $\mathbf{v}_p^{max} = \begin{bmatrix} 15 & 15 & 3.5 \end{bmatrix}^T$ and $\mathbf{v}_e^{max} = \frac{\mathbf{v}_p^{max}}{1.5}$ and a maximal absolute value of $v_p^{maxA} = 15$ for the pursuer and $v_e^{maxA} = 10$ for the evader is defined.

- The numerical solution of the PEG is computed by solving it for each initial positions $(x^1, y^1, z^1) \in X \times Y \times Z$, while x^1 and y^1 take integer values in a 61×61 grid, with $X = [-30, 30]$ and $Y = [-30, 30]$ in the pursuer's reference frame (pursuer's position is the origin). In each simulation, the initial altitude of both UAVs is 20, i.e., $z^1 = -20$. This is necessary for the visualization of the value function.

- The simulations are carried out for $S = 3$, $S = 5$, $S = 7$, and $S = 9$, i.e., each player has 3^3, 5^3, 7^3 9^3 available strategies in each time step k for the full game, while in the decomposed version three sub-games are played in each time step k with 3, 5, 7, 9 available strategies, respectively.

- The stage duration is chosen to be $\Delta T = 0.1$, while the velocity control is sampled with $\Delta t = 0.005$.

- A capture distance $d_\epsilon = 5$ is chosen.

Evaluation

To verify whether the decomposition approach yields approximate results to the full game, a difference between the stage counts for capture of both variants for each regarded initial state is carried out. In this way, it can be seen where the decomposed game deviates from the full one. As depicted in Figure 4.1, 89.57% of the initial states yield for both variants the same terminal time, while all differences are between -3 and 4 stages. The average of the value difference is given with 0.039 stages, having a standard deviation of 0.37 stages. Thus, the decomposed action-space game solution yields similar results to the full-dimensional action space game solution. These result are expected due to the fact that the non-holonomic connections are neglected, due to the reduction to point-mass motion, and due to the convexity of the cost functional.

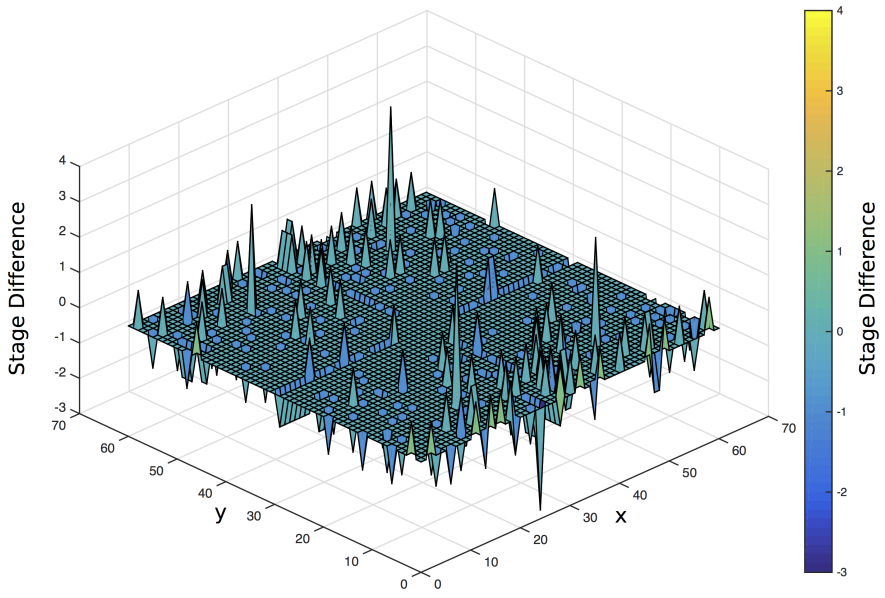

Figure 4.1: *Difference of the values of the game with full-dimensional action space and game with decomposed action space.*

Figure 4.2 depicts the computational time for one stage depending on the number of strategies. The progression for the decomposition approach is represented by the green curve, while the red curve shows it for the full game. A comparing of those two curves reveals that with increasing S the computational time of one stage of the full game increases tremendously. While for $S = 3$ the computational time of the decomposed approach is with $0.393s/Stage$ about twice as fast as the full game with $0.65s/Stage$, the first significant gain can be observed for $S = 5$ (decomposition: $0.73s/Stage$, full game: $13.37s/Stage$, factor: 18.32). For $S = 7$ and $S = 9$ the computational time of the decomposed is, with $1.32s/Stage$ and $2.65s/Stage$, in respect to the full game solution, with $98.44s/Stage$ and $453.61s/Stage$, 74.58 or 171.17 times faster. Due to the vast increasing of the computational time with increasing S, no more simulations with $S > 9$ could be carried out in reasonable time. Regarding the fact that this approach yields approximate results in much less time, it will be used for the single-action decomposition from this point on throughout the remaining work.

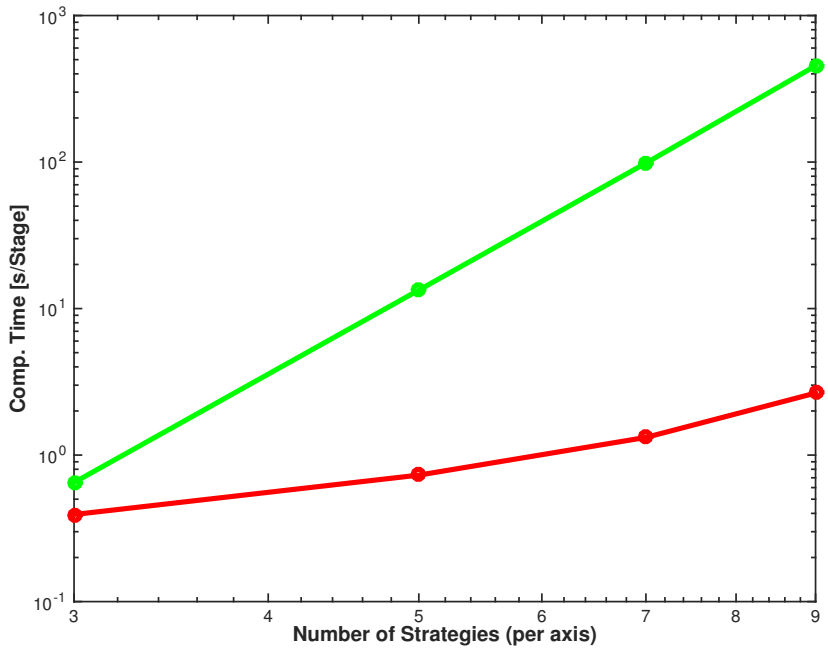

Figure 4.2: *Comparison of computational times for one stage.*

4.2 Multi-Player Game Decomposition

In respect to two-player pursuit-evasion games, multi-player pursuit-evasion games with an arbitrary number of players are much more complex to solve. The game changes from a pure non-cooperative into a cooperative one between the players of one team, and to a non-cooperative one between the parties. Thus, the complexity of solving such games increases massively. The requirements on the solution approach are the time-complexity reduction by decomposition of the full game, and the parallel computability by allocating the decomposed game to multiple processing units. This results in a faster solution derivation. The computational time is constant with an increasing number of adversarial players as long as each involved player has as many processors as foes available for the solution computation. Hence, an appropriate game structure has to be found. The concept and the results of the following parts in this chapter have been published in [AB16].

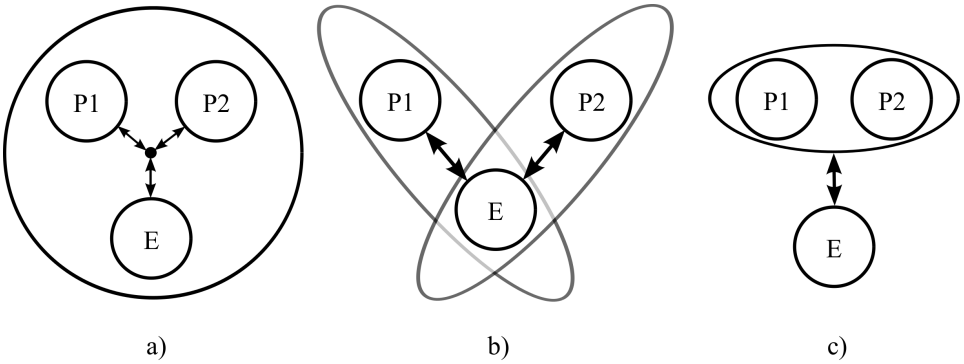

Figure 4.3: *Two-pursuer-one-evader pursuit-evasion game structures.*

Two possible game structures are proposed and compared to the full game regarding time-complexity, convergence, and solution of a two-pursuer-one-evader PEG. Figure 4.3a) depicts the original game structure of the PEG, while 4.3b) and 4.3c) depict two other structures for this game. In method 4.3b), the Multiple Two-Player Game Decomposition (MTPGD), the game is solved by decomposing the multi-player game into multiple two-player games between each pursuer and each evader. In the specific case of a three-player PEG, it is decomposed into two two-player games. Since those two games have both distinct solutions a fusion of the latter must be performed. In the second method depicted in 4.3c), the Team-Subsumption Two-Player Game (TSTPG), each team is treated as one super-player, resulting in only one two-player PEG. Hence, all distinct states like the position, velocity and attitude of all players within a team are encapsulated into a single super-player state and the strategy space of this super-player contains each possible strategy combinations of all encapsulated players, meaning if each of them has n admissible strategies, the super-player has n^m admissible strategies, with m being the number of players within the team. Both pursuers try to minimize their distance to the evader, while the evader tries to maximize it. At this point, there is no cooperation (agreement) between the pursuers assumed. The pursuing team wins, if one of the pursuers catches the evader and the evader wins if the game never stops (diverges). With the help of simulation results all three different structures of the game have to be analyzed and an answer to the following questions has to be given:

- Does the game converge?

- Does the solution of a game with changed structure yield other results than the original one?
- Does another structure give a gain in run-time?

Therefore, a two-pursuer-one-evader PEG with multi-rotor UAVs is defined and solved with the above described approaches. It is assumed that all multi-rotor UAVs have an identical system structure and identical dynamical constraints as described in chapter 7. All systems can be controlled by providing a linear velocity reference vector in x, y, z-direction. The solution of the PEG provides optimal linear velocity vectors for all involved players.

4.2.1 Two-Pursuer-One-Evader Pursuit-Evasion Game

The PEG $\mathbf{\Pi} = (\mathbf{N}, \mathbf{K}, \mathcal{X}, \mathbf{U}, \mathfrak{f}, \hat{\mathbf{X}}, \kappa, \iota, \mathbf{\Gamma}, L)$ between the three UAV players is described by following properties:

Players

$\mathbf{N} = \mathbf{P} \cup \mathbf{E}$ is the set of all players, while $\mathbf{P} = \{p_1, p_2\}$ is the team of pursuers and $\mathbf{E} = \{e\}$ the evader.

Termination

$\mathbf{K} = \{1, \ldots, K\}$ is the integer set of stages with K time steps. K is called the terminal time. K is finite if the pursuing team $\mathbf{P}$ catches the evader e and infinite if e manages to escape. e is caught if a pursuer $p \in \mathbf{P}$ reaches a terminal position in $\xi^k \subset X \times Y \times Z$ and thus $(\chi^k, k) \in \mathbf{\Xi}$ holds. ξ^k is defined by a sphere with radius d_ϵ around the evader's position $\mathbf{pos}_e^k = (x_e^k, y_e^k, z_e^k)^T$ in the inertial frame.

State Space

The state space $\mathbf{X}$ of each player is given by $\mathbf{X} = X \times Y \times Z \times V_x \times V_y \times V_z \times \Phi \times \Theta \times \Psi \times P \times Q \times R$ with $\mathbf{x}_{i \in \mathbf{N}}^{k \in \mathbf{K}} \in \mathbf{X}$ having the subspaces: position space

$(X \times Y \times Z)$, linear velocity space $(V_x \times V_y \times V_z)$, Euler angles $(\Phi \times \Theta \times \Psi)$, and angular velocity space $(P \times Q \times R)$.

$\mathcal{X} = (\mathbf{X})^3$ is the state space of the game with $\chi^k = (\mathbf{x}_{p_1}^k, \mathbf{x}_{p_2}^k, \mathbf{x}_e^k) \in \mathcal{X}$, while $\mathbf{x}_i^k \in \mathbf{X}$ is the respective state vector of player $i \in \mathbf{N}$ in stage $k \in \mathbf{K}$. Hence,

$$\chi^k = (\mathbf{x}_{p_1}^k, \mathbf{x}_{p_2}^k, \mathbf{x}_e^k) = \left(\begin{bmatrix} x_{p_1}^k \\ y_{p_1}^k \\ z_{p_1}^k \\ v_{x,p_1}^k \\ v_{y,p_1}^k \\ v_{z,p_1}^k \\ \phi_{p_1}^k \\ \theta_{p_1}^k \\ \psi_{p_1}^k \\ p_{p_1}^k \\ q_{p_1}^k \\ r_{p_1}^k \end{bmatrix}, \begin{bmatrix} x_{p_2}^k \\ y_{p_2}^k \\ z_{p_2}^k \\ v_{x,p_2}^k \\ v_{y,p_2}^k \\ v_{z,p_2}^k \\ \phi_{p_2}^k \\ \theta_{p_2}^k \\ \psi_{p_2}^k \\ p_{p_2}^k \\ q_{p_2}^k \\ r_{p_2}^k \end{bmatrix}, \begin{bmatrix} x_e^k \\ y_e^k \\ z_e^k \\ v_{x,e}^k \\ v_{y,e}^k \\ v_{z,e}^k \\ \phi_e^k \\ \theta_e^k \\ \psi_e^k \\ p_e^k \\ q_e^k \\ r_e^k \end{bmatrix} \right). \tag{4.9}$$

The evolution of the PEG is given by:

$$\begin{aligned} \mathfrak{f}(\chi^k, \mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e) = \chi^{k+1} = (\mathbf{x}_{p_1}^{k+1}, \mathbf{x}_{p_2}^{k+1}, \mathbf{x}_e^{k+1}) = \\ \Big(f(\mathbf{x}_{p_1}^k, h(\mathbf{u}_{p1}^k)), f(\mathbf{x}_{p_2}^k, h(\mathbf{u}_{p2}^k)), f(\mathbf{x}_e^k, h(\mathbf{u}_e^k)) \Big) \end{aligned} \tag{4.10}$$

with the difference function $\mathbf{x}^{k+1} = f(\mathbf{x}^k, \upsilon^k)$ from equation 7.3.

Information

Attribute Information

It is assumed that each player has a complete attribute information, thus $\pi_i^k = \Pi^k, \forall k \in K, \forall i \in \mathbf{N}$.

State Information

The state information $\iota_i^k \in \iota \subseteq \mathcal{X}$, $\forall k \in \mathbf{K}$ and $\forall i \in \mathbf{N}$ is of type *feedback (perfect state) information*, hence $\iota_i^k = \{\chi^k\}, k \in \mathbf{K}$. For the observation set $\hat{\mathbf{X}}_i^k$ of each player $i \in \mathbf{N}$, $\hat{\mathbf{X}}_i^k = \chi^k$ holds. As a consequence, the observation function is given by $\kappa_i^k(\chi^k) = \chi^k$.

Action Space

The finite and discrete action spaces $\mathbf{U}_i \subset V_x \times V_y \times V_z$ (linear velocity spaces in the body fixed frame of a UAV) of each player $i \in \mathbf{N}$ are constant throughout the game. Using the single-action decomposition, the action sets are defined for a player i as $\mathbf{U}_{ix} = \left(u_{i,u,1} + a\frac{u_{i,u,2} - u_{i,u,1}}{s_{i,u}}, 0, 0 \right)$, $\mathbf{U}_{iy} = \left(0, u_{i,v,1} + b\frac{u_{i,v,2} - u_{i,v,1}}{s_{i,v}}, 0 \right)$, and $\mathbf{U}_{iz} = \left(0, 0, u_{i,w,1} + c\frac{u_{i,w,2} - u_{i,w,1}}{s_{i,w}} \right)$ with $a \in [0, s_{i,u}]_{\mathbb{N}}$, $b \in [0, s_{i,v}]_{\mathbb{N}}$ and $c \in [0, s_{i,w}]_{\mathbb{N}}$ while $(s_{i,u} + 1)$, $(s_{i,v} + 1)$, and $(s_{i,w} + 1)$ are the number of available actions of player i for each decomposed single-action game. Given the maximal velocity vector $\mathbf{v}_i^{max} = (v_{i,x}^{max}, v_{i,y}^{max}, v_{i,z}^{max})^T \in V_x \times V_y \times V_z$ of a player i, the continuous action spaces are given by $[u_{i,u,1}, u_{i,u,2}] = [-\frac{v_{i,x}^{max}}{2}, \frac{v_{i,x}^{max}}{2}]$, $[u_{i,v,1}, u_{i,v,2}] = [-\frac{v_{i,y}^{max}}{2}, \frac{v_{i,y}^{max}}{2}]$ and $[u_{i,w,1}, u_{i,w,2}] = [-\frac{v_{i,z}^{max}}{2}, \frac{v_{i,z}^{max}}{2}]$.

4.2.2 Full Game

Cost Functional

The cost functional of the full game is defined as follows:

$$L(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k) = \sum_{k=1}^{K} \mathrm{P}\left(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k, \chi^k \right) \tag{4.11}$$

with $\mathrm{P}(\cdot)$ given by

$$\mathrm{P}\left(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k, \chi^k \right) = \sum_{i=1}^{2} d(\mathbf{u}_{p_i}^k, \mathbf{u}_e^k, \chi^k) \tag{4.12}$$

and giving an estimation of the costs in stage k if the action tuple $(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k)$ is played. In particular $d(\mathbf{u}_{p_i}^k, \mathbf{u}_e^k, \chi^k)$ provides the change in distance of one pursuer p_i to the evader, while the sum of all $d(\cdot)$'s constitutes the costs in stage k.

Value Function

The value function $V(k, \chi^k)$ is defined as

$$V(k, \chi^k) = \min_{\mathbf{u}_{p1}^k} \min_{\mathbf{u}_{p2}^k} \max_{\mathbf{u}_e^k} L(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k). \tag{4.13}$$

Action Choice

Since it has to be guaranteed that a solution is found, all sub-problems are solved by finding the Nash-equilibrium in mixed strategies. This yields in an optimal probability distribution $\mathbf{q}^{k*}$ over the action space $\mathbf{U}_{p_1} \times \mathbf{U}_{p_2} \times \mathbf{U}_e$ in stage k. In real applications only pure actions can be performed, therefore it has been defined that the optimal action tuple in pure strategies $\mathbf{u}^{k*} = (\mathbf{u}_{p_1}^{k*}, \mathbf{u}_{p_2}^{k*}, \mathbf{u}_e^{k*})$ is determined by choosing the pure actions where the probability distributions $\mathbf{q}^{k*} = (\mathbf{q}_{p_1}^{k*}, \mathbf{q}_{p_2}^{k*}, \mathbf{q}_e^{k*})$ are maximal. With the solution of the value function the optimal actions $\mathbf{u}^{k*}$ are given by

$$\mathbf{u}^{k*} = arg\ V(k, \chi^k), \forall k \in \mathbf{K}. \tag{4.14}$$

The optimal linear reference velocities for the UAV controller are computed by

$$\mathbf{v}_i^{r,k*} = \text{SAT}(\mathbf{v}_i^k + \mathbf{u}_i^{k*}, \mathbf{v}_i^{max}), i \in \mathbf{N} \tag{4.15}$$

while the function $\text{SAT}(\cdot)$ limits the reference velocity vector to the maximum velocity vector.

4.2.3 Multiple Two-Player Game Decomposition

Cost Functional

In the MTPGD game multiple two-player games are solved while at the end the solutions of the distinct games are combined to one overall solu-

tion. Since, a two-pursuer-one-evader game is regarded, the game can be decomposed into two two-player PEGs with following cost functionals:

$$L_1(\mathbf{u}_{p_1}^k, \mathbf{u}_{e'}^k) = \sum_{k=1}^{K} \mathrm{P}\left(\mathbf{u}_{p_1}^k, \mathbf{u}_{e'}^k, \chi^k\right) \tag{4.16}$$

$$L_2(\mathbf{u}_{p_2}^k, \mathbf{u}_{e''}^k) = \sum_{k=1}^{K} \mathrm{P}\left(\mathbf{u}_{p_2}^k, \mathbf{u}_{e''}^k, \chi^k\right) \tag{4.17}$$

with $\mathrm{P}(\cdot)$ being defined as

$$\mathrm{P}\left(\mathbf{u}_{p_i}^k, \mathbf{u}_{e'}^k, \chi^k\right) = t(\mathbf{u}_{p_i}^k, \mathbf{u}_{e'}^k, \chi^k), i \in \mathbf{P} \tag{4.18}$$

giving an estimation of the costs in stage k if the action tuple $(\mathbf{u}_{p_i}^k, \mathbf{u}_{e'}^k)$ is played. In particular $t(\mathbf{u}_{p_i}^k, \mathbf{u}_{e'}^k, \chi^k), i \in \mathbf{P}$ provides the change in distance of the evader to pursuer p_i, while taking into account the actual position of the other pursuer $p_{1\vee 2}$ in stage k. In other words, L_1 provides the payoff for a two-player game between p_1 and e, while p_2's position is taken into account whereas the position of p_2 is assumed to remain static during the actual stage. The same applies to L_2, vice versa. The definition of the distinct cost functionals for this decomposition method turned out to be very crucial, regarding the value of the game and its conformability to the value of the original game. The fusion of all sub-game solutions to one reasonable overall game solution is a highly challenging problem and depends strongly on the regarded application and even the number of pursuers is a critical factor. The analysis of a vast number of simulation results figured out that the proposed method provides results with a high similarity to the full game as described later in this chapter.

Value Function

The value functions are defined as

$$V_1(k, \chi^k) = \min_{\mathbf{u}_{p_1}^k} \max_{\mathbf{u}_{e'}^k} L_1(\mathbf{u}_{p_1}^k, \mathbf{u}_{e'}^k). \tag{4.19}$$

$$V_2(k, \chi^k) = \min_{\mathbf{u}_{p_2}^k} \max_{\mathbf{u}_{e''}^k} L_2(\mathbf{u}_{p_2}^k, \mathbf{u}_{e''}^k). \tag{4.20}$$

Action Choice

As described above the optimal solutions in each stage of the game are calculated by solving both value functions by computing the Nash-equilibrium in mixed strategies for each sub-problem (stage) of the game. Whereas each pursuer gets assigned a single solution $\mathbf{u}_{p_i}^{k*}, \forall i \in \mathbf{P}$ by solving the two value functions from above, the evader has an optimal solution $\mathbf{u}_e^{k*}$ for playing against pursuer p_1 and an optimal solution $\mathbf{u}_{e'}^{k*}$ for playing against pursuer p_2 which have to be combined to one optimal solution. With the solution of the two value functions, the optimal action tuples are given by

$$(\mathbf{u_{p_1}}^{k*}, \mathbf{u_{e'}}^{k*}) = arg\ V_1(k, \chi^k), \forall k \in \mathbf{K}. \tag{4.21}$$

$$(\mathbf{u_{p_2}}^{k*}, \mathbf{u_{e''}}^{k*}) = arg\ V_2(k, \chi^k), \forall k \in \mathbf{K}. \tag{4.22}$$

The optimal strategies describe linear velocities in x, y, z-direction, hence the combination of both optimal evader strategies $\mathbf{u}_{e'}^{k*}$ and $\mathbf{u}_{e''}^{k*}$ can be performed by calculating the vector sum of each optimal strategy vector of e:

$$\mathbf{u_e}^{k*} = \mathbf{u}_{e'}^{k*} + \mathbf{u}_{e''}^{k*}. \tag{4.23}$$

The optimal linear reference velocities for the UAV controller are computed by

$$\mathbf{v}_i^{r,k*} = \text{SAT}(\mathbf{v}_i^k + \mathbf{u}_i^{k*}, \mathbf{v}_i^{max}), i \in \mathbf{N} \tag{4.24}$$

while the function $\text{SAT}(\cdot)$ limits the reference velocity vector to the maximum velocity vector.

4.2.4 Team-Subsumption Two-Player Game

Unlike the MTPGD, in which the cost functional is split into many cost functionals describing many two-player games, the TSTPG requires a entirely different game structure. The basic idea of the TSTPG is that all players of a team are encapsulated into one single super-player, while the cost functional of the game is a combination of two cost functionals defined for each pursuer with the evader. This results in the following new game definition of the PEG ${}^{2}\mathbf{\Pi} = (\mathbf{N}, \mathbf{K}, \mathcal{X}, \mathbf{U}, \mathfrak{f}, \hat{\mathbf{X}}, \kappa, \iota, \mathbf{\Gamma}, L)$ between the three UAV players:

Players

$\mathbf{N} = \mathbf{P} \cup \mathbf{E}$ is the set of all players, while $\mathbf{P} = \{{}^{S}p\}$, with ${}^{S}p = \{p_1, p_2\}$ is a super-pursuer representing two pursuers and $\mathbf{E} = \{e\}$ is the evader.

Termination

$\mathbf{K} = \{1, \ldots, K\}$ is the integer set of stages with K time steps. K is called the terminal time. K is finite if the super-pursuer ${}^{S}p$ catches the evader e and infinite if e manages to escape. e is caught if a pursuer $p_i \in {}^{S}p, i = 1, 2$ reaches a terminal position in $\xi^k \subset X \times Y \times Z$ and thus $(\mathbf{pos}_{p_i}^k, k) \in \mathbf{\Xi}$ holds. ξ^k is defined by a sphere with radius d_ϵ around the evader's position $\mathbf{pos}_e^k = (x_e^k, y_e^k, z_e^k)^T$ in the inertial frame.

State Space

The state space $\mathbf{X}$ of e is given by $\mathbf{X} = X \times Y \times Z \times V_x \times V_y \times V_z \times \Phi \times \Theta \times \Psi \times P \times Q \times R$ with $\mathbf{x}_e^{k \in \mathbf{K}} \in \mathbf{X}$ having the subspaces: position space $X \times Y \times Z$, linear velocity space $V_x \times V_y \times V_z$, Euler angles $\Phi \times \Theta \times \Psi$, and angular velocity space $P \times Q \times R$. For ${}^{S}p$ the state space is given by $\mathbf{X^2}$ with $\mathbf{x}_{{}^{S}p}^{k \in \mathbf{K}} = (\mathbf{x}_{p_1}^{k \in \mathbf{K}}, \mathbf{x}_{p_2}^{k \in \mathbf{K}}) \in \mathbf{X}^2$.

$\mathcal{X} = (\mathbf{X})^3$ is the state space of the game with $\chi^k = (\mathbf{x}^k_{S_p}, \mathbf{x}^k_e) \in \mathcal{X}$. Hence,

$$\chi^k = (\mathbf{x}^k_{S_p}, \mathbf{x}^k_e) = \left(\begin{bmatrix} x^k_{S_p} \\ y^k_{S_p} \\ z^k_{S_p} \\ v^k_{x,S_p} \\ v^k_{y,S_p} \\ v^k_{z,S_p} \\ \phi^k_{S_p} \\ \theta^k_{S_p} \\ \psi^k_{S_p} \\ p^k_{S_p} \\ q^k_{S_p} \\ r^k_{S_p} \end{bmatrix}, \begin{bmatrix} x^k_e \\ y^k_e \\ z^k_e \\ v^k_{x,e} \\ v^k_{y,e} \\ v^k_{z,e} \\ \phi^k_e \\ \theta^k_e \\ \psi^k_e \\ p^k_e \\ q^k_e \\ r^k_e \end{bmatrix} \right), \tag{4.25}$$

with

$$\begin{bmatrix} x^k_{S_p} \\ y^k_{S_p} \\ z^k_{S_p} \\ v^k_{x,S_p} \\ v^k_{y,S_p} \\ v^k_{z,S_p} \\ \phi^k_{S_p} \\ \theta^k_{S_p} \\ \psi^k_{S_p} \\ p^k_{S_p} \\ q^k_{S_p} \\ r^k_{S_p} \end{bmatrix} = \left(\begin{bmatrix} x^k_{p_1} \\ y^k_{p_1} \\ z^k_{p_1} \\ v^k_{x,p_1} \\ v^k_{y,p_1} \\ v^k_{z,p_1} \\ \phi^k_{p_1} \\ \theta^k_{p_1} \\ \psi^k_{p_1} \\ p^k_{p_1} \\ q^k_{p_1} \\ r^k_{p_1} \end{bmatrix}, \begin{bmatrix} x^k_{p_2} \\ y^k_{p_2} \\ z^k_{p_2} \\ v^k_{x,p_2} \\ v^k_{y,p_2} \\ v^k_{z,p_2} \\ \phi^k_{p_2} \\ \theta^k_{p_2} \\ \psi^k_{p_2} \\ p^k_{p_2} \\ q^k_{p_2} \\ r^k_{p_2} \end{bmatrix} \right) \tag{4.26}$$

The evolution of the PEG is given by $\mathfrak{f}(\chi^k, \mathbf{u}^k_{S_p}, \mathbf{u}_e) =$

$$\chi^{k+1} = (\mathbf{x}^{k+1}_{S_p}, \mathbf{x}^{k+1}_e) = \left({}^S f(\mathbf{x}^k_{S_p}, {}^S h(\mathbf{u}^k_{S_p})), f(\mathbf{x}^k_e, h(\mathbf{u}^k_e)) \right), \tag{4.27}$$

while

$${}^S f(\mathbf{x}^k_{S_p}, {}^S h(\mathbf{u}^k_{S_p})) = (f(\mathbf{x}^k_{p_1}, h(\mathbf{u}^k_{p1})), f(\mathbf{x}^k_{p_2}, h(\mathbf{u}^k_{p2})) \tag{4.28}$$

and with the difference function $\mathbf{x}^{k+1} = f(\mathbf{x}^k, \upsilon^k)$ defined above in equation 7.3. The function ${}^{S}h$ is defined as

$$ {}^{S}h : V_x^2 \times V_y^2 \times V_z^2 \to \Upsilon^2, \tag{4.29}$$

yields an input matrix ${}^{\mathbf{S}}\upsilon$ for players p_1 and p_2 given an admissible action $\mathbf{u}_{{}^{S}p}^k$.

Information

The same attribute and state information as defined for the other two cases is assumed.

Action and Strategy Space

Since, the pursuers p_1 and p_2 are represented by a super-player ${}^{S}p$, a strategy of the latter is a mapping $\gamma_{{}^{S}p}^k : \iota_{{}^{S}p}^k \mapsto \mathbf{U}_{p_1} \times \mathbf{U}_{p_2}$, $\forall k \in \mathbf{K}$. The evader's strategies are mappings $\gamma_e^k : \iota_e^k \mapsto \mathbf{U}_e$, $\forall k \in \mathbf{K}$. In this case $\gamma_{{}^{S}p}^k(\iota_i^k) = \mathbf{U}_{p_1} \times \mathbf{U}_{p_2} = \mathbf{U}_{{}^{S}p}$ having the elements $\mathbf{u}_{{}^{S}p} = (\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k)$ and $\gamma_e^k(\iota_e^k) = \mathbf{U}_e^k$ having the elements $\mathbf{u}_e^k$, $\forall k \in \mathbf{K}$ is defined. The exact declaration of the action spaces is the same as proposed above with single-action game decomposition.

Cost Functional

The cost functional defined as follows:

$$ L(\mathbf{u}_{{}^{S}p}^k, \mathbf{u}_e^k) = \mathrm{P}\left(\mathbf{u}_{{}^{S}p}^k, \mathbf{u}_e^k, \chi^k\right) \tag{4.30}$$

with $\mathrm{P}(\cdot)$ defined as

$$ \mathrm{P}\left(\mathbf{u}_{{}^{S}p}^k, \mathbf{u}_e^k, \chi^k\right) = \sum_{k=1}^{K} d(\mathbf{u}_{p_1}^k, \mathbf{u}_e^k, \chi^k) \cdot \sum_{k=1}^{K} d(\mathbf{u}_{p_2}^k, \mathbf{u}_e^k, \chi^k) \tag{4.31}$$

and giving an estimation of the costs in stage k if the action tuple $(\mathbf{u}_{{}^{S}p}^k, \mathbf{u}_e^k)$ is played. In particular $d(\mathbf{u}_{p_i}^k, \mathbf{u}_e^k, \chi^k)$ provides the change in distance of one pursuer p_i to the evader.

Value Function

The value function $V(k, \chi^k)$ is defined as

$$V(k, \chi^k) = \min_{\mathbf{u}^k_{S_p}} \max_{\mathbf{u}^k_e} L(\mathbf{u}^k_{S_p}, \mathbf{u}^k_e). \tag{4.32}$$

Action Choice

Here, too, the solution method described above is used to solve the value function. The present game definition reduces the three-player game to a two player game by representing the pursuer team by a super-pursuer, having an extended strategy space. The solution of the value function yields the optimal actions $\mathbf{u}^{k*}$

$$\mathbf{u}^{k*} = (\mathbf{u}^{k*}_{Sp}, \mathbf{u}^{k*}_e) = arg\ V(k, \chi^k), \forall k \in \mathbf{K}. \tag{4.33}$$

Since, the optimal action of the super-pursuer $\mathbf{u}^{k*}_{Sp}$ is an action combination of p_1 and p_2 the optimal actions can be assigned to $u^{k*}_{p_1}$ and $u^{k*}_{p_2}$.The optimal linear reference velocities for the UAV controller are computed by

$$\mathbf{v}^{r,k*}_i = \text{SAT}(\mathbf{v}^k_i + \mathbf{u}^{k*}_i, \mathbf{v}^{max}_i), i \in \mathbf{N} \tag{4.34}$$

while the function $\text{SAT}(\cdot)$ limits the reference velocity vector to the maximum velocity vector.

4.2.5 Time-Complexity Analysis

Regarding the solution process for one call of NPG for the above defined game structures having a number of N-players, while each player has a number of S admissible actions, an optimal strategy has to be found out of S^N combinations. The full game is transformed to one optimization problem with dimension S^N, the MTPGD game is transformed to N optimization problems with dimension S^2, and the the TSTPG game is transformed to one optimization problem with dimension S^N. Regarding the time complexity of NPG, this means the full game requires $\mathcal{O}(S^{3N})$ time, the MTPGD

$\mathcal{O}(NS^6)$ time, and the TSTPG $\mathcal{O}(S^{3N})$ time to find a solution. While the MTPGD provides a relevant time-complexity reduction with an increasing number of players and strategy combinations, the TSTPG yields no improvement towards the original game using this algorithm. Another big advantage of the MTPGD in opposition to the other approaches is, that N distinct optimization problems have to be solved enabling a parallel computation. In other words, a player with N processors can solve the game requiring $\mathcal{O}(S^6)$ time.

4.2.6 Comparison

In order to compare the proposed multi-player game structures, several numerical simulations have to be carried out. The following assumptions are made for the PEG simulations:

- Each variant is simulated by running simulations with 61×61 different initial values $\chi^1_{x,y}$, with:

$$\chi^1_{x,y} = \left(\begin{bmatrix} x \\ y \\ 0 \\ v^{max}_{x,p_1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ v^{max}_{x,p_2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 30 \\ 30 \\ 0 \\ v^{max}_{x,e} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right), \quad x, y \in [0, 60].$$

- The evader's initial position is always $(30, 30, 0)^T$ and that of pursuer p_2 is always $(0, 0, 0)^T$. Pursuer p_1 starts from varying positions to be able to analyze all different position constellations between all agents.

- The maximum translational velocity of the pursuers is set to $\mathbf{v}^{max}_{p_1} = \mathbf{v}^{max}_{p_2} = (15, 15, 3.5)^T$, and the maximal absolute velocity to $\mathbf{v}^{maxA}_{p_1} = \mathbf{v}^{maxA}_{p_2} = 15$. Also, $\mathbf{v}^{max}_{e} = (10, 10, 2)^T$, and $\mathbf{v}^{maxA}_{e} = 10$, are set for the evader, respectively.

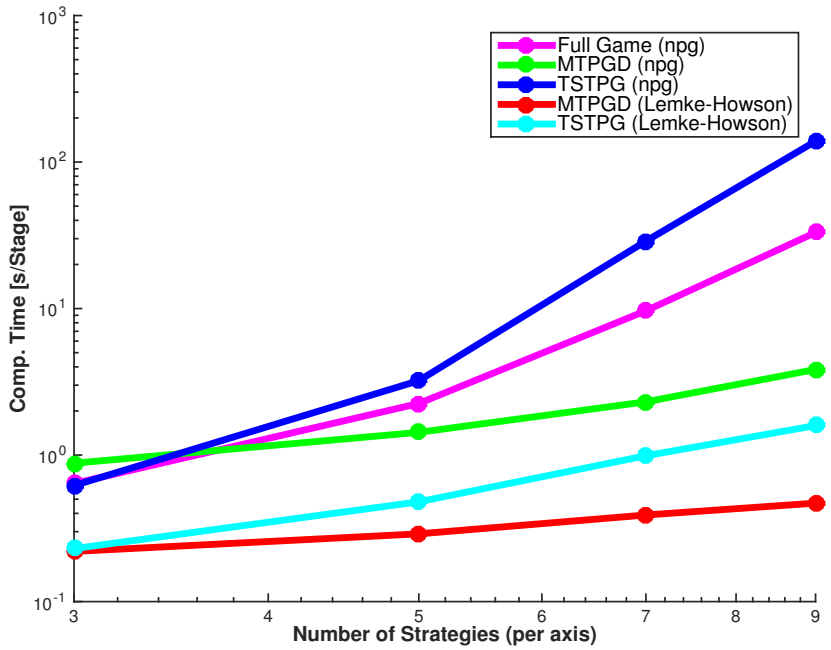

Figure 4.4: *Average runtime as a function of the number of strategies in one stage.*

- The number of discrete actions in one stage for each player is set to $s_{p_1} = s_{p_2} = s_e = s = 27$, 125, 343, and 729, resulting in four simulation series with each having 3721 game simulations.
- The stage duration of the PEG is set to $\Delta T = 0.1$ while the velocity and attitude controller are sampled with $\Delta t = 5e - 3$.
- The capture radius is set to $d_\epsilon = 5$.

The results revealed that each game converged in all three game variants. Figure 2 depicts the average runtime of each of the four simulation series for each variant as a function of the strategy options of each player in one stage of the game. The total number of simulated games is 44652. To analyze the

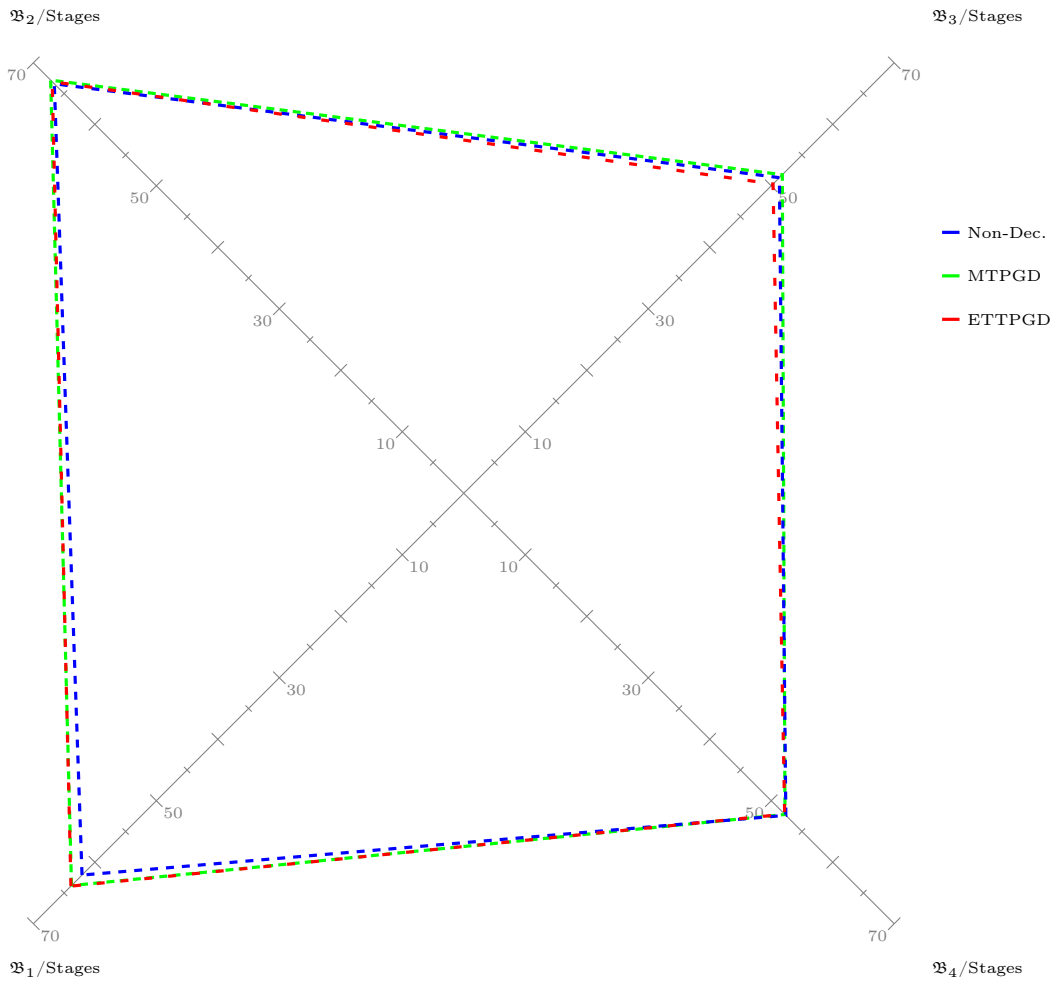

Figure 4.5: *Average of terminal stages in each quadrant.*

results, the position space $\mathfrak{E}^j$ is divided into quadrants $\mathfrak{B}_1^j, \ldots, \mathfrak{B}_4^j$:

$$\begin{aligned}
\mathfrak{B}_1 &= \left\{\chi_{x,y}^1 | x, y \in [0, 30]\right\} \\
\mathfrak{B}_2 &= \left\{\chi_{x,y}^1 | x \in [0, 30], y \in [31, 60]\right\} \\
\mathfrak{B}_3 &= \left\{\chi_{x,y}^1 | x, y \in [31, 60]\right\} \\
\mathfrak{B}_4 &= \left\{\chi_{x,y}^1 | x \in [31, 60], y \in [0, 30]\right\} \\
\mathfrak{E}^j &= \mathfrak{B}_1 \cup \cdots \cup \mathfrak{B}_4
\end{aligned}$$

Regarding the average run-time of all simulation series in figure 4.4, the results reflect mostly the time complexity analysis in the last section. While the

original game reveals a shorter run-time in respect to TSTPG with increasing strategy space, the MTPGD, otherwise, enables a calculation equally fast for $s = 3$, ≈ 1.57 times faster for $s = 5$, ≈ 4.2 times faster for $s = 7$, and ≈ 8.6 times faster for $s = 9$, using the npg algorithm. The run-time of the TSTPG and the MTPGD can be reduced even more, when using the Lemke-Howson algorithm, which is only available for two-player games. Thus, for the TSTPG solution a calculation ≈ 2.8 times faster for $s = 3$, ≈ 4.7 times faster for $s = 5$, ≈ 9.75 times faster for $s = 7$, and ≈ 20.79 times faster for $s = 9$. For the MTPGD solution a calculation ≈ 2.9 times faster for $s = 3$, ≈ 7.75 times faster for $s = 5$, ≈ 24.77 times faster for $s = 7$, and ≈ 70.36 times faster for $s = 9$.

Figure 4.5 depicts the average of terminal stages in each quadrant of the position space. It can be seen that all three approaches yield almost identical average values in each quadrant ($< |2|$ stages). The standard deviation in $\mathfrak{B}_1$, $\mathfrak{B}_2$, $\mathfrak{B}_3$ and $\mathfrak{B}_4$ of the difference of the MTPGD and the full game are given by 2.23 stages ($\equiv 0.223s$ capture time), 6.86 stages ($\equiv 0.686s$ capture time), 2.34 stages ($\equiv 0.234s$ capture time), and 11.83 stages ($\equiv 1.183s$ capture time). The results total up to an average of 0.66 stages difference and a standard deviation of 6.81 stages ($\equiv 0.681s$ capture time). Due to the decomposition process, a portion of information gets lost in the distinct 2-Player games, leading to these deviations.

The difference of the TSTPG to the full game sums up to an average value of 0.23 stages and a standard deviation of 3.53 stages ($\equiv 0.353s$ capture time). The average value's standard deviation in $\mathfrak{B}_1$, $\mathfrak{B}_2$, $\mathfrak{B}_3$ and $\mathfrak{B}_4$ are given by 1.44 stages ($\equiv 0.144s$ capture time), 2.33 stages ($\equiv 0.233s$ capture time), 2.06 stages ($\equiv 0.206s$ capture time), and 5.97 stages ($\equiv 0.597s$ capture time).

Despite the fact that the average of the difference of the terminal stage values is almost zero, a standard deviation of up to 11.83 stages in $\mathfrak{B}_4$ could be observed. The results indicates, that in some cases the strategies computed with the MTPGD differ from those of the full game. Due to the fact that the pursuers are regarded separately, the evader considers only the strategy option of each antagonist independently. In opposition to the single-action game decomposition, where no qualitative detriment regarding the solution is observed, the MTPGD differs from the original solution under specific circumstances. The fusion of the distinct sub-game solutions is very cru-

cial. Although the property of game theory, that the solution of a game depends on the decision of each involved player, is not violated, the solution of the regarded game is no longer a pure three-player game solution. It is a composition of two distinct two-player games. The main advantage of this approach is the computational time gain. Figure 4.4 depicts the average runtime of each game, starting at the above defined initial positions, as a function of the strategy options each player has in one stage of the game per axis. With increasing strategy space the TSTPG has even a bigger run-time as the full game. It can also be seen, that the run-time of the MTPGD is considerably smaller with increasing strategy space. Since the two new structures regard only two-player games, more sophisticated algorithms, only available for two-player games, can be used. As depicted on figure 4.4, the Lemke-Howson algorithm gives an significant reduction to the run-time of the TSTPG and MTPGD. The TSTPG solution is even faster calculated with the Lemke-Howson algorithm then the MTPGD solution with the npg algorithm. However, the fastest combination is constituted by the MTPGD and the Lemke-Howson algorithm. Due to the tremendous time-complexity reduction of the game with the MTPGD, this approach yields an acceptable trade-off between time-complexity and solution quality.

4.3 Conclusion

First, the decomposition of the action sets of each player in a game is shown, resulting in three distinct games. It is presented that the results of both methods yield close results, while the time-complexity could be reduced markably. Moreover, two different game structures for multi-player dynamic games are introduced and applied to a two-pursuer-one-evader pursuit-evasion game. It is shown that the MTPGD provides a game-theoretical solution approach for a multi-player game that considerably reduces the run-time of the solution algorithm. Both, the MTPGD and TSTPG approach provided close results to the original game. In addition, the MTPGD provides a solution approach which enables a parallel computation of all distinct two-player games. That means, with N processors, the run-time can be improved by a factor of N. Both methods enable the usage of the Lemke-Howson algorithm for the solution of the game, which is a more sophisticated algorithm

only available for two-player games. In summary, by the decomposition of the action space and the usage of the MTPGD game solved with the Lemke-Howson algorithm, the run-time of a game with 3 strategies per axis, per player and per stage, can be improved by a factor of ≈ 5.8, with 5 strategies by a factor of ≈ 141.8, with 7 by a factor of ≈ 1847.3 and with 9 strategies by a factor of ≈ 12043.52. For the usage of the TSTPG solved with the Lemke-Howson algorithm the run-time of a game with 3 strategies per axis, per player and per stage, can be improved by a factor of ≈ 5.6, with 5 strategies by a factor of ≈ 86, with 7 by a factor of $\approx 727,16$ and with 9 strategies by a factor of ≈ 3558.62. Since, [ASB14b; ASB16] proved that a real-time implementation of a two-player PEG with two identical UAVs is applicable, especially the MTPGD enables the real-time applicability for PEGs with multiple UAVs.

5 Cooperation and Behavior Assignment

In this chapter a solution concept for general multi-team games is presented. It is aimed to define a solution concept that allows a superordinate cooperation between team-members. Since this work focuses on PEGs with UAVs, this class of games is used to verify the presented solution concept. In respect to two-player pursuit-evasion games, multi-player pursuit-evasion games with an arbitrary number of players are much more complex. The game changes from a pure non-cooperative into a cooperative one between the players within a team and to a non-cooperative one between the parties. Furthermore, a certain degree of flexibility for the team's strategies is desired. In other words, the teams should be able to negotiate about different behavioral strategies and assign them to each team-mate. This negotiation process can be defined as a superordinate cooperative game.

The problem which is treated in this chapter has been firstly stated in [AB12], briefly: *Is a team of pursuers able to catch a faster evader by cooperation in a visibility-based PEG with imperfect information?* The solution approach described in [AB12] includes the use of a cascaded system structure for the UAV agents, similar to that depicted in figure 5.4. Due to a communication channel between the pursuers, they are able to maximize their information set. This can be used for a battue against the evader and to contain or ambush the evader, which requires that the pursuers are able to change their behavior from time to time. Therefore, two behaviors, the pursue and the battue behavior will be introduced. In [AB12] an urban environment is assumed, which will be neglected in this chapter. In order to move in an en-

vironment with many obstacles, a collision avoidance behavior as introduced in [ASB14a] for PEG is required. The PEG in an environment with moving and static obstacles will be treated in chapter 7.6.

Other than stated in [AB12], the PEG will be separated into two levels, the cooperative and non-cooperative level. This gives a higher flexibility to the cooperating teams. The system structure of each player in a perfect information PEG could then be defined as depicted in figure 5.1. Due to the perfect state information, each player has full knowledge about the game state in each time step. This system structure is an analogical representation of a RNBC-structure realization, which is introduced in [ASB14a] for a non-cooperative two player PEG as a preliminary work and will be presented in the next chapter in detail.

This structure is extended by adding the "Cooperative Behavior Assignment" layer on top, analogical to figure 5.1. The implementation of this structure is done in [Sch14] where two-pursuer one-evader PEGs with UAVs moving on a 2-D plane with faster pursuers in games with perfect state information structure are analyzed. The main tasks of this work are, amongst others, on the one hand to find out, which game-theoretical cooperation approach yields the best results for the pursuers, and on the other hand whether a cooperation yields a gain in the pursuer's outcome. This implementation is used in this work and extended to 3D motion and to games with imperfect information and delayed information sharing.

The results of the multi-player PEG with UAVs and cooperating pursuers moving in a 3D environment in PEGs with perfect state information, presented in this work, have been published in [ASB15].

5.1 Superordinate Cooperation in PEGs

The two-pursuer one-evader game defined in 4.2.1 is extended by a superordinate cooperation level enabling multiple pursuers and evaders to team up. In simulations it will be shown, that the outcome of the pursuers in a two-pursuer one-evader PEG can be enhanced by cooperation. The pursuers are able to negotiate which pursuing behavior (*pursuit* or *battue*) is assigned to them by applying game-theoretical solution methods for cooperative games.

The two pursuers one-evader problem stated above with UAV agents with identical dynamical constraints has to be solved assuming that:

- The 3-D environment is unbounded and without obstacles.
- The evader is slower than the pursuers.
- All players have perfect state information and complete attribute information.

A solution to this problem is sought that fulfils following requirements:

- All agents have to consider that the solution of the problem depends on the decisions of each participant.
- The players must be able to react to unexpected behavior (closed-loop solution).
- The pursuers have to cooperate.

Therefore, following solution approach is proposed:

- Definition of a superordinate cooperative game between the pursuers enabling a strategical behavior change, solved with game-theoretical methods (*Nash Equilibrium* [Nas50a], *Nash Bargaining Solution* [Nas53], *Pareto Efficiency* [McL08]).
- Game-theoretical solution approach for the subordinate non-cooperative game.

Figure 5.1 illustrates the general structure of the players (in this case only the pursuers). In the cooperative game set-up a behavioral strategy is assigned by solving a static normal-form cooperative game between the pursuers. Each player has two admissible behavioral strategies, which are *pursuit* and *battue*. The payoff matrix of this game is constituted as defined in table 5.1. A payoff tuple $({}^{\mathfrak{K}}L_1^{b*}, {}^{\mathfrak{K}}L_2^{b*})$ denotes the optimal payoff of playing the non-cooperative game against the evader for a duration of $\mathfrak{K}$ stages, while acting according to the behavioral strategy $\mathbf{b} \in \{p, b\}$, while p stands for *pursuit* and b for *battue*, respectively. The sub-game solutions of the non-cooperative

Behavioral Strategy	*pursuit*	*battue*
pursuit	$({}^{\mathfrak{K}}L_1^{p*}, {}^{\mathfrak{K}}L_2^{p*})$	$({}^{\mathfrak{K}}L_1^{p*}, {}^{\mathfrak{K}}L_2^{b*})$
battue	$({}^{\mathfrak{K}}L_1^{b*}, {}^{\mathfrak{K}}L_2^{p*})$	$({}^{\mathfrak{K}}L_1^{b*}, {}^{\mathfrak{K}}L_2^{b*})$

Table 5.1: *Static cooperative game for behavior assignment.*

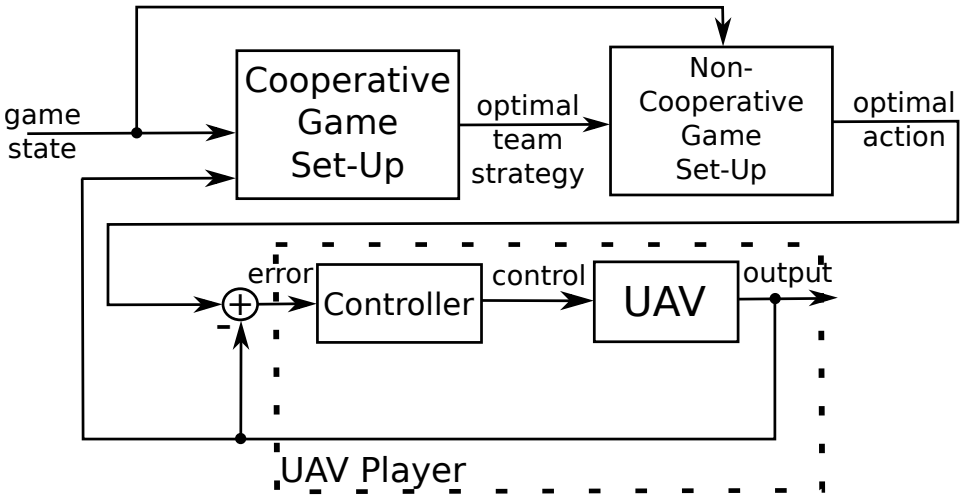

Figure 5.1: *Player's structure in game with perfect state information.*

game can be memorized to avoid a recalculation. The solution of that game, yields an optimal team strategy. Now that each pursuer knows its optimal behavior, the perviously calculated optimal actions for the non-cooperative game can be recalled, if and only if the evader plays as expected, i.e. optimal. If the evader doesn't play optimal, the behaviors are kept anyway for the duration of $\mathfrak{K}$ stages. So the sub-game solutions of the non-cooperative game have to be calculated for the actual, unexpected, state of the game. The sub-game solutions for each stage of the non-cooperative game, i.e the optimal actions, are passed to the UAV controller and executed by the UAV system. Up to this point, a perfect state information is assumed, thus the adversary state estimation is a perfect state observer with no time delay, just like the team-information sharing has no time delay in such a game. Later, it will be shown, how an imperfect state information with delayed observation sharing affects the solution of the game.

5.2 Team-Behavior Game

As stated before, the pursuers are able to chance their behavior within a given time horizon of $\mathfrak{K}$ stages. The two behaviors are *pursuit* and *battue*. In the following it is described which cost functionals describe the behavior of the pursuers within the PEG.

5.2.1 Battue and Pursuit

Assuming that the evader is permanently maximizing the distance to the pursuers, the latter can either minimize the distance (*pursuit*) to the evader or follow the so called battue point (*battue*). To understand how the battue point is determined the situation as depicted on figure 5.2 is regarded (blue pursuer is kept at a constant position for demonstration purposes). The green pursuer is supposed to drive the evader towards the blue one. Therefore, a line through the blue pursuer's position $\mathbf{pos}_{p'}^{k+1}$ and the evader's position $\mathbf{pos}_{e}^{k+1}$ is set. The battue point is a point on this line with a specific distance to the evader's position in opposite direction to the blue pursuer. Pursuing the battue point results in driving the evader towards a teammate. The battue point is calculated by

$$\mathbf{bp}_p = \mathbf{pos}_{p'}^{k+1} + \xi \cdot (\mathbf{pos}_{e}^{k+1} - \mathbf{pos}_{p'}^{k+1}) \tag{5.1}$$

with

$$\xi = \frac{\|\mathbf{pos}_{p'}^{k+1} - \mathbf{pos}_{p}^{k+1}\|^2 + \|\mathbf{pos}_{e}^{k+1} - \mathbf{pos}_{p'}^{k+1}\|^2}{\|\mathbf{pos}_{e}^{k+1} - \mathbf{pos}_{p'}^{k+1}\| \cdot \|\mathbf{pos}_{p'}^{k+1} - \mathbf{pos}_{p}^{k+1}\|} - c. \tag{5.2}$$

The constant c in the latter equation affects the distance of the battue point to the evader. With a value of $c = 0.9$ the best results could be achieved, while this value is determined empirically. If both pursuers play *pursue* the cost functional is given by:

$$L(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_{e}^k) = \sum_{k=1}^{K} \mathrm{P}\left(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_{e}^k, \chi^k\right) \tag{5.3}$$

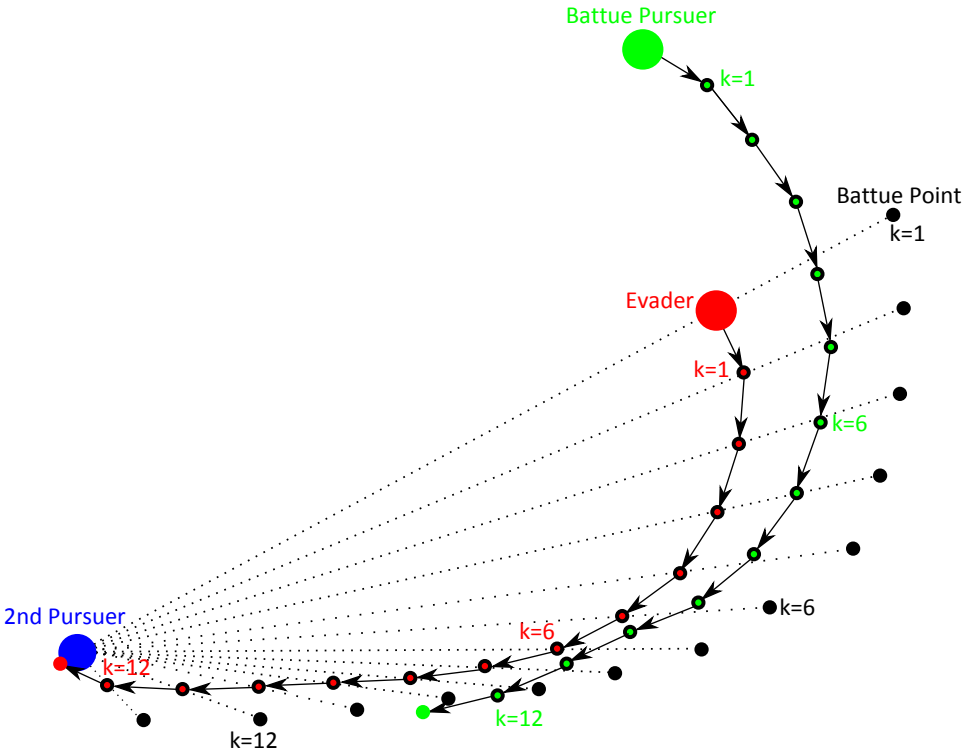

Figure 5.2: *Green pursuer drives evader (red) towards blue pursuer.*

with $\mathrm{P}(\cdot)$ being defined as

$$\mathrm{P}\left(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k, \chi^k\right) = \sum_{i=1}^{2} d(\mathbf{u}_{p_i}^k, \mathbf{u}_e^k, \chi^k) \tag{5.4}$$

and giving an estimation of the costs in stage k if the action tuple $(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k)$ is played. In particular $d(\mathbf{u}_{p_i}^k, \mathbf{u}_e^k, \chi^k)$ provides the change in distance of one pursuer p_i to the evader, while the sum of all $d(\cdot)$'s constitutes the costs in stage k. If a pursuer plays *battue* its cost functional is given by:

$$L(\mathbf{u}_{p_1}^k, \mathbf{u}_{p_2}^k, \mathbf{u}_e^k) = \sum_{k=1}^{K} \mathrm{H}(\chi^{k+1}, \mathbf{u}_p, \chi^k) = \sum_{k=1}^{K} \|\mathbf{bp}_p - \mathbf{pos}_p^{k+1}\|, \tag{5.5}$$

while the outcome of the remaining players results from $d(\mathbf{u}_{p_i}^k, \mathbf{u}_e^k, \chi^k)$.

To make a choice which behavior is assigned to the pursuers for the regarded time horizon, a *static two-player cooperative game* is solved with the outcome

$$\eta(\sigma) = (\|\mathbf{pos}_{e,\sigma}^{k+\kappa} - \mathbf{pos}_{p_1,\sigma}^{k+\kappa}\|, \|\mathbf{pos}_{e,\sigma}^{k+\kappa} - \mathbf{pos}_{p_2,\sigma}^{k+\kappa}\|), \tag{5.6}$$

while $\mathbf{pos}_{i,\sigma}^{k+\kappa}$ is the position of player $i \in \mathbf{N}$ of the PEGs in stage $k+\kappa$, if the pursuers played in the stages $[k, k+\kappa]$ the behavioral strategy combination $\sigma = (\sigma_{p_1}, \sigma_{p_2}) \in \Sigma \times \Sigma$, with $\Sigma = p, h$ being the behavioral strategy space with the elements p for *pursuit* and h for *battue*. This results in a normal-form game with the strategies *pursuit* and *battue* for each player, and the outcomes $\eta(\sigma)$, while both players want to minimize their outcome. The solution of this game provides a behavioral strategy combination, which is applied by the pursuers in the subsequent stages. This problem is solved by using different game-theoretical approaches to determine which one yields the best results. Therefore, the *Nash Equilibrium*, the *Nash Bargaining Solution*, and the *Pareto Efficiency* are utilized. To evaluate the simulation results, two questions have to be answered:

- Do cooperating pursuers achieve better results?
- Which solution approach for the cooperative game yields the best results?

Therefore, the solutions of the cooperative game with four different solution variants have to be compared, namely the *Nash Equilibrium* (Variant 1), *Nash Bargaining Solution* (Variant 2), *Pareto Efficiency* (Variant 3), and no cooperative game is played while each pursuer minimizes the distance to the evader (Variant 4).

5.2.2 Comparison for Games with Perfect State Information

The following assumptions are made for the PEG simulations:

- Each variant is simulated by running simulations with 61×61 different initial values $\chi_{x,y}^1$, with:

$$\chi^1_{x,y} = \left(\begin{bmatrix} x \\ y \\ -2 \\ v^{max}_{x,p_1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -2 \\ v^{max}_{x,p_2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 30 \\ 30 \\ -2 \\ v^{max}_{x,e} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right), \quad x, y \in [0, 60].$$

- The evader's initial position is always $(30, 30, -2)^T$ and that of pursuer p_2 is always $(0, 0, -2)^T$. Pursuer p_1 starts from varying positions to be able to analyze all different position constellations between all agents.
- The maximum translational velocity of the pursuers is set to $\mathbf{v}^{max}_{p_1} = \mathbf{v}^{max}_{p_2} = (15, 15, 3.5)^T$, and the maximal absolute velocity to $\mathbf{v}^{maxA}_{p_1} = \mathbf{v}^{maxA}_{p_2} = 15$. Also, $\mathbf{v}^{max}_{e} = (12, 12, 2)^T$, and $\mathbf{v}^{maxA}_{e} = 12$, are set for the evader, respectively.
- The number of different discrete actions for each player is set to $s_{p_1} = s_{p_2} = s_e = 9$, while $s_{p_1,u} = s_{p_2,u} = s_{e,u} = s_{p_1,v} = s_{p_2,v} = s_{e,v} = 2$, and $s_{p_1,w} = s_{p_2,w} = s_{e,w} = 1$.
- The stage duration of the PEG is set to $\Delta T = 0,1$ while the velocity and attitude controller are sampled with $\Delta t = 5e - 3$.
- The stage duration for the superordinate cooperative game is set to $10 \cdot \Delta T$.
- The capture radius is set to $d_\epsilon = 5$.

The analyze the results, the position space $\mathfrak{E}^j$ is divided into quadrants $\mathfrak{B}^j_1, \ldots, \mathfrak{B}^j_4$:

$$\begin{aligned} \mathfrak{B}_1 &= \left\{ \chi^1_{x,y} | x, y \in [0, 30] \right\} \\ \mathfrak{B}_2 &= \left\{ \chi^1_{x,y} | x \in [0, 30], y \in [31, 60] \right\} \\ \mathfrak{B}_3 &= \left\{ \chi^1_{x,y} | x, y \in [31, 60] \right\} \\ \mathfrak{B}_4 &= \left\{ \chi^1_{x,y} | x \in [31, 60], y \in [0, 30] \right\} \\ \mathfrak{E}^j &= \mathfrak{B}_1 \cup \cdots \cup \mathfrak{B}_4 \end{aligned}$$

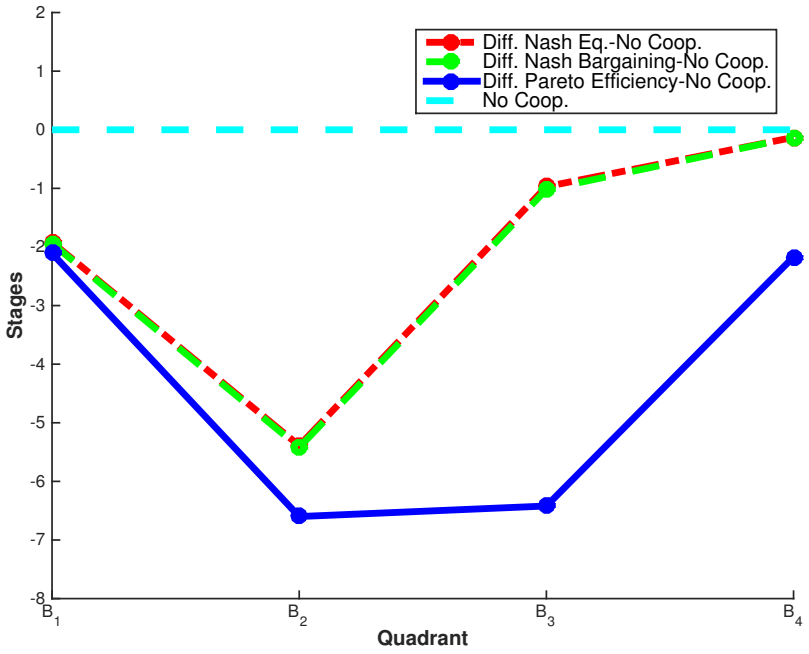

Figure 5.3: *Difference of average terminal stages of coop. and non.-coop solution.*

Figure 4.5 depicts how many stages are needed to capture the evader in average for each quadrant in respect to the game with no cooperation. While the solution derived with the *Nash Equilibrium* and the *Nash Bargaining Solution* are almost equal, the solution derived with the *Pareto Efficiency* points up a significant improvement regarding the terminal stage, especially in respect to the solution without cooperation and thus without behavior assignment. Taking a closer look at figure 4.5 reveals, that the initial positions of all players are affecting the solution considerably as documented in table 5.2, 5.3, 5.4, and 5.5. Most notably in $\mathfrak{B}_3$, where the two pursuers "surround" the evader, the *Pareto Efficiency* solution yields an average improvement of 13.25% in opposition to the non-cooperative solution. These two facts are an important indication for a good pursuing strategy: If the pursuers surround the evader, playing the cooperative game with the *Pareto Efficiency* approach will yield a remarkable improvement in capture time.

Method	*Average of Stages*	*Improvement*
Nash Equilibrium	62.78	3.54%
Nash Bargaining Solution	62.76	3.59%
Pareto Efficiency	62.59	3.37%
No Cooperation	64.70	–

Table 5.2: *Average of terminal stages in* $\mathfrak{B}_1$.

Method	*Average of Stages*	*Improvement*
Nash Equilibrium	67.30	8.00%
Nash Bargaining Solution	67.27	8.05%
Pareto Efficiency	66.09	9.98%
No Cooperation	72.69	–

Table 5.3: *Average of terminal stages in* $\mathfrak{B}_2$.

Method	*Average of Stages*	*Improvement*
Nash Equilibrium	53.90	1.79%
Nash Bargaining Solution	53.85	1.89%
Pareto Efficiency	48.45	13.25%
No Cooperation	54.87	–

Table 5.4: *Average of terminal stages in* $\mathfrak{B}_3$.

Method	*Average of Stages*	*Improvement*
Nash Equilibrium	53.27	0.24%
Nash Bargaining Solution	53.27	0.24%
Pareto Efficiency	51.23	4.23%
No Cooperation	53.40	–

Table 5.5: *Average of terminal stages in* $\mathfrak{B}_4$.

5.3 Team-Behavior Game with Delayed Observation Sharing and Imperfect Information

In general, a perfect state information is unrealistic. Perfect state information signifies that each player is able to observe the whole state space in each time step of a game with an instantaneous perfect state observer. Regarding

real applications, like *search and rescue missions* or *air combat maneuvering*, an agent $i \in \tau$ is only able to estimate its own states and the state of others within its observable space, i.e. observable set $\hat{X}_i^k \subset X$ in stage k with the help of a set of arbitrary sensors and filters/observers as depicted in figure 5.4. The observable set of player i in stage k results directly from the sensing area of player i's sensors. There exists an observation function $\kappa : X \to \hat{X}_i^k$, with

$$\hat{\mathbf{x}}_i^k = \kappa(\mathbf{x}^\mathbf{k}), \tag{5.7}$$

giving an estimation of the state of the game $\hat{\mathbf{x}}^k$, $\forall k \in \mathbf{K}$. Due to a physically limited rate of sensing and due to information processing the state estimations are available after a certain constant time delay of α stages. Certainly each real sensor has a physically limited sensing range. Hence, each agent can only observe its own state vector, and the state vector of each agent within its sensing area. This opens the door for observation sharing within a team of agents. It is assumed that a secure data link between all $m \in \mathbf{P}_m$, $\forall m \in \{1, 2, \ldots, M\}$ and between all $l \in \mathbf{E}_l$, $\forall l \in \{1, 2, \ldots, L\}$ exists. The transmission time β stages for a full information exchange between all players within a team is assumed to be constant. All in all, in real applications each player i gathers as much information as possible, resulting generally in an β-step delayed imperfect state information $\iota_i^k = \hat{\mathbf{x}}_i^{k-\beta}$. Moreover, i is a member of a team $\mathbf{Q} \subset \tau$, while i is able to share its state information ι_i^k, with all $q \in \mathbf{Q}$, $\forall q \neq i$ and vice versa. In particular, the state information of player i results in:

$${}^{OS}\iota_i^k = \{\iota_i^k, \iota_l^{k-\beta}, \ldots, \iota_m^{k-\beta}\}, \tag{5.8}$$

while $k \in \mathbf{K}$, $o = \{m, \ldots, l\}$, and $o \cup i = \mathbf{J}$.

Regarding figure 5.4, the control structure of one UAV player i in a multiplayer PEG is depicted. The sensors and filters are estimating on the one hand the state of player i, and on the other hand an adversary state estimator is estimating the states of each foe within the observable set of player i. The estimation of those states takes α stages to be completed. In a next step, this information is shared with all teammates, which takes another β steps. At this point, each player within a team knows the states of all his teammates and the states of all foes within an observable set of each teammate $\alpha+\beta$ steps

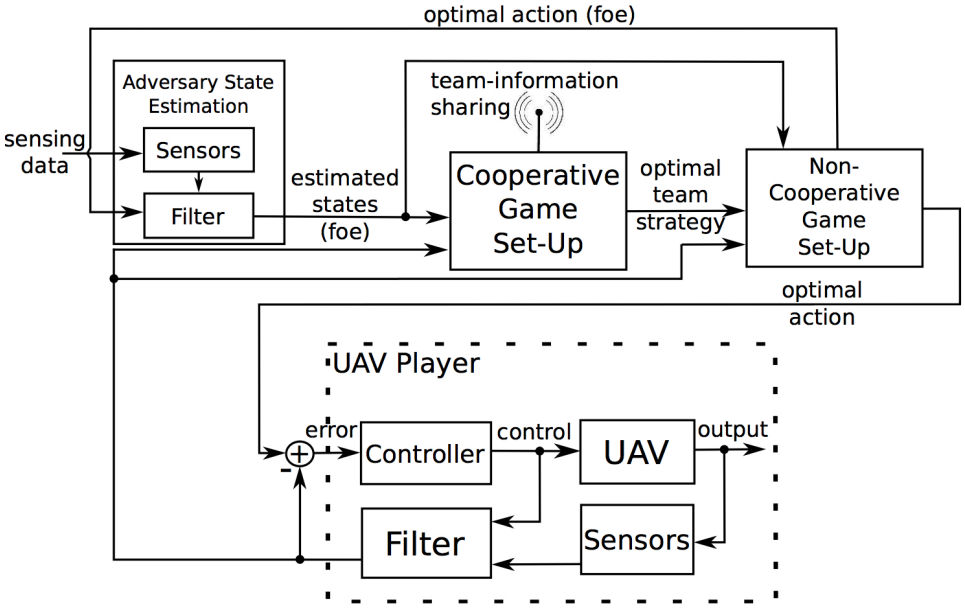

Figure 5.4: *Player's structure in a game with observation sharing.*

ago. Based on that information an estimate of the actual position of each observable player can be conducted and the cooperative game can be player to obtain an optimal behavioral strategy, based on the observable game state $\hat{\mathbf{x}}^k$ of the regarded team. This optimal solution is used to determine the non-cooperative game set-up, and estimate the optimal actions of each UAV.

Due to the high complexity of obtaining a game-theoretical solution the absolute state estimation time is in general much smaller then the stage duration Δt of one stage k. Thus, α is assumed to be 0. Vice versa, a secure team-information sharing can be assumed to generate more overhead. Hence, it is assumed that β is a multiple of k, with $\beta = r \cdot k$, $r \geq 0$, and $r \in \mathbb{N}$.

The delayed observation sharing with imperfect information as presented above seems to be very close to reality. This type of games, enables a large set of opportunities for strategical behavior in PEGs. Regarding the two-player PEG from above, the results reveal that the game converges for each initial state of the game. It can be said, that a PEG between two players having the same dynamics with perfect state information converges for each initial state, as long as the pursuer is faster than the evader. The other

way around, it holds that such a game will never converge if the pursuer is slower or as fast as the evader. The same holds if the number of pursuers is increased by a plausible quantity. If the evader is faster, there will exist an arbitrary number of escape paths.

The two-pursuer one-evader game with cooperating pursuers and delayed observation sharing, represents a more realistic set-up for PEGs. Now, it is conceivable that the pursuers gain a big advantage in opposition to the evader. In other words, if one pursuer is able to observe the evader, the second pursuer will be informed about the evaders position while the evader might not even know about the presence of a second pursuer. This circumstance opens the door for many different pursuing strategies, e.g. ambushing the evader and others. The battue behavior described before makes more sense in such a configuration then in a set-up with perfect state information. Although, as illustrated in figure 5.3, the results of the cooperating pursuers reveal an improvement in capture time, the main contribution of this approach is to establish a general framework for dynamic games with cooperative and non-cooperative nature, giving the ability of switching the general behavior of each player from time to time to improve the outcome. It is expected that the key advantage of the pursuer's battue strategy comes to light in multi-pursuer one-evader games with imperfect state information and a faster evader.

5.3.1 PEGs with Zero-Delay Observation Sharing

To verify the latter, a simulations series with a similar set-up for the two-pursuer one-evader game with cooperating pursuers is chosen. The evader's initial position is changed to $(15, 15, 0)$ and the second pursuers initial position to $(3, 3, 0)$. Moreover, an observation set $\hat{X}_i^k$, which contains all positions in a sphere around player i's position in the 3D-Euclidean space with sensing radius $r_S = 15 \cdot \sqrt{2}$, $i \in \mathbf{N}$ and $\forall k \in \mathbf{K}$ is assumed. This means, that initially at least one pursuer is observing the evader and vice versa as depicted on figure 5.5. From this point on, only the *Pareto Efficiency* approach will be used for the cooperative game between the pursuers since it produced the best results. In this simulation the pursuers are able to share their game states with their team-mates without delay. At this point, no

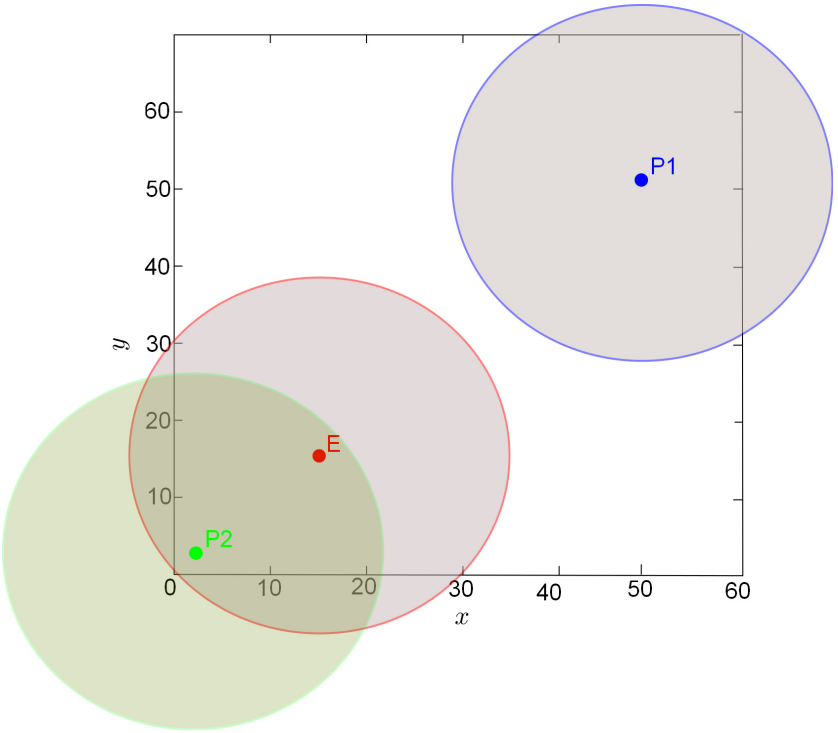

Figure 5.5: *Imperfect information: initial positions and sensing radii.*

delay is assumed to enable a more precise analysis of the effect of a delayed state information, which is done in the next section of this chapter.

A comparison of the results of the perfect state information game and the imperfect information game with an evader being 20% slower than the evaders reveals the advantage of the pursuing team (figure 5.6). Table 5.6, 5.7, 5.8, and 5.9 underpins that by showing the percentage improvement in capture time in all four quadrants. Especially, when the first pursuer starts in quadrant $\mathfrak{B}_2$ or $\mathfrak{B}_4$ the improvement in capture time is more than 40%.

Since in the perfect state information game the evader "sees" both pursuer coming, it is able to act from the beginning against the incoming "threat". In case of the imperfect information game, the evader knows initially only the first pursuer's state and does not even know that there is another pursuer on its way to capture it. The evader knows about the presence of the second pursuer not until it undercuts its sensing radius r_S.

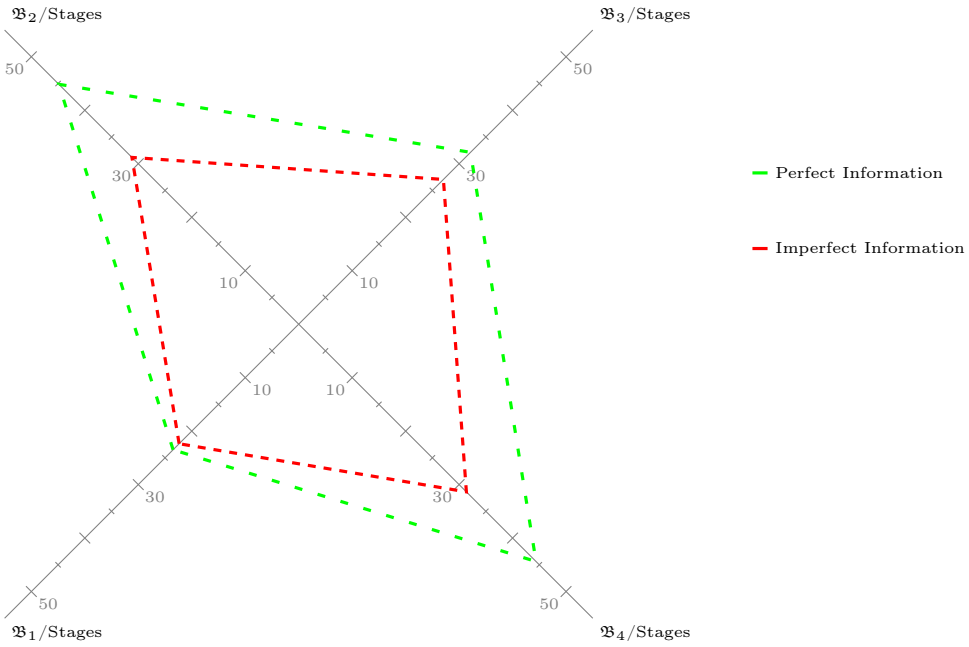

Figure 5.6: *Average of terminal stages in each quadrant (evader* 20% *slower).*

Information	*Average of Stages*	*Improvement*
Perfect	23.49	–
Imperfect	22.37	5.00%

Table 5.6: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_1$ *(evader* 20% *slower).*

Information	*Average of Stages*	*Improvement*
Perfect	44.86	–
Imperfect	31.18	43.87%

Table 5.7: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_2$ *(evader* 20% *slower).*

In the upcoming simulation series it is analyzed how the results change, when the speed of the evader is increased. First, the evaders maximal speed is raised to 90% of the maximal speed of the pursuers. The results are depicted on figure 5.7 and a comparison of the simulation results of the perfect

Information	*Average of Stages*	*Improvement*
Perfect	32.13	–
Imperfect	27.05	18.78%

Table 5.8: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_3$ *(evader* 20% *slower).*

Information	*Average of Stages*	*Improvement*
Perfect	44.38	–
Imperfect	31.36	41.51%

Table 5.9: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_4$ *(evader* 20% *slower).*

and imperfect game cases is given in table 5.10, 5.11, 5.12, and 5.13. As one can see, especially in quadrant $\mathfrak{B}_2$, $\mathfrak{B}_3$, and $\mathfrak{B}_4$ the improvement is much bigger than before although the evader's speed is higher. At a first glance this may seem paradox but there is a simple explanation. The initial situation where the pursuers do not start side by side, which is particularly when the first pursuer starts not in quadrant $\mathfrak{B}_1$, and moreover the initial positions where the first pursuer is not in the sensing radius of the evader, bring out the real benefits of the cooperative approach presented above. In most cases (especially in $\mathfrak{B}_2$, $\mathfrak{B}_3$, and $\mathfrak{B}_4$) the pursuers' choice of the behavioral strategies result in an ambushing approach. The pursuer, which is sensed by the evader, drives the evader towards the other pursuer. Optimally, this brings the evader on a direct course to the second pursuer, that is also approaching the evader. When entering the sensing radius, the evader has almost no chance of escaping the approaching pursuer. This situation also explains, why the increased speed of the evader, results in a relatively improved capture time. Since, the dynamics of the evader allow a higher maneuverability at lower speeds, i.e. the evader is able to fly narrower curves at lower speed, ambushing could be prevented in many cases when the evader is 20% slower, delaying its capture.

At this point, a much more interesting question arises: *Is it possible to capture an faster evader under this circumstances?*. To answer that question the evaders speed is raised incrementally.

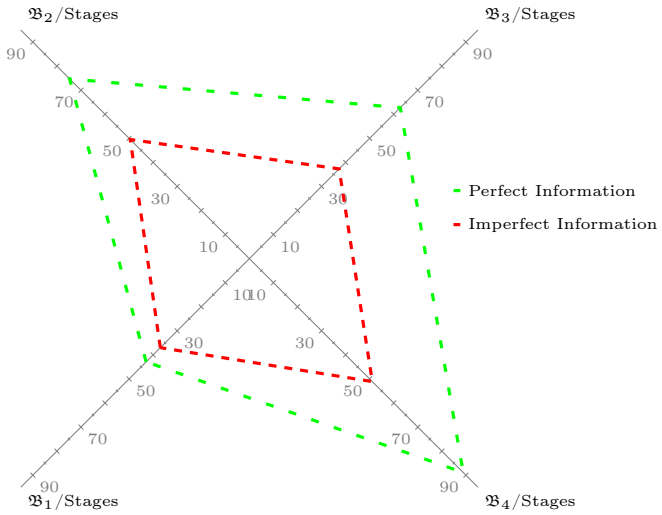

Figure 5.7: *Average of terminal stages in each quadrant (evader* 10% *slower).*

Information	*Average of Stages*	*Improvement*
Perfect	42.90	–
Imperfect	37.00	15.94%

Table 5.10: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_1$ *(evader* 10% *slower).*

Information	*Average of Stages*	*Improvement*
Perfect	74.74	–
Imperfect	49.51	50.95%

Table 5.11: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_2$ *(evader* 10% *slower).*

Information	*Average of Stages*	*Improvement*
Perfect	62.73	–
Imperfect	37.30	68.17%

Table 5.12: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_3$ *(evader* 10% *slower).*

In the next simulation series the evader's maximal speed is set equal to the maximal speed of the pursuers. From this point on it is obvious that

Information	*Average of Stages*	*Improvement*
Perfect	88.86	–
Imperfect	51.10	73.89%

Table 5.13: *Terminal stages in games with perfect and imperfect state information in quadrant* $\mathfrak{B}_4$ *(evader* 10% *slower).*

the evader has the power to escape the pursuers. Therefore, a comparison between the perfect information and the imperfect information case by means of average terminal time in each quadrant is not possible because some or even all terminal stages are infinite. For instance, regarding the values of the game with perfect state information depicted on figure 5.8, the convergence of the game is given only for initial positions within the terminal set plus some few initial positions in the vicinity of the latter. This proves that the capture of an equally powerful evader in a game with perfect state information is virtually impossible. Hence, it is unnecessary to regard situations with a higher speed for the evader because the result would be qualitatively the same.

Now, what happens in the game with imperfect information? The values of this game are illustrated in figure 5.9. As expected, the cooperative behavioral strategy of the pursuers comes into effect. The results reveal how the pursuers must be positioned so that the first pursuer is able to drive the evader towards the second one such that the ambush succeeds. The initial positions where the evader can be captured trends lobe-like away from the evader in opposite direction of the second pursuers position, while there are still positions inside the lobe where capture is not possible.

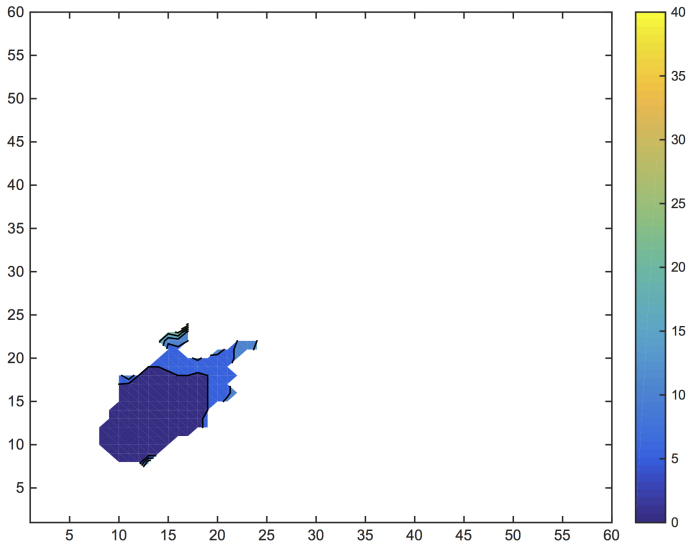

Figure 5.8: *Perfect state information (all players with equal max speed).*

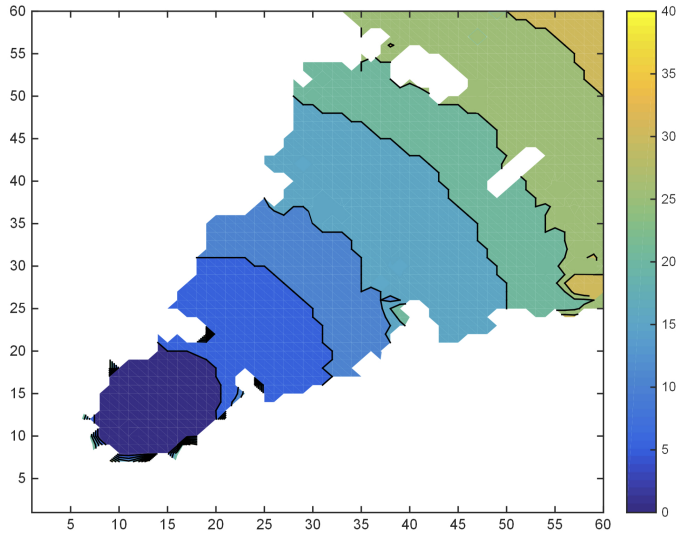

Figure 5.9: *Imperfect state information (all players with equal max speed).*

Figure 5.10 depicts the values of the game with imperfect information with an evader 1.25 times faster than the pursuers. Now, a new problem appears which makes the capture even more difficult. The ambushing strategy works fine for equally powerful evaders and should probably work also for faster evaders, but a faster evader is able to cross and leave the sensing area of the pursuer. And the faster the evader is, the faster it will leave the sensing area. So the time of the pursuer to drive the evader in the right direction is limited. From this point on, the pursuers have to estimate the evaders position based on the last sensed state of the evader, which is anything but optimal. It is assumed that the direction of the evader when leaving the sensing radius does not change, while it is assumed that the evader accelerates till its maximal speed is reached.

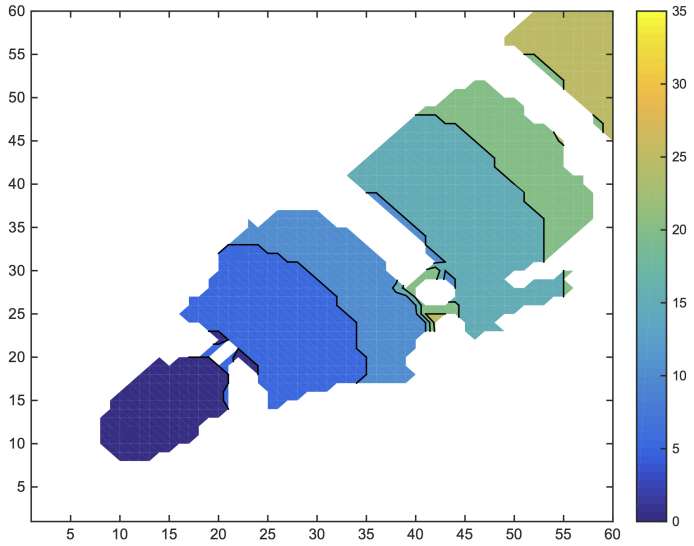

Figure 5.10: *Imperfect state information (evader 25% faster).*

As one can see, with increasing evader speeds like 1.5 times (figure 5.11), 1.75 times (figure 5.12) and 2.0 times (figure 5.13) faster than the pursuers, the set of initial positions, where the evader is captured, shrinks more and more (Note that setting higher maximal speeds for the evader is not possible due to the fact that the utilized UAV system becomes very uncertain or even unstable when setting too high reference values for the speed). Figure 5.14

illustrates the trend of the terminal stages with increasing maximal speed of the evader. The average is calculated by assigning a relatively high value to all games with infinite terminal time, which is 300. The latter is also the value at which the simulations stopped and the divergence of the game is assumed.

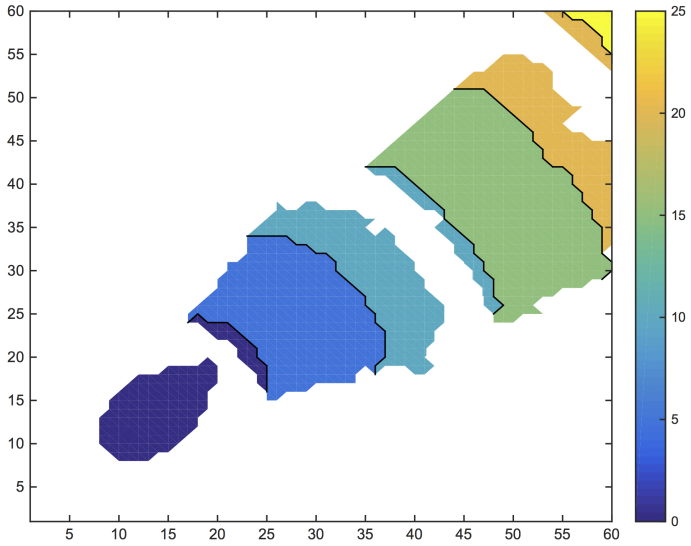

Figure 5.11: *Imperfect state information (evader* 50% *faster).*

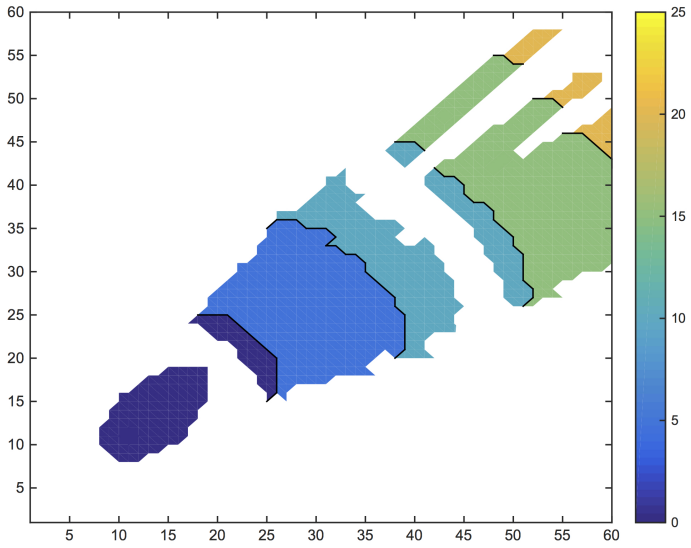

Figure 5.12: *Imperfect state information (evader* 75% *faster).*

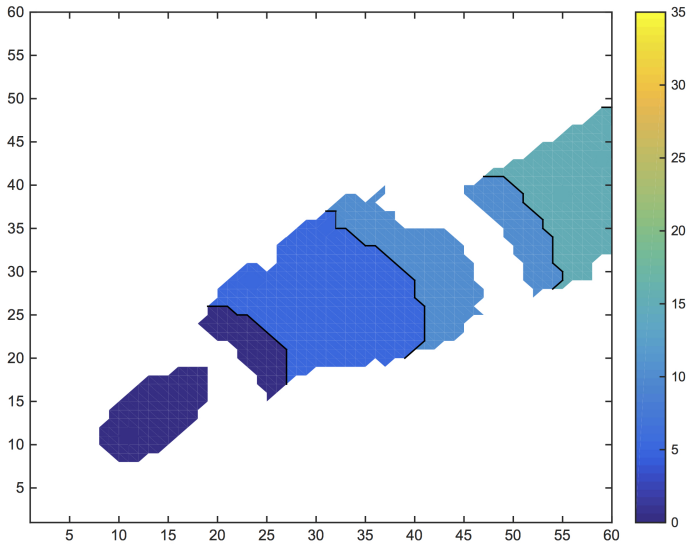

Figure 5.13: *Imperfect state information (evader* 100% *faster).*

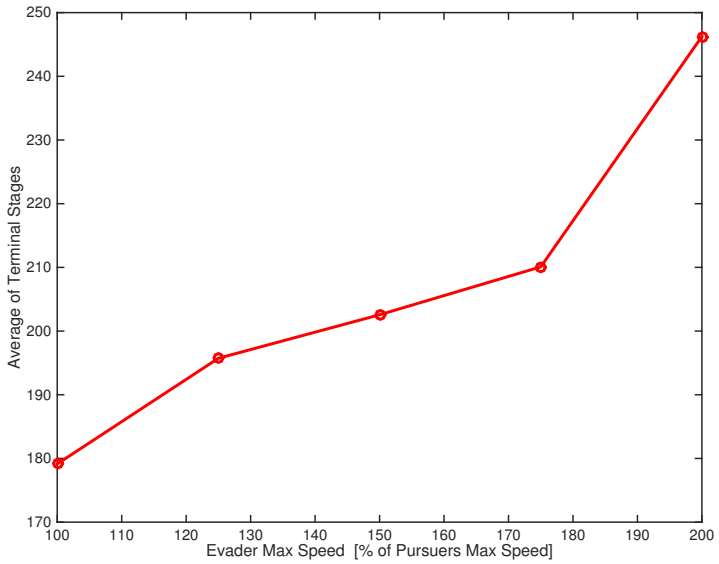

Figure 5.14: *Imperfect state information: average of terminal stages with increasing evader speed.*

5.3.2 PEGs with N-Delay Observation Sharing

In this section, the effect of a delayed information exchange between teammembers is analyzed. Therefore, the same simulation setup as in the last section is chosen but this time the evader's maximal speed is kept constant in each simulation series, while the pursuers and the evader have the same maximal speed. In addition, the observation sharing delay is increased by 1 stage in each simulation series and the effect on the average terminal stage is regarded. In the last section's simulation series, where all players had the same maximal speed and an observation delay of zero, an average terminal time of 179.14 stages is reached. All player's receiving delayed information, have to estimate the actual state of the game. Linear Kalman filters are used for this purpose. Figure 5.15 depicts the course of the average terminal stage with and without estimated states. It can be seen that an observation delay of one step has a minimal impact on the average terminal time with and without estimation. Regarding a delay > 2 it can be seen, that with the estimation states, the capture is still possible even with a delay of 6 stages,

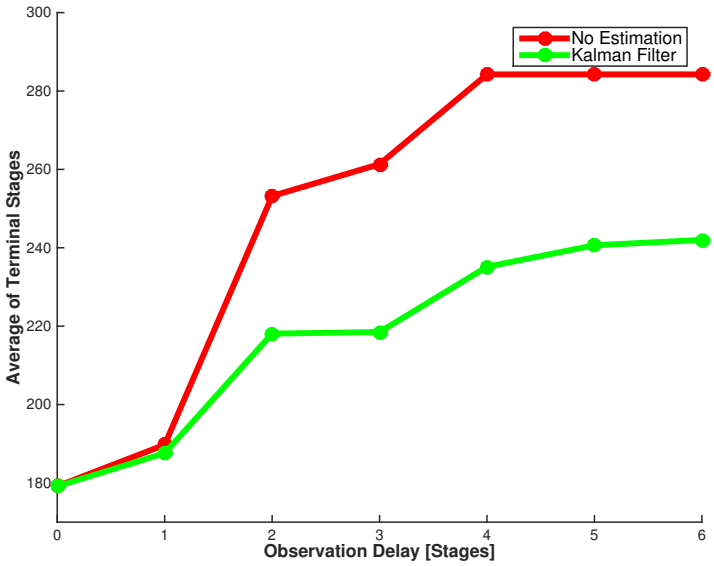

Figure 5.15: *Course of terminal stages with increasing observation delay.*

while the usage of the received states makes the capture impossible having a delay > 3. Here, due to the fact that for initial positions inside the terminal set of the game, the latter terminates immediately, the limit value for the average terminal stages results for the regarded position space and terminal set in 284.24 stages.

5.4 Conclusion

In the first instance, this chapter treats a solution concept for general multi-agent games on the example of PEGs with UAVs. It is shown how a superordinate cooperation between team-members results in a better outcome. Regarding the PEG, two behavioral strategies for the pursuers are proposed, the *pursuit* and the *battue* behavior. By playing a cooperative game, the pursuers are able to assess which behavior combination yields the best outcome for the pursuers over a fixed time horizon. Regarding games with perfect state information, the gain in the outcome is not significant. In more realistic scenarios with imperfect information and observation sharing between

team-members, the supremacy of the approach comes to light. Since the evader has an information inferiority in situations where it cannot observe all pursuers, one observable pursuer is able to drive the evader towards a second pursuer. If the second pursuer's initial position allows a convergence of the game, as depicted for instance on fig. 5.9, it will be able to capture an equally powerful or even a faster evader. It could be also observed that the set of initial positions, where the pursuer is caught, shrinks with increasing evader speed (fig. 5.10-5.13). Another important point is that the observation sharing between team-members could take a delay which is a multiple of the stage duration Δt. Regarding the imperfect information game with delayed observation sharing and an equal speed for all players, it could be observed, that a delay of one step is of no significant consequence but a delay beyond that carries weight. Nevertheless, a state estimation with linear kalman filters considerably raises the number of initial states where the evader is caught, having a delay of up to 6 stages.

6 Implementation and Comparative Study

6.1 RNBC Structure for UAV Agents

In general, complex UAV systems supply a number of behaviors forming an overall complex system behavior. For the purpose of controlling all these behaviors in a dependable way a well-defined control structure is needed. Therefore, the RNBC structure is used successfully in different application domains [Bad06; Bar+09]. [Kan+10] proposes a RNBC structure for an autonomous helicopter platform with eight behavioral levels. The upper levels of this architecture include the more sophisticated but slower behaviors. The lower levels contain the more dynamic behaviors having a shorter reaction time. For instance, the levels 1 (axis-level control), 2 (robot-level control), and 4 (homing) realize a typical cascaded control structure for a UAV position control. This nested structure is extended with the collision avoidance behavior, which is integrated in the intermediate level 3. In this section a conceptional realization of the pursuing UAV agents taking place in a PEG is presented. The proposed RNBC structure for the UAV agents is depicted in figure 6.1.

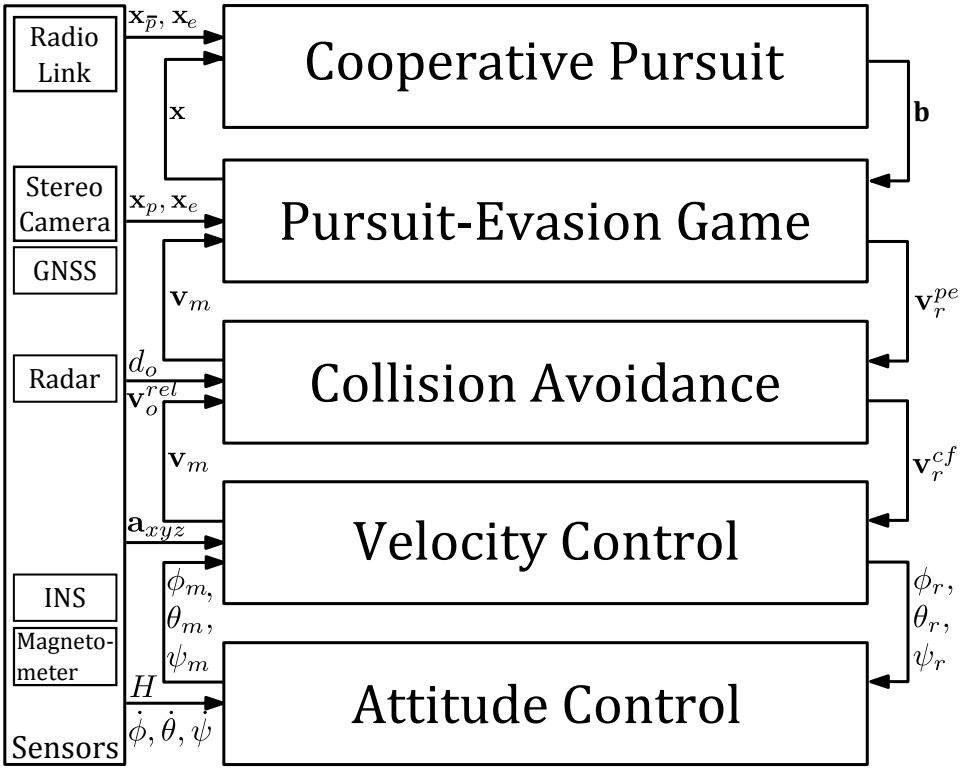

Figure 6.1: *RNBC structure for UAV agents.*

The lowest layer controls the attitude of the UAV and is fed by the sensors with the measured angular velocities $\dot{\phi}, \dot{\theta}, \dot{\psi}$ and the magnetic field H, while being provided with the reference angles ϕ_r, θ_r, ψ_r from the velocity control layer. The reference velocity $\mathbf{v}_r^{cf}$ provided by the collision avoidance layer forms the input of the latter. It assures a collision-free flight and is controlled with help of the linear acceleration measurements $\mathbf{a}_{xyz}$ and the angles ϕ_m, θ_m, ψ_m. The estimated velocity $\mathbf{v}_m$ is fed to the collision avoidance layer. In presence of obstacles the collision avoidance behavior computes a new reference velocity vector $\mathbf{v}_r^{cf}$, based on the reference $\mathbf{v}_r^{pe}$ provided by the pursuit/evasion behavior. The obstacles are detected, e.g., by a radar sensor giving the distance d_o and the relative velocity vector to the UAV agent $\mathbf{v}_o^{rel}$. In the pursuit/evasion layer an optimal strategy, in particular, a reference velocity $\mathbf{v}_r^{pe}$, is calculated, causing a pursuing or evasive behavior for the pursuing or the evading UAV, respectively. Therefore, information about the adversarial UAV has to be gathered. A sensor, e.g., a stereo cam-

era, can be used to estimate the states $\mathbf{x}_e$ and $\mathbf{x}_p$, respectively, in particular, the position and velocity of the foe. The own states $\mathbf{x}_p$ and $\mathbf{x}_e$, respectively, are estimated on-board, supported by the GPS, inertial, and magnetometer sensor measurements. The estimated velocity $\mathbf{v}_m$ is fed from the subjacent collision avoidance layer, while the current position is provided by the GPS sensor directly. In the uppermost layer, the cooperative pursue, the approach described in chapter 5 is implemented. An observation sharing of all pursuer states is done over a radio link between all team members. These information is used to decide the behavioral strategy $\mathbf{b} \in \Sigma$ of the pursuer.

After defining the system architecture, all necessary structures are given to built a real experimentation platform for PEGs with UAVs. The implementation of the real system platform is described in the next chapter in detail. Before moving to the next step, a very important step must be carried out. To be able to asses the PEG approach described in this work, it has to be compared to an approach which has been designed for the same or a similar task.

6.2 Comparison to the Performance Map Approach [RT07]

In literature, approaches for multi-player PEGs with UAVs, regarding faster evaders, can hardly be found. A very promising method approaching this problem appears in the PhD thesis of J. Reimann [RT07]; the Performance Map Approach (PMA). In this section, the PMA is briefly described, and a comparison between the latter and the game-theoretical approach (GTA), described in this work, is performed. Therefore, several simulations, under equal conditions for both approaches, are done. The simulations are ran for three different configurations: (1) faster pursuers, (2) equally fast pursuers and evader, and (3) faster evader with four distinct simulation series. All of those simulation series regard 1000 different PEG simulations. Each of the simulation configurations is done with the same initial conditions of these 1000 simulations. The initial conditions are set by using the Monte-Carlo Method, meaning that the initial positions of all players are uniformly distributed random integer numbers inside a 40×40 grid. The four simulations

series regard: (1) Both teams use the GTA, (2) The pursuers use the PMA and the evader the GTA, (3) The pursuers use the GTA and the evader the PMA, and (4) Both teams use the PMA.

6.2.1 The Performance Map Approach for PEGs with UAVs

The PMA for UAV agents has been developed within the scope of the PhD thesis "Using Multiplayer Differential Game Theory to Derive Efficient Pursuit-Evasion Strategies for Unmanned Aerial Vehicles" [RT07] of J. Reimann at the Georgia Institute of Technology, USA. His work tackles the problem of PEG with UAVs in 2D and proposes three different solution approaches. The PMA has been mainly developed to solve the problem, similar to this work, of catching an equally or more powerful evader with multiple pursuers. The main idea is that through cooperation, more complex tasks can be performed by the pursuers. Therefore, "minimum time information" from each pursuer and evader is used to intercept the evaders. In other words, the author assumes a finite 2-D integer position space and calculates the minimum time needed for each player to reach each point in this position space originally in a game-theoretical way. To get the solution the minimum time Hamilton-Jacobi-Bellman-Isaacs equation

$$\min_{\mathbf{u}_p \in U_p} \max_{\mathbf{u}_e \in U_e} \{\langle \frac{\partial V}{\partial \mathbf{x}}, f(\mathbf{x}, \mathbf{u}_p, \mathbf{u}_e, t) \rangle\} + 1 = 0 \tag{6.1}$$

has to be solved. Since the computational complexity using the differential game framework is assumed to be too high, the author replaces the latter equation by a collection of simpler Hamilton-Jacobi-Bellman equations which are given in equation 6.2. The author denotes that the latter two equation sets are not equivalent. Indeed, a differential game is cut into many single player optimization problems. Due to that step, information is lost and thus the solution of the equations is in general not identical. The main property of a game, which is that the solution depends on the decisions of each involved player, is violated. Of course, the single player optimization problems are much simpler to solve. Reimann uses the Fast Marching

Approach ([SV03],[Set98]) to solve the Hamilton-Jacobi-Bellman equations given in equation 6.2.

$$\begin{aligned}
&\min_{\mathbf{u}_{p_1}\in U_p}\{\langle\frac{\partial V_{p_1}}{\partial \mathbf{x}_{p_1}}, f(\mathbf{x}_{p_1},\mathbf{u}_{p_1},t)\rangle\}+1=0\\
&\qquad\qquad\vdots\\
&\min_{\mathbf{u}_{p_n}\in U_p}\{\langle\frac{\partial V_{p_n}}{\partial \mathbf{x}_{p_n}}, f(\mathbf{x}_{p_n},\mathbf{u}_{p_n},t)\rangle\}+1=0\\
&\min_{\mathbf{u}_{e_1}\in U_p}\{\langle\frac{\partial V_{e_1}}{\partial \mathbf{x}_{e_1}}, f(\mathbf{x}_{e_1},\mathbf{u}_{e_1},t)\rangle\}+1=0\\
&\qquad\qquad\vdots\\
&\min_{\mathbf{u}_{e_n}\in U_p}\{\langle\frac{\partial V_{e_n}}{\partial \mathbf{x}_{e_n}}, f(\mathbf{x}_{e_n},\mathbf{u}_{e_n},t)\rangle\}+1=0.
\end{aligned} \tag{6.2}$$

After solving the minimum time problem, a so called minimum time map is established (figure 6.2). The minimum time map consists of regions overlaying the whole position space for each involved player. All positions within a player's region can be reached fastest by the player itself. As long as the evader stays in its region, it cannot be caught. The capture time for an evader is given by the minimum time at the boundaries of its region. If a region boundary adjoins the position space boundary, as depicted in figure 6.2, the evader is able to escape. Hence, the evader will try to minimize the distance to that boundary point in his region with the highest time value. On the other side, the pursuers will also try to minimize the distance to the point with the maximal time value which is adjacent to the border of the evaders region. This results in an interception or containment strategy. Reimann proposes two escaping strategies. In the first one, the so called "Simple Evasion Strategy", the evader decides at the beginning which escape corridor it wants to use to escape and it will not change the strategy till the end of the game (open-loop approach). In the second one, the "Adaptive Pursuit and Evasion Strategies", both the evaders and the pursuers are able to change their strategy (escape routes or interception points) from time to time, based on the actual system state (closed-loop solution). The game ends when the pursuers catch the evader or when the evader is able to reach the border of the regarded position space.

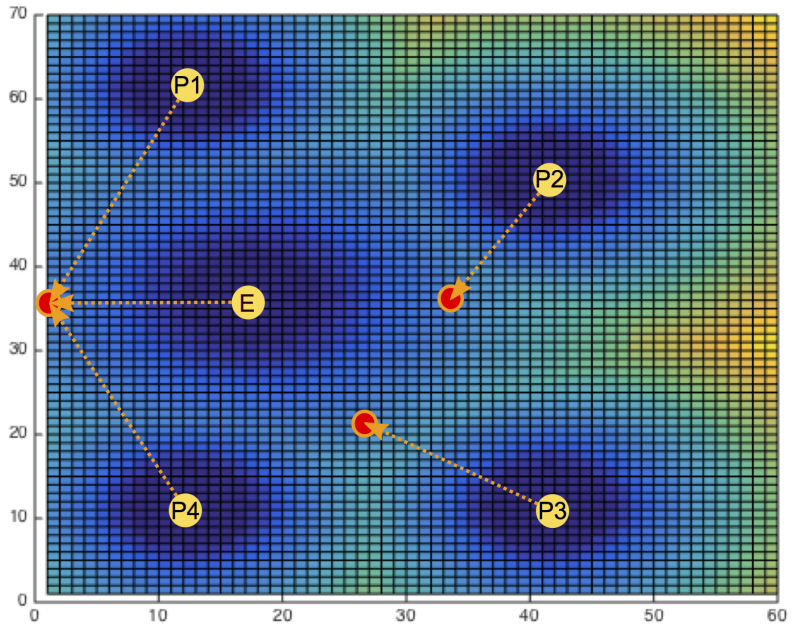

Figure 6.2: *PMA: minimum time map.*

The author used for the simulations very simple vehicle dynamics which are:

$$\frac{d\mathbf{r}}{dt} \leq v_i, i \in \tau, \tag{6.3}$$

with v_i being the maximum velocity of a player i. Reimann presented a total of 15 simulations where he showed that under specific conditions a faster evader can be caught with this approach. It must be noted, that the complexity increases exponentially with the size of the mesh.

In the next section the PMA with "Adaptive Pursuit and Evasion Strategies" is used to be compared with the GTA from this work since both are closed-loop approaches. The PMA with "Adaptive Pursuit and Evasion Strategies" is implemented in MATLAB, while using the toolbox "Toolbox Fast Marching" [Pey09].

6.2.2 Simulation Set-Up

The following assumptions are made for all simulations:

- The pursuing team **P** consists of three pursuers p_1, p_2, and p_3.
- The evading team **E** consist of one evader e.
- The players are multi-rotor UAVs with the model described in 7.3.
- A perfect state information structure is assumed.
- Since the PMA works only on a bounded 2-D grid, the operation space is limited to $X^o \times Y^o$, with $X^o, Y^o \in [80, 80]$.
- All initial positions are assumed to be on a 40×40 grid which covers the plane $[20, 60] \times [20, 60]$ in $X^o \times Y^o$.
- Each series is simulated by running $i = 1..1000$ simulations with uniformly distributed random initial positions $\{(\tilde{x}^i_{p_1}, \tilde{y}^i_{p_1}), (\tilde{x}^i_{p_2}, \tilde{y}^i_{p_2}), (\tilde{x}^i_{p_3}, \tilde{y}^i_{p_3}), (\tilde{x}^i_e, \tilde{y}^i_e)\}$ inside a 40×40, with the resulting initial game states:

$$\chi^1_i = \left(\begin{bmatrix} \tilde{x}^i_{p_1} \\ \tilde{y}^i_{p_1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \tilde{x}^i_{p_2} \\ \tilde{y}^i_{p_2} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \tilde{x}^i_{p_3} \\ \tilde{y}^i_{p_3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} \tilde{x}^i_e \\ \tilde{y}^i_e \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \right), \quad \tilde{x}^i_\cdot, \tilde{y}^i_\cdot \in [20, 60], i = 1..1000.$$

- The maximum translational velocity of the pursuers is set to $\mathbf{v}^{max}_{p_1} = \mathbf{v}^{max}_{p_2} = \mathbf{v}^{max}_{p_3} = (15, 15, 0)^T$, and the maximal absolute velocity to $\mathbf{v}^{maxA}_{p_1} = \mathbf{v}^{maxA}_{p_2} = \mathbf{v}^{maxA}_{p_3} = 15$. Also, $\mathbf{v}^{max}_e = (10, 10, 0)^T$, and $\mathbf{v}^{maxA}_e = 10$, are set for the evader in configuration one, $\mathbf{v}^{max}_e = (15, 15, 0)^T$, and $\mathbf{v}^{maxA}_e = 15$ for configuration two, and $\mathbf{v}^{max}_e = (16.5, 16.5, 0)^T$, and $\mathbf{v}^{maxA}_e = 16.5$ for configurations three, respectively.

- For the game-theoretical approach: the number of different discrete actions for each player is set to $s_{p_1} = s_{p_2} = s_{p_3} = s_e = 9$, while $s_{p_1,u} = s_{p_2,u} = s_{p_3,u} = s_{e,u} = s_{p_1,v} = s_{p_2,v} = s_{p_3,v} = s_{e,v} = 2$, and $s_{p_1,w} = s_{p_2,w} = s_{p_3,w} = s_{e,w} = 1$.
- The stage duration of the PEG is set to $\Delta T = 0.1$ while the velocity and attitude controller are sampled with $\Delta t = 5e - 3$.
- The game-theoretical approach with cooperating pursuers (Pareto efficiency solution), as described in section 5.2 is used, while using the MTPGD for the non-cooperative part of the PEG.
- The capture radius is set to $d_\epsilon = 5$.
- The evader escapes when it leaves the operation space $X^o \times Y^o$ or when 300 stages are exceeded.
- Simulations pattern:

Configuration	$\mathbf{v}_{p_i}^{maxA}$	$\mathbf{v}_e^{maxA}$
1	15	10
2	15	15
3	15	16.5

Table 6.1: *Simulation configurations.*

Series	Pursuers	Evader
1	GTA	GTA
2	PMA	GTA
3	GTA	PMA
4	PMA	PMA

Table 6.2: *Approach combinations for each series (*1000 *simulations).*

6.2.3 Performance Measure

A performance measure Λ is defined to be able to assess the two regarded algorithms. Therefore, the following four criteria are taken into account:

- The average of terminal stages $\overline{K_C}$ for all games which converged (evader is caught).
- The variance of terminal stages σ_K^2 for all games which converged.
- The number of games which diverged D (evader is not caught).
- The total number of computational operations O (time complexity).

Those factors are normalized to the highest value of each regarded run per configuration, and are summed up, having a maximum value of 4:

$$\Lambda = \hat{\overline{K_C}} + \hat{\sigma_K^2} + \hat{D} + \hat{O} \tag{6.4}$$

Note, that the smaller the value the better.

6.2.4 Simulation: Slower Evader

The first simulation series regards the case, when both teams **P** and **E** are choosing their actions by using the game-theoretical approach (GTA). Figure 6.3 illustrates the results in form of a histogram. It can be seen that 114 simulations have a value of 1, which means that the initial condition is inside the terminal set or in the vicinity and thus the evader is caught immediately or after one action. There are 236 simulations that diverged. Regarding the remaining simulation results, an average capture time of 14.50 stages and a variance of 65.63 can be seen. Note, that the games that diverged here when using the GTA approach for both teams happened only because the evader is able to reach the boundary of the regarded position space. An open position space as regarded before would lead to a convergence of all games. In the second simulation series the pursuers are using the PMA and the evader the GTA. It can be seen that 114 simulations terminated after 1 stage. When using the PMA, the pursuers are not able to capture the evader in 392 of the simulations which is a first indicator for the superiority of the evader using the GTA. An average capture time of 13.59 stages with variance 53.96 results for this configuration. The results of the third simulation series, where the pursuers are using the GTA and the evader the PMA, are depicted in fig. 6.5. A total of 128 diverging games can be observed and a total number of 116 simulations ended after one stage. The result of the average capture

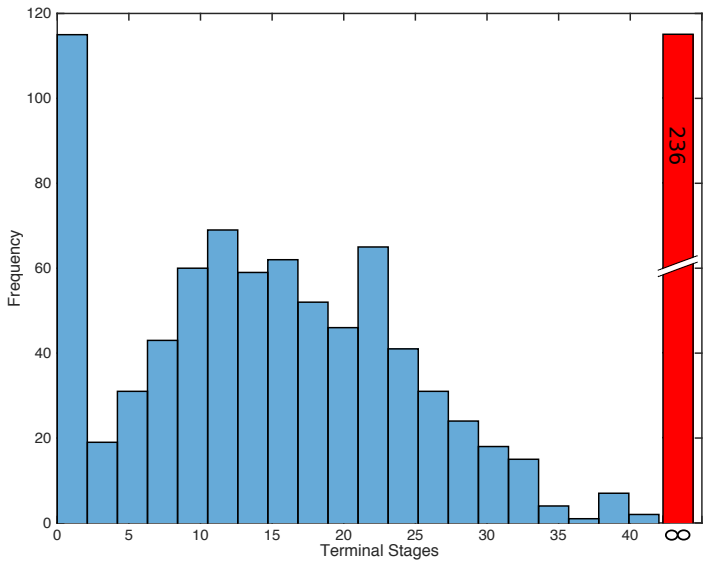

Figure 6.3: *Slower evader: GTA only.*

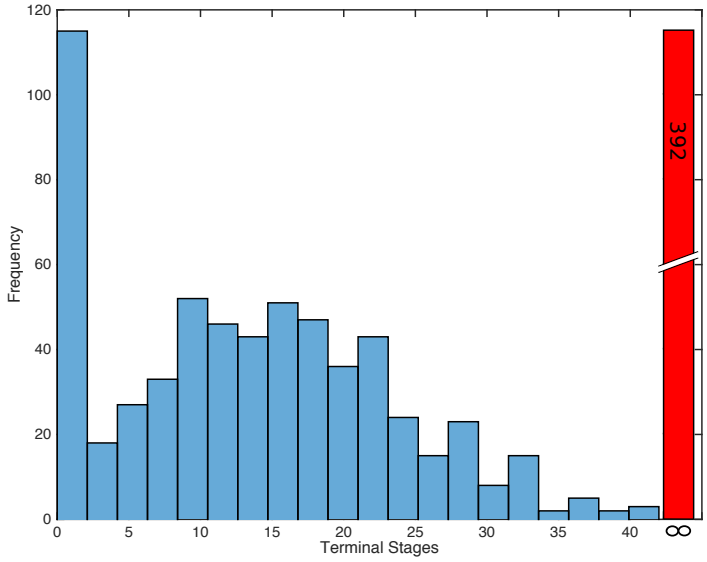

Figure 6.4: *Slower evader: pursuers: PMA vs. evaders: GTA.*

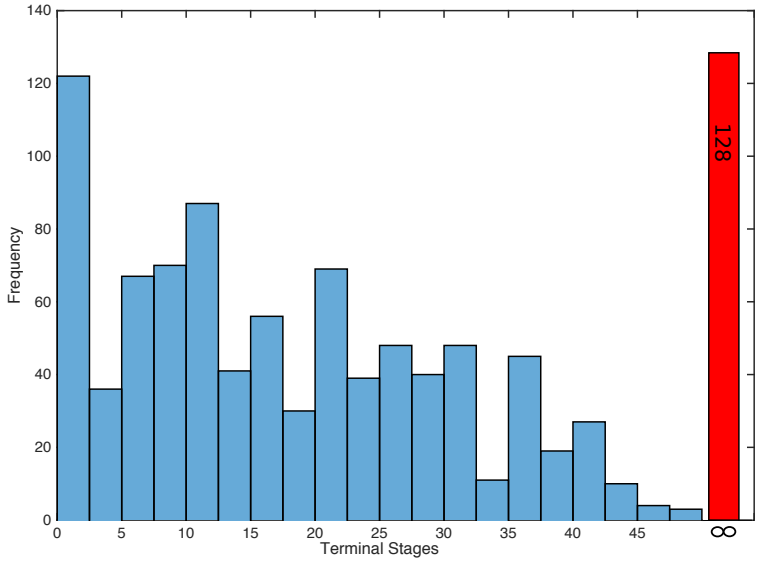

Figure 6.5: *Slower evader: pursuers: GTA vs. evaders: PMA.*

time is for this combination 17.23 stages with a variance of 132.38. The simulation results of the series in which all players have been using the PMA can be seen in fig. 6.6. Here, a total number of 345 of diverging simulations could be stated and a total number of 116 simulations ended after one stage. The average capture time results in 14.99 stages with a variance 91.82. All values of the relevant criteria are summarized in table 6.3.

P/E	$\widehat{\overline{K_C}}$	$\hat{\sigma^2}$	$\hat{D}$	$\hat{O}$	Λ
GTA/GTA	0.8416	0.4958	0.602	0.0452	1.9846
PMA/GTA	0.7887	0.4076	1	1	3.1963
GTA/PMA	1	1	0.3265	1	3.3265
PMA/PMA	0.87	0.6936	0.6811	0.9567	3.2014

Table 6.3: *Results GTA vs. PMA: slower evader.*

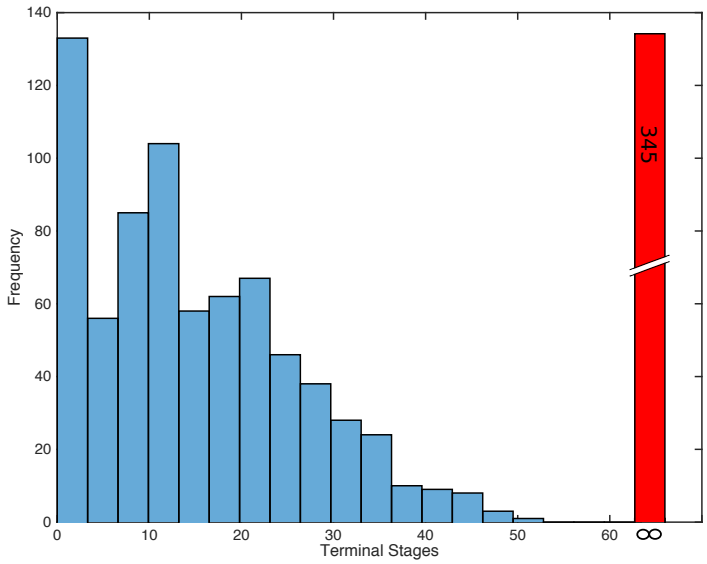

Figure 6.6: *Slower evader: PMA only.*

6.2.5 Simulations: Equally Fast Evader

In the second configuration all players have the same maximal speed. Moreover, the same set of the 1000 initial player positions, as utilized for configuration one, are used.

In the GTA vs. GTA simulation series it can be seen (figure 6.7), that way more games diverge (760) than in configuration one. The average terminal stage results in 7.07 with variance 16.28. Moreover, 118 game ended after one stage. It can be observed, that for a relatively small set of initial conditions the convergence of the game is achievable. Only initial states in the neighborhood of the terminal set result in the capture of the evader, which is also the result achieved in section 5 and depicted on figure 5.8. In the next simulation series it is examined how the PMA pursuers perform against the GTA evader (figure 6.8). The evader is able to escape in 868 games, while 115 games converged after one stage. That results in an average terminal stage of 2.94. The two latter simulation series show, that the GTA evader achieves better results against the PMA pursuers. The other way around, GTA pursuers vs PMA evaders is examined here. With a total of 532 es-

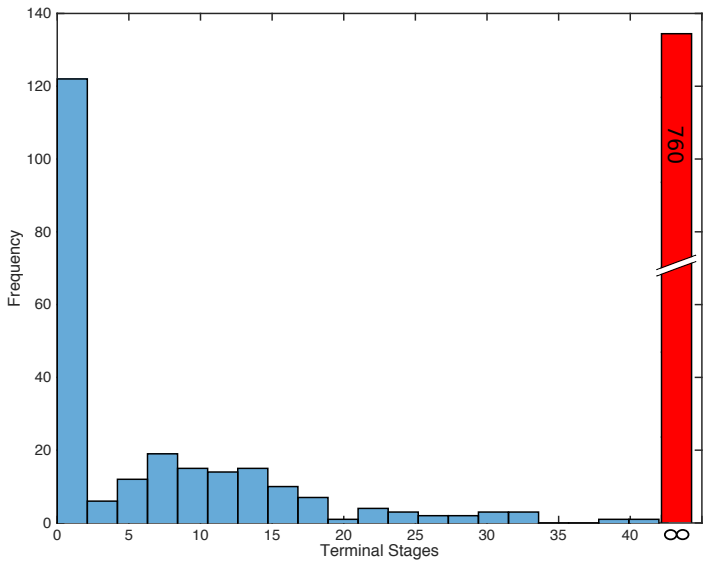

Figure 6.7: *Equally fast evader: GTA only.*

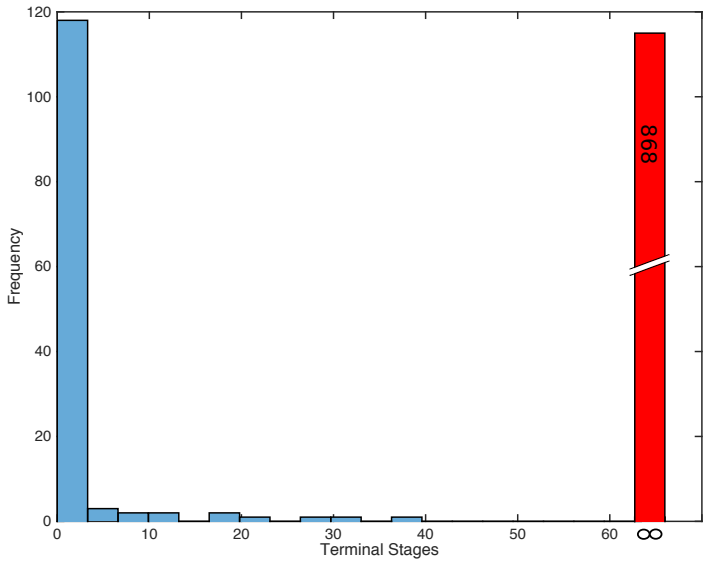

Figure 6.8: *Equally fast evader: pursuers: PMA vs. evaders: GTA.*

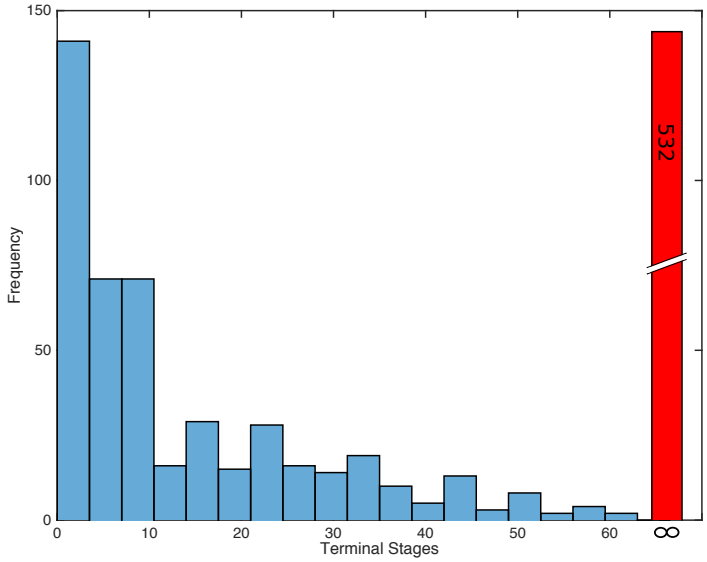

Figure 6.9: *Equally fast evader: pursuers: GTA vs. evaders: PMA.*

capes and an average terminal stage of 13.55 and a variance of 98.04, while 115 game ended after one stage, the GTA evader shows again its superiority against the PMA evader, regarding the evasion behavior. To complete the simulations, the PMA pursuers vs PMA evader games are examined, too (figure 6.10). It can be observed, that there are 791 escapes with an average of 10.17 for the terminal stage and a variance of 51.44, while 115 games ended after one step. In respect to simulation series three, where the evader uses also the PMA to escape, the pursuers did worse when using the PMA regarding the convergence of the game. The performance for this configuration can be found in table 6.4.

P/E	$\widehat{\overline{K_C}}$	$\hat{\sigma^2}$	$\hat{D}$	$\hat{O}$	Λ
GTA/GTA	0.5218	0.1661	0.8756	0.0453	1.6088
PMA/GTA	0.217	0.0795	1	1	2.2965
GTA/PMA	1	1	0.6129	1	3.6129
PMA/PMA	0.7506	0.5247	0.9113	0.9533	3.1399

Table 6.4: *Results GTA vs. PMA: equally fast evader.*

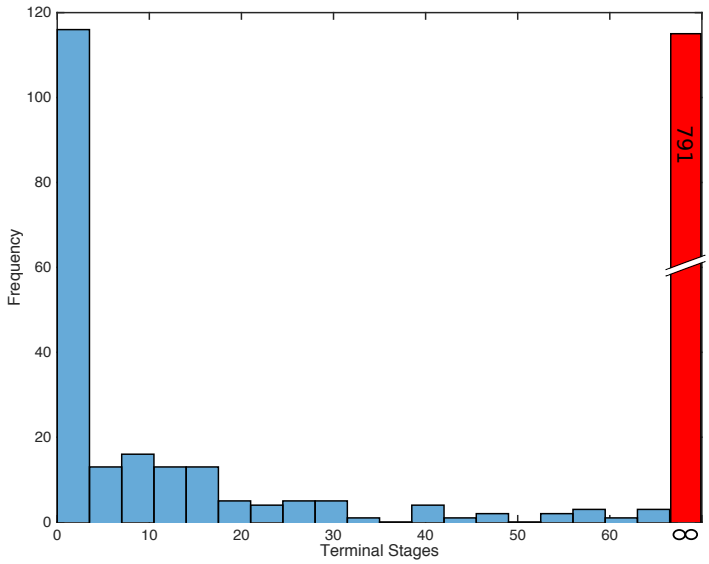

Figure 6.10: *Equally fast evader: PMA only.*

6.2.6 Simulations: A Faster Evader

In the third configuration the evader is 10% faster than the pursuers. Here, too, the same set of the 1000 initial player positions, as utilized for the other configurations, is used.

In simulation series one it can be observed (figure 6.11), that some more games diverge (809) in respect to configuration two. The average terminal stage and variance is also less with 5.32 and 9.80, respectively. This is because the relatively small set of initial states where the evader can be caught (with configuration two) shrinks for configuration three. In other words, more games diverge, while the neighborhood in which a game converges gets smaller for a faster evader. This results also in a smaller average terminal stage and variance for the games that converge. For simulation series two with configuration three (figure 6.12) the evader is able to escape in 887 games, while 111 games converged after one stage. That results in an average terminal stage of 1.14 and variance of 0.14. As regarded for configuration one and two, the two latter simulation series reveal, that the GTA evader did much better against the PMA pursuers than against the GTA pursuers.

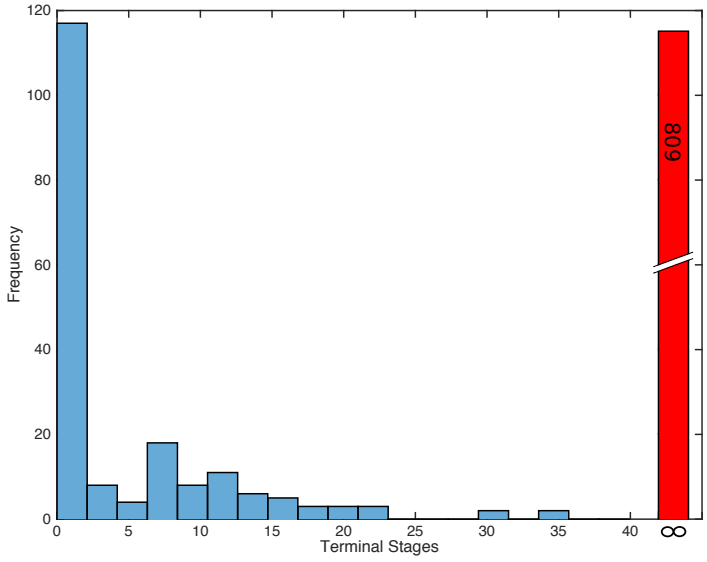

Figure 6.11: *Faster evader: GTA only.*

Next, series three is regarded, where the GTA pursuers compete against the PMA evader (figure 6.13). With a total of 400 escapes and an average terminal stage of 15.72 and a variance of 154.32, while 163 game ended after one stage, the GTA evader is able to enforce much more diverging games than the PMA evader. Lastly, the PMA pursuers vs PMA evader games are examined (figure 6.14). It can be seen, that with 840 escapes and with an average of 6.62 for the terminal stage with variance 32.38, while 112 games ended after one step. The performance of this run is summarized in table 6.5.

P/E	$\widehat{\overline{K_C}}$	$\hat{\sigma^2}$	$\hat{D}$	$\hat{O}$	Λ
GTA/GTA	0.3384	0.0635	0.9121	0.0452	1.3592
PMA/GTA	0.0725	0.001	1	1	2.074
GTA/PMA	1	1	0.451	1	3.451
PMA/PMA	0.4211	0.21	0.947	0.9560	2.534

Table 6.5: *Results GTA vs. PMA: faster evader.*

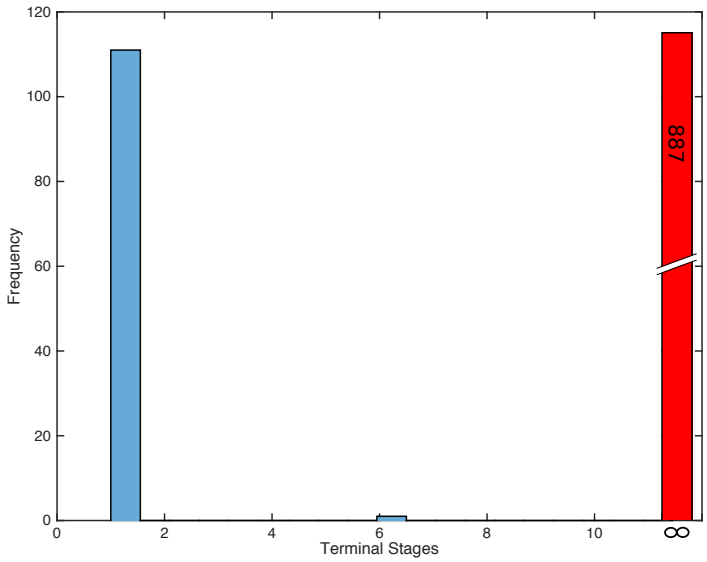

Figure 6.12: *Faster evader: pursuers: PMA vs. evaders: GTA.*

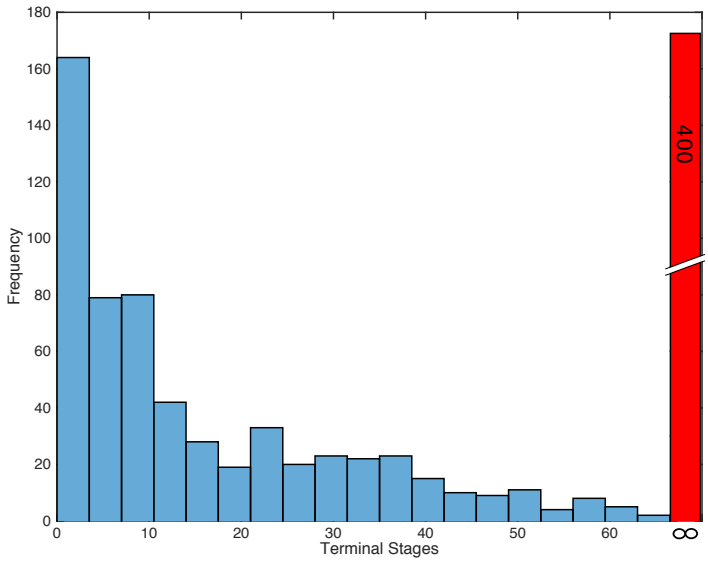

Figure 6.13: *Faster evader: pursuers: GTA vs. evaders: PMA.*

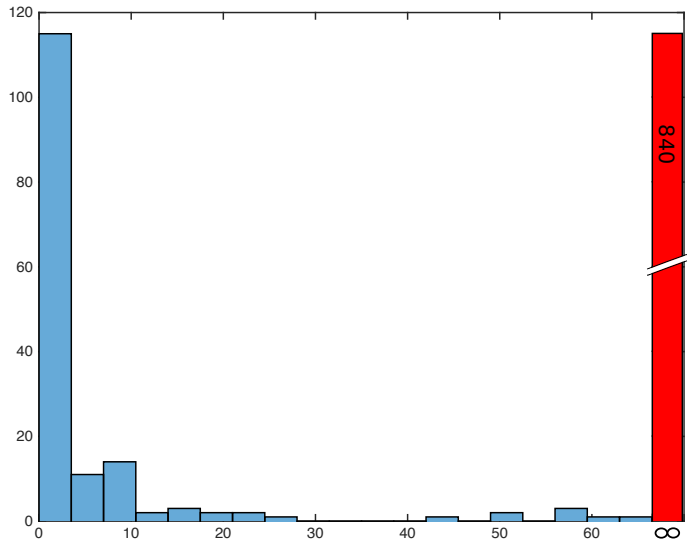

Figure 6.14: *Faster evader: PMA only.*

6.2.7 Conclusion

In the first part of this chapter, the system architecture of each of the UAV players is presented. The UAV agents are modeled by using a behavior based description, the RNBC structure. While the pursuers follow a one to one description of the structure depicted in fig. 6.1, the evaders are missing the uppermost layer. By regarding PEGs with more than one evader, a cooperation between evaders seems reasonable. Hence, evaders with a "Cooperative Evasion" layer are conceivable.

In the second part, a comparison of the game-theoretical approach for PEGs with UAVs presented in this work to the Performance Map Approach for PEGs with UAVs is given. This approach has been designed with the aim of being able to catch faster evaders by outnumbering. In order to compare the approaches in a fair way, some restrictions to the GTA had to be set like the fixed and finite set of position states the game can reach. This impacts especially the termination condition. The game ends when the evader reaches the boundary of the position state. Moreover, a performance measure is introduced including the average of terminal stages, the variance of terminal

stages, the diverged games and the operations needed for calculation of the regarded approach combination.

Nevertheless, for a slower evader, it can be seen that in series one and four, the average terminal stages are similar with 14.50 (GTA only) and 14.99 (PMA only), while the number of games that diverged are also almost equal. The two criteria that had a big difference are on the one hand the variance of the terminal stage (65.63 for GTA and 91.82 for PMA) and on the other hand the total operations needed to compute the simulation results. While for simulation series one (GTA only) $1.045 \cdot 10^8$ operations are needed, $2.21 \cdot 10^9$ are needed for simulation series four (PMA only), which is more than 21 times more. The latter makes the biggest difference between the approaches. Regarding the Λ's of each series for a slower evader, series one (GTA only) has the best performance with $\Lambda_1^1 = 1.9846$ by far. All other series had very similar results with $\Lambda_1^2 = 3.1963$, $\Lambda_1^3 = 3.3265$, and $\Lambda_1^4 = 3.2014$. The reason for the high values of series two and three is easily explained. It is observed that in series two (P:PMA/E:GTA) the GTA evader does slightly better against the PMA pursuers regarding the average terminal stage and the variance. In reverse this results in a higher number of diverging games. Moreover, in series three (P:GTA/E:PMA) a higher average value and a higher variance is calculated. In addition, the number of diverging games is much smaller than in the other series. Since both approaches have to be used in series two and three, the operations needed for calculation reach their maximum here, which is only slightly bigger than that of the fourth series (PMA only). In this configuration the fourth and the first series yield similar results, but the big difference in operations needed for the calculation yields a much worse performance for series four (PMA only).

Making the evader equally fast yields a bigger gap between all different series regarding the performance. While series one (GTA only) gives again the best performance with $\Lambda_2^1 = 1.6088$, the second best performance could be achieved in the second series (P:PMA/E:GTA). Due to the fact that almost in all games the GTA evader is able to escape the PMA pursuers, and thus capture occurred only when one pursuer started in the vicinity of the terminal set, resulting in a very low value for the average terminal stage and the variance, the performance reaches a value of $\Lambda_2^2 = 2.2965$. In series three (P:GTA/E:PMA), the worst performance could be observed.

Since the GTA pursuers are able to catch the PMA evader in 468 of the games and since the average value of terminal stages (13.55), the variance (98.04) and the number of operations ($2.57 \cdot 10^9$) needed for calculation have high values, the performance is given by $\Lambda_2^3 = 3.6129$. Lastly, in series four (PMA only) a relatively high value for all four criteria can be observed leading to a performance of $\Lambda_2^4 = 3.1399$ and the second worse result for this configuration.

The results for the final configuration where the evader is faster than all pursuers, have the same trend as regarded in the last configuration, while the gap between the performance values is even bigger. This is because the effects observed for a equally fast evader affect the results even more when the evader is faster. This yields to the performance values $\Lambda_{31} = 1.3592$, $\Lambda_3^2 = 2.074$, $\Lambda_3^3 = 3.451$, and $\Lambda = 2.534$.

The comparison leads to the following three conclusions: (1) In a PEG with perfect state information structure, as it is demanded by the PMA, it is almost impossible to catch an equally fast evader. It is only possible when the pursuer's initial positions are inside or in the vicinity of the terminal set. Since in [RT07] only 15 selected simulations with appropriate initial conditions are presented and assessed, no significant statement about the approach could be made before. (2): The GTA escape/pursuing behavior has an advantage over the PMA. Regarding the number of converging games in series two (P:PMA/E:GTA) and three (P:GTA/E:PMA) for each configuration, it can be seen that the GTA approach yields always the better results. In other words, more diverging games when the evader is using the GTA and more captures when the pursuers are using the GTA. Regarding series one (GTA only) and four (PMA only) in each configuration, the number of diverging games and the average terminal stage are approximately equal, but the variance of series four (PMA only) is always much bigger. (3): The time complexity of the GTA algorithm, even when restricting the mesh size, is significantly smaller than the one of the PMA. Regarding the operations for the calculation of the solutions in each configuration the PMA needed 21.14, 21.06, and 21.15 times more operations, respectively.

7 Realization of the Experimentation Platform

7.1 Autonomous Hex-Rotor UAV for PEGs

Experiments with a real platform demand the design and the implementation of real UAV agents. Therefore, a hex-rotor system which has been designed in [Kir15] is utilized. Within the scope of the work in [Kir15], an attitude and linear velocity controller have been designed and tested and are used for the implementation of the UAV agents for the PEGs. The flight control, attitude control, and velocity control run on two embedded computers which are described in the next section of this chapter. The system architecture of this system follows the RNBC structure and includes the last two layers of the RNBC structure depicted in figure 6.1. For this work, the Pursuit-Evasion Layer and the Cooperative Pursue Layer are implemented as well on the real system. The collision avoidance layer is neglected for the experiments, but in chapter 7.6 simulations of two-player PEGs in an environment with static and moving obstacles are treated with the collision avoidance layer included. Figure 7.1 depicts a block diagram of the realization of the autonomous agent. In the following sections, a more detailed description of all key components is given.

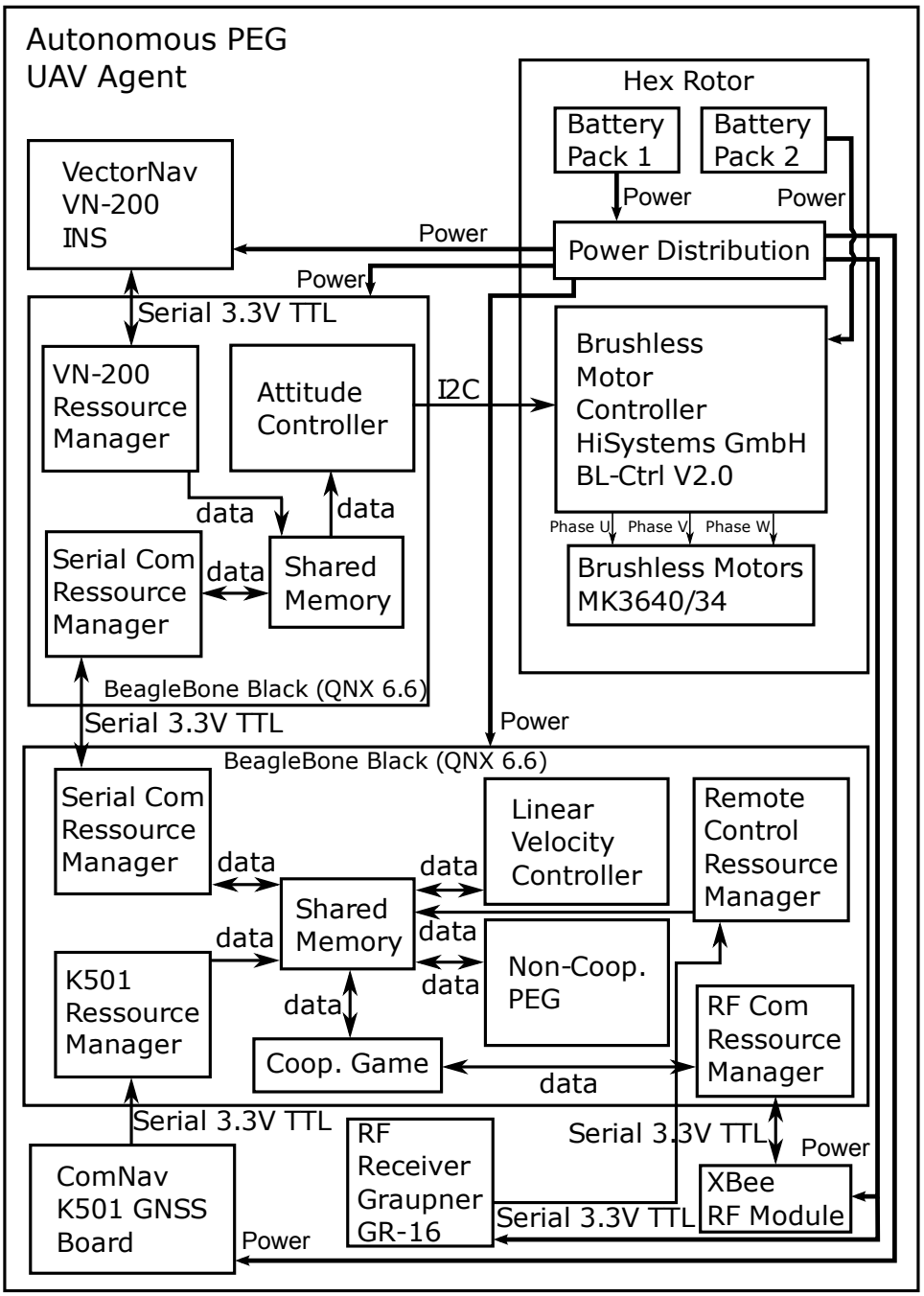

Figure 7.1: *Implementation of the autonomous agent.*

7.2 Hardware

Hex Rotor

Most of the basic components of the MK-Hexa XL manufactured by *HiSystems GmbH* are used for the realization of the test platform. The basic set includes, amongst others, six rotors, six motors, six cantilevers, a mounting plate, six brushless motor controllers, a power distribution board and a landing skid. Furthermore, two battery packs (one for the motor and one for the electronic parts) and a battery holder are implemented. The original flight control unit is removed and replaced by an own realization described below. Figure 7.2 shows the completed hex rotor with a custom electronics box on its top.

Figure 7.2: *Hex-rotor UAV.*

Embedded Computers

There are many low-power and small-size computers available, e.g., Raspberry Pi [Ras14], Cubieboard [Cub14], BeagleBoard [Tex14a], and some vari-

ants. Many of those single-board computers are open-source hardware, assembled with a low-frequency processor. For this work two BeagleBone Black (figure 7.3) are utilized, a community-supported development platform with a TI Sitara AM335x 1GHz ARMCortex A8 processor and 512MB DDR3 RAM ([Tex14b]). The embedded computers run QNX 6.6, a RTOS enabling the implementation and execution of real-time applications written in C/C++ programming language. Two BeagleBone Black are implemented on the hex rotor, while the first computer is running the attitude control and the second one the linear velocity control, the pursuit-evasion game, and the cooperative pursue.

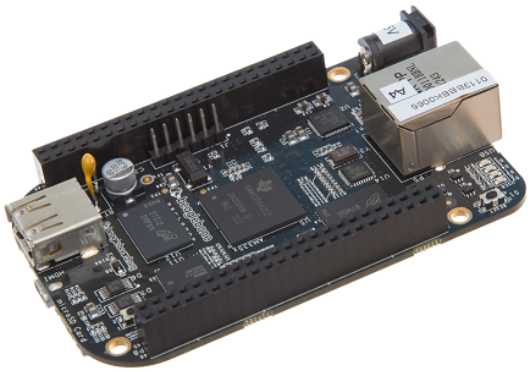

Figure 7.3: *BeagleBone black [Bbb].*

Inertial Navigation System

The Inertial Navigation System (INS) used to measure all states required for the attitude control is the Vectornav VN-200 Rugged (figure 7.4). It includes a 10-axis MEMS IMU (3-axis accelerometer, 3-axis gyroscope, 3-axis magnetometer and barometer) and a GPS receiver. It runs Kalman filtering algorithms providing position, velocity, and orientation, while having a size of $36x33x9.5mm$ and a weight of $16g$.

GNSS

The Vectornav VN-200 Rugged is able to provide 3-axis linear velocity estimations. Nevertheless, experiments revealed that the required accuracy

Figure 7.4: *Vectornav VN-200 rugged [Vn2].*

for the linear velocity control cannot be achieved. Despite the manufacturer specifications of an accuracy of $+/-0.05m/s$, an accuracy of about $+/-1m/s$ is measured. In order to provide the linear velocity controller with proper linear velocity measurements, a differential GNSS is used. The utilized GNSS includes the ComNav K501 GNSS Board (figure 7.5) with an RF module for the communication with the ComNav T-300 GNSS base station (figure 7.6). The ComNav GNSS is able to receive GPS, BeiDou and GLONASS signals, while providing a linear velocity accuracy of $0.03m/s$.

Figure 7.5: *ComNav K501 GNSS board [K50].*

RF Module

The XBee-PRO modules by Digi International [Xbeb], depicted on figure 7.7, are used for the inter-team communication. The XBee-PRO has a range of up to $100m$ in urban/indoor areas and up to $1600m$ in outdoor/RF LoS while a serial communication with $1200bps$-$250kbps$ is possible.

Figure 7.6: *ComNav T-300 GNSS base station [T30].*

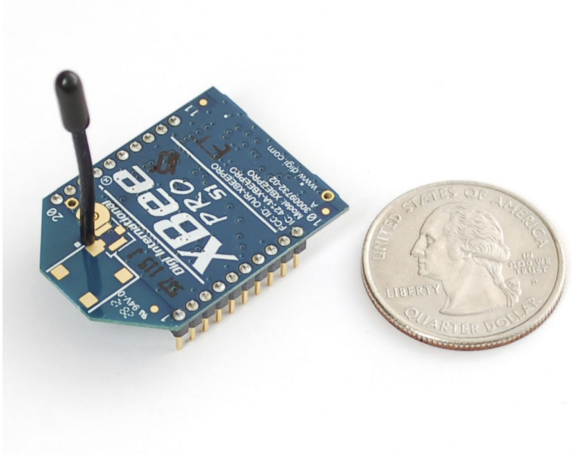

Figure 7.7: *XBee RF module [Xbea].*

Remote Control

The Graupner MX-20 RF remote control (figure 7.8) is used for the manual control of the hex rotor. Moreover, all implemented modes can be activated by several triggers. The implemented modes are manual control, assisted control (linear velocity control with reference velocities provided over the D-Pad) or PEG (fully autonomous). The Graupner MX-20 has 12 channels and operates at a $2.4GHz$ frequency.

Figure 7.8: *Graupner MX-20 [Mx2].*

7.3 Dynamical Model

The detailed derivation of the dynamic model used in this work can be found in [Voo09]. For modeling the hex-rotor dynamics the mechanical configuration depicted in figure 7.9 is assumed. The body fixed frame and the inertial frame are denoted by $\mathbf{e^B}$ and $\mathbf{e^I}$, respectively. The UAV is defined as a point mass. To derive the equations of motions, the following notations are necessary. $\mathbf{P^I} = (x, y, z)^T$ is the position vector of the hex-rotor's center of gravity in the inertial frame, $\mathbf{P^B} = (x_B, y_B, z_B)^T$ is the position vector of the hex-rotor's center of gravity in the body fixed frame, $\mathbf{v} = (u, v, w)^T$ are the linear velocities in body fixed frame, $\boldsymbol{\omega} = (p, q, r)^T$ are the angular rates for roll, pitch, and yaw in body fixed frame, and $\boldsymbol{\Theta} = (\phi, \theta, \psi)^T$ is the vector of Euler angles. A key component of the hex-rotor model is the transformation between inertial and body frames. Rigid body dynamics are derived with respect to the body frame that is fixed in the center of gravity of the hex-rotor. However, to simulate the motion of the hex-rotor in the inertial frame a transformation of the coordinates is needed. If the multi-rotor's attitude is parameterized in terms of Euler angles the transformation can be performed using the rotation matrix $\boldsymbol{R(\Theta)}$, which is a function of roll, pitch, and yaw angles. Using s and c as abbreviations for $\sin(\cdot)$ and

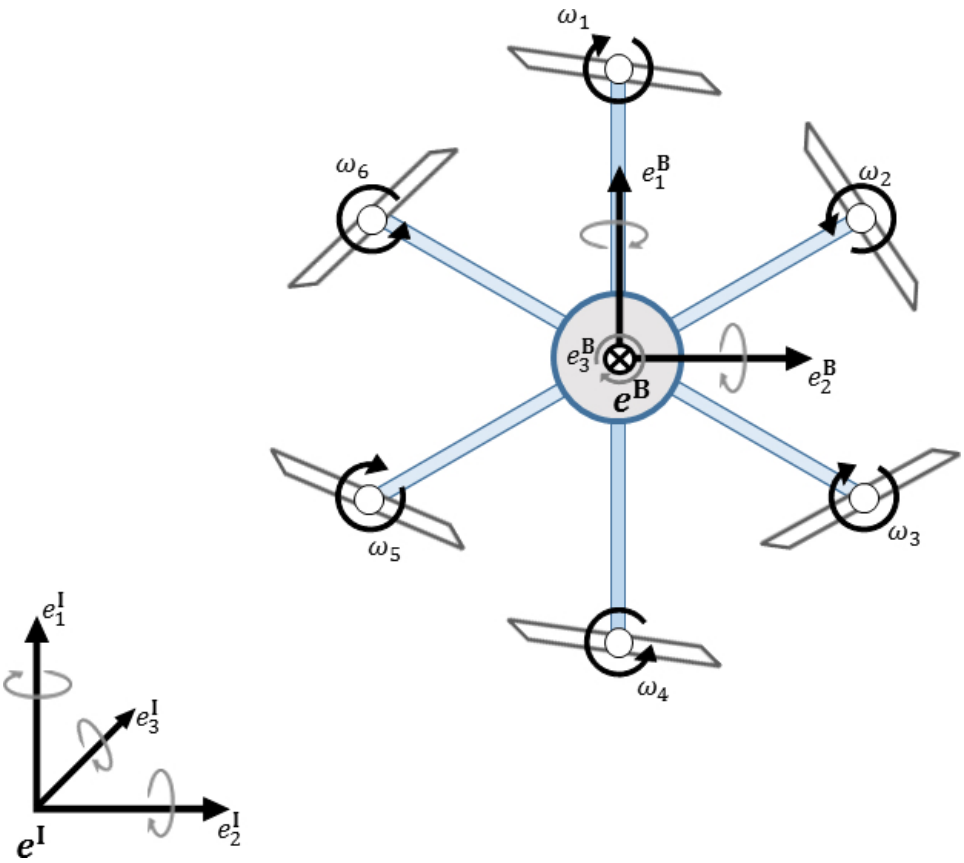

Figure 7.9: *Mechanical configuration of a hex rotor with body fixed and inertial frame [Kir15].*

cos(·) the linear velocities $(v_x, v_y, v_z)^T$ defined in the inertial frame can be obtained by $\boldsymbol{R(\Theta)} \cdot \mathbf{v}$, i.e.,

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{7.1}$$

The transformation of positions defined in the body frame into the corresponding positions in the inertial frame can be obtained by

$$\begin{bmatrix} \mathbf{P^I} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R(\Theta)} & \mathbf{P^I_{B,org}} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P^B} \\ 1 \end{bmatrix}. \tag{7.2}$$

The equations of motion are derived from the first principles (Newton-Euler laws [Bea06]) to describe both, the translational and rotational motion of the multi-rotor leading to the following discrete-time non-linear state-space model with the state vector $\mathbf{x} = \left[\, x^k \; y^k \; z^k \; u^k \; v^k \; w^k \; \phi^k \; \theta^k \; \psi^k \; p^k \; q^k \; r^k \,\right]^T = \left[\, x_1^k \; x_2^k \; x_3^k \; x_4^k \; x_5^k \; x_6^k \; x_7^k \; x_8^k \; x_9^k \; x_{10}^k \; x_{11}^k \; x_{12}^k \,\right]^T$:

$$\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \\ x_4^{k+1} \\ x_5^{k+1} \\ x_6^{k+1} \\ x_7^{k+1} \\ x_8^{k+1} \\ x_9^{k+1} \\ x_{10}^{k+1} \\ x_{11}^{k+1} \\ x_{12}^{k+1} \end{bmatrix} = \begin{bmatrix} x_1^k \\ x_2^k \\ x_3^k \\ x_4^k \\ x_5^k \\ x_6^k \\ x_7^k \\ x_8^k \\ x_9^k \\ x_{10}^k \\ x_{11}^k \\ x_{12}^k \end{bmatrix} + \begin{bmatrix} x_4^k \\ x_5^k \\ x_6^k \\ -(cx_7^k sx_8^k cx_9^k + sx_7^k sx_9^k)\dfrac{\upsilon_1^k}{m} \\ -(cx_7^k sx_8^k sx_9^k + sx_7^k cx_9^k)\dfrac{\upsilon_1^k}{m} \\ g - cx_7^k cx_8^k \dfrac{\upsilon_1^k}{m} \\ x_{10}^k \\ x_{11}^k \\ x_{12}^k \\ \dfrac{I_y - I_z}{I_x} x_{11}^k x_{12}^k + \dfrac{L}{I_x}\upsilon_2^k - \dfrac{I_R}{I_x} x_8^k g(\boldsymbol{\upsilon}) \\ \dfrac{I_z - I_x}{I_y} x_{10}^k x_{12}^k + \dfrac{L}{I_y}\upsilon_3^k - \dfrac{I_R}{I_y} x_7^k g(\boldsymbol{\upsilon}) \\ \dfrac{I_x - I_y}{I_z} x_{10}^k x_{11}^k + \dfrac{1}{I_z}\upsilon_4^k \end{bmatrix} \Delta t, \tag{7.3}$$

with $\boldsymbol{\upsilon} = (\upsilon_1, \upsilon_2, \upsilon_3, \upsilon_4)^T, \boldsymbol{\upsilon} \in \boldsymbol{\Upsilon}$ being the inputs for altitude, roll, pitch and yaw, I_x, I_y and I_z are the inertia about x-, y-$z-$axes, I_R is the rotor moment of inertia, l is the length between the center of gravity of the UAV and the center of one rotor, g is the gravitation constant, $g(\boldsymbol{\upsilon})$ is a function of $\boldsymbol{\upsilon}$ depending on the rotor's angular velocities $\omega_1 \ldots \omega_6$, m is the mass of the UAV and Δt is the sampling time. It can be seen that the angular accelerations depend only on the angular rates and the input vector $\boldsymbol{\upsilon}$, while the linear accelerations depend on the Euler angles and $\boldsymbol{\upsilon}$. Hence, the state-space model can be divided into two interlinked sub-models $M1$ and $M2$. In [Kir15] all parameter for the hex rotor system depicted in figure 7.2 are identified. The values of the parameters can be found in the following table:

Parameter	Value	Unit
m	3.305	kg
l	0.34	m
I_x	0.297584	$kg \cdot m^2$
I_y	0.297584	$kg \cdot m^2$
I_z	0.342668	$kg \cdot m^2$
I_R	$2.70253 \cdot 10^{-5}$	$kg \cdot m^2$
Thrust constant	$7.413 \cdot 10^{-6}$	$kg \cdot m$
Yaw constant	$4.3411 \cdot 10^{-7}$	$kg \cdot m$

Table 7.1: *Identified system parameters.*

The model structure is suitable for an integration of the attitude and velocity control behavior in the RNBC structure. The attitude control behavior, controlling subsystem $M1$ is ordered in the lowest (and fastest) layer and the velocity control behavior, controlling $M2$ in the second layer of the RNBC structure.

7.4 Attitude Control Layer

The attitude control behavior is realized with a non-linear attitude controller based on the back-stepping approach (BSA). The BSA is a recursive algorithm for designing stabilizing controls for a class of nonlinear dynamical systems [KKK95]. Applying the BSA [BS05], the following control inputs are obtained. The roll control input υ_2 is given by

$$\upsilon_2 = \frac{I_x}{L}(k_1 \frac{{z_1}^k - {z_1}^{k-1}}{\Delta t} - x_{11}x_{12}\frac{I_y - I_z}{I_x} - x_{11}(-\frac{I_R}{I_x})g(\boldsymbol{\upsilon}) + z_1 + k_2 z_2) \quad (7.4)$$

with

$$z_1 = x_7^r - x_7, \quad (7.5)$$

$$z_2 = x_{10}^r + k_1 z_1 - x_{10}, \quad (7.6)$$

and

$$g(\boldsymbol{\upsilon}) = \omega_1 - \omega_2 + \omega_3 - \omega_4 + \omega_5 - \omega_6. \quad (7.7)$$

The control υ_3 for the pitch angle is given by

$$\upsilon_3 = \frac{I_y}{L}(k_3 \frac{{z_3}^k - {z_3}^{k-1}}{\Delta t} - x_{10}x_{12}\frac{I_z - I_x}{I_y} - x_{10}\frac{I_R}{I_y}g(\boldsymbol{\upsilon}) + z_3 + k_4 z_4) \quad (7.8)$$

and z_3 and z_4 are defined as

$$z_3 = x_8^r - x_8 \quad (7.9)$$

and

$$z_4 = x_{11}^r + k_3 z_3 - x_{11}. \quad (7.10)$$

The input command υ_4, controlling the yaw angle is given by

$$\upsilon_4 = I_z(k_5 \frac{{z_5}^k - {z_5}^{k-1}}{\Delta t} - x_{10}x_{11}\frac{I_x - I_y}{I_z} + z_5 + k_6 z_6) \quad (7.11)$$

with

$$z_5 = x_9^r - x_9 \quad (7.12)$$

and

$$z_6 = x_{12}^r + k_5 z_5 - x_{12}. \quad (7.13)$$

Table 7.2 summarizes the parameters for $k_1 \dots k_6$ proposed by [Kir15] for the given hex-rotor system.

Parameter	Value
k_1	2
k_2	2
k_3	2
k_4	2
k_5	0.3
k_6	3

Table 7.2: *Back-stepping controller parameter set.*

7.5 Velocity Control Layer

In this section, the implementation of the linear velocity control layer from [Kir15], controlling the translational velocities in $x-$ $y-$, and $z-$direction is

presented. The attitude of the hex rotor is controlled by the back-stepping attitude controller described above, which has a very fast behavior. For the outer velocity control loop the system is simplified to a static system as described in [Voo09].

First, new input variables are defined:

$$\begin{bmatrix} x_6 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} \widetilde{U}_5 \\ \widetilde{U}_6 \\ \widetilde{U}_7 \end{bmatrix} = \begin{bmatrix} h_1(x_7^r, x_8^r, x_9^r, \upsilon_1) \\ h_2(x_7^r, x_8^r, x_9^r, \upsilon_1) \\ h_3(x_7^r, x_8^r, x_9^r, \upsilon_1). \end{bmatrix} \tag{7.14}$$

For the control of the translational velocities of the hex rotor system(static velocity subsystem $\mathbf{h}$), a proportional controller is used.

$$\begin{bmatrix} \widetilde{U}_5 \\ \widetilde{U}_6 \\ \widetilde{U}_7 \end{bmatrix} = \begin{bmatrix} k_8 \cdot (x_6^r - x_6) \\ k_9 \cdot (x_4^r - x_4) \\ k_{10} \cdot (x_5^r - x_5) \end{bmatrix} \tag{7.15}$$

The inverse of the subsystem $\mathbf{h}^{-1}(\widetilde{U}_5, \widetilde{U}_6, \widetilde{U}_7)$ has to be found to be able to find the solution for the reference attitude $[x_7^r, x_8^r, x_9^r]^T$ and thus the input υ_1. Desired translational velocities in each direction can be achieved by the hex-rotor system without changing the yaw angle. Therefore, the system is simplified by setting the reference for the yaw angle to zero ($\psi^r = x_9^r = 0$). Hence,

$$\begin{bmatrix} \widetilde{U}_5 \\ \widetilde{U}_6 \\ \widetilde{U}_7 \end{bmatrix} = \begin{bmatrix} g - cosx_7^r \cdot cosx_8^r \frac{1}{m} \upsilon_1 \\ -cosx_7^r sinx_8^r \frac{1}{m} \upsilon_1 \\ sinx_7^r \frac{1}{m} \upsilon_1 \end{bmatrix} \tag{7.16}$$

With help of the following substitutions, an analytical solution for the non-linear equations 7.16 can be found:

$$\begin{aligned} \alpha &= sin(x_7^r) \\ cos(x_7^r) &= \pm\sqrt{1-\alpha^2} \\ \beta &= sin(x_8^r) \\ cos(x_8^r) &= \pm\sqrt{1-\beta^2}. \end{aligned} \tag{7.17}$$

Thus the equation is given by

$$\begin{bmatrix} \widetilde{U}_5 \\ \widetilde{U}_6 \\ \widetilde{U}_7 \end{bmatrix} = \begin{bmatrix} g - ((\pm\sqrt{1-\alpha^2}) \cdot (\pm\sqrt{1-\beta^2}) \cdot \frac{1}{m} \upsilon_1) \\ \mp\sqrt{1-\alpha^2}\frac{\beta}{m}\upsilon_1 \\ \frac{\alpha}{m}\upsilon_1 \end{bmatrix} \tag{7.18}$$

and the solution yields for $\widetilde{U}_6 \neq 0$

$$\beta = \pm \left[\left(\frac{g - \widetilde{U}_5}{\widetilde{U}_6} \right)^2 + 1 \right]^{-\frac{1}{2}} \tag{7.19}$$

$$\upsilon_1 = \pm m \cdot \sqrt{\frac{\widetilde{U}_6^2}{\beta^2} + \widetilde{U}_7^2} \tag{7.20}$$

$$\alpha = \widetilde{U}_7 \cdot \frac{m}{\upsilon_1} \tag{7.21}$$

The solution for υ_1 gives two possible solutions but since the value has to be always positive, because no negative thrust can be created by the propellers, a possible negative solution can be neglected. Thus, only one admissible solution for υ_1 exists while $x_7^r = arcsin(\alpha)$ is in the interval $[-\pi/2, +\pi/2]$. Regarding the second line in equation 7.16, the sign of x_7^r gives the sign of x_8^r because if x_7^r is negative, x_8^r will be positive and vice versa. Hence, there is also only one solution for $x_8^r = arcsin(\beta)$ left.

Next, the reference values $[x_7^r, x_8^r, x_9^r]^T$ are transformed to reference angles and passed to the attitude controller, which results in the desired velocity. In case of $\widetilde{U}_6 = 0$ the analytical solution can be obtained by:

$$\beta = 0 \tag{7.22}$$

$$\upsilon_1 = m \cdot \sqrt{(\widetilde{U}_5 - g)^2 - \widetilde{U}_7^2} \tag{7.23}$$

$$\alpha = \widetilde{U}_7 \cdot \frac{m}{\upsilon_1}. \tag{7.24}$$

7.6 Collision Avoidance

For the sake of completeness, the collision avoidance layer is presented here, too. A real-system implementation is not accomplished yet, but simulations of PEGs in an environment with static and moving obstacles are provided in this section.

7.6.1 The Repulsion Force Approach

In [Ale+13] a comparison between several collision avoidance techniques for UAVs is conducted, using the same dynamical model for a multi-rotor as used in this work. The repulsion force approach provided the best results regarding obstacle avoidance and complexity of the algorithm. Therefore, this approach is the first choice here. As proposed in [Ale+13] the algorithm computes a repulsive force to each obstacle i according to the formula

$$F_i = sat\left[\cos\alpha_i \left(\frac{c_1 + c_2 \mathbf{v}^{rq}}{r_i - d^S}\right)^2\right], \tag{7.25}$$

while $sat(\cdot)$ constrains to the maximal possible repulsion force. The angle to an obstacle i is denoted by α_i, $\mathbf{v}^{rq} = \dfrac{\mathbf{v}^r}{\mathbf{v}^q_{max}}$ is the quotient of the relative velocity vector $\mathbf{v}^r$ to the obstacle i and the maximum velocity vector of the UAV $\mathbf{v}^q_{max}$, r_i is the distance to the obstacle i, d^S is the safety distance in which the repulsion force is at its maximum (fig. 7.10). The two parameters c_1 and c_2 are used for repulsion force calibration at zero and maximum velocity, respectively. Very satisfying results are achieved in the simulations with $c1 = c2 = 10$. The sum of all repulsion forces for n obstacles is given by

$$F_{sum} = \sum_{i=1}^{n} F_i(r_i, \alpha_i, \mathbf{v}^r). \tag{7.26}$$

A function $g : \Re^3 \to \Re^3$ provides a velocity vector $\mathbf{v}^o$ to be added to $\mathbf{v}^{pe}_r$, hence $\mathbf{v}^o = g(F_{sum})$, and thus $\mathbf{v}^{cf}_r = \mathbf{v}^{pe}_r + \mathbf{v}^o$.

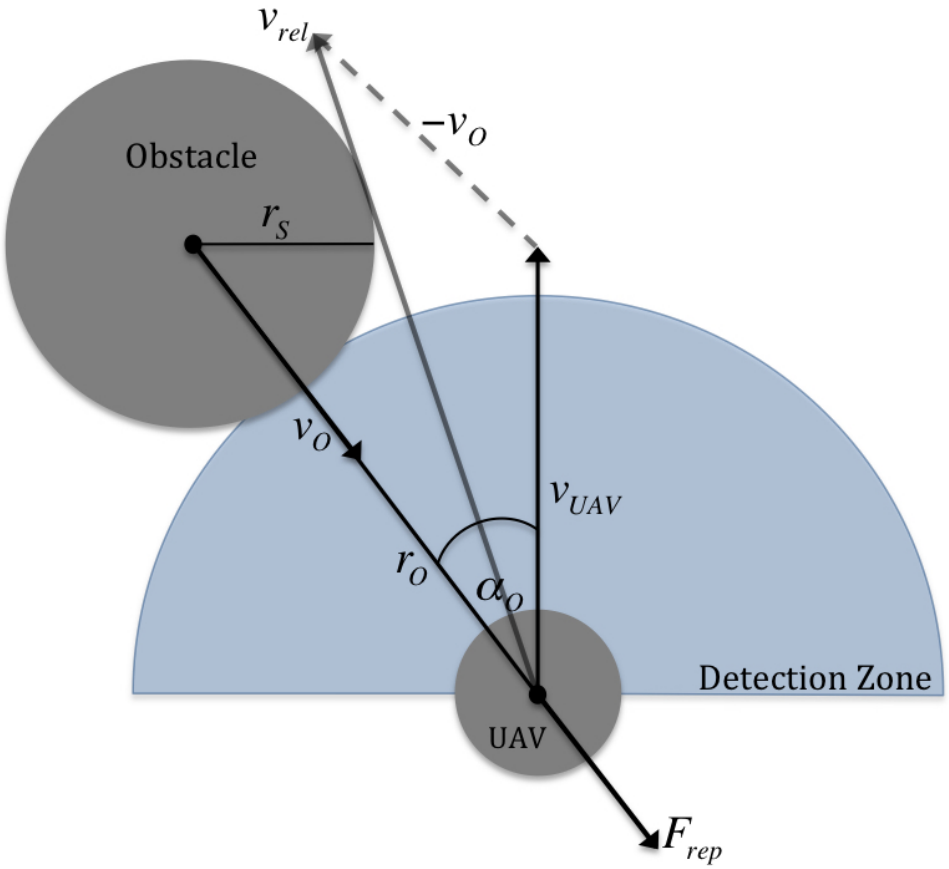

Figure 7.10: *Repulsion force principle.*

7.6.2 Simulations

Simulation Set-Up

- Since the strategies represent a velocity change in three linear directions of p and e a maximum velocity $\mathbf{v}^p_{max} = \begin{bmatrix}15 & 15 & 3.5\end{bmatrix}^T$ and $\mathbf{v}^e_{max} = \mathbf{v}^p_{max} \cdot 1.5^{-1}$ and a maximal absolute value of $v^p_{maxA} = 15$ for the pursuer and $v^e_{maxA} = 10$ for the evader are defined.
- The numerical solution of the PEG is computed by solving it for each initial positions $(x^1, y^1, z^1) \in X \times Y \times Z$, while x^1 and y^1 take integer values in a 61×61 grid, with $X = [-30, 30]$ and $Y = [-30, 30]$ in the pursuers reference frame. In each simulation, the initial altitude of

both UAVs is 20, i.e., $z^1 = -20$. This is necessary for the visualization of the value function.

- $s = 6$ is chosen, meaning that each player has 7^3 strategies available in each time step k.
- The stage duration is chosen to be $\Delta T = 20 \cdot \Delta t$, while $\Delta t = 0.005$ is the period time of the velocity control behavior.
- A capture distance $d_\epsilon = 5$ is chosen, since it is the maximum change in distance, which can be achieved in $\Delta t = 1$ regarding $v^p_{maxA} = 15$ and $v^e_{maxA} = 10$.
- For the collision avoidance a detection radius of $r^d = 10$ and a desired safety distance of $d^S = 3$ to the obstacles are assumed. The collision avoidance approach is evaluated as follows: A safety distance violation is detected if d^S is undercut, while undercutting $d^C = 1$ results in a collision. Note, that all obstacles have a spherical shape and that the pursuer and the evader are not considered as obstacles.

PEG with Unknown Static Obstacles

The upper part of figure 7.11 depicts the value of stages needed for capture. Since p is superior regarding the maximum velocity, it is always able to catch e. The value function of the stages depicted on the lower part of figure 7.11 shows the result of the same PEG, but with four static obstacles (blank nodes in the grid). The value is identical to the upper one where the collision avoidance did not interfere. It can be observed, that starting at initial positions forcing e towards an obstacle yields in smaller values, while initial positions beside or behind the obstacles lead to much higher values. Still, p is always able to catch e. While there is not even a d^S violation on pursuers' side, no collision happened for p and e. Only e suffered from some d^S violations when starting in front of an obstacle.

PEG with a Unknown Moving Obstacle

In this simulation both players had to face a moving obstacle on a frontal collision course. The resulting trajectories are depicted in figure A.10. The

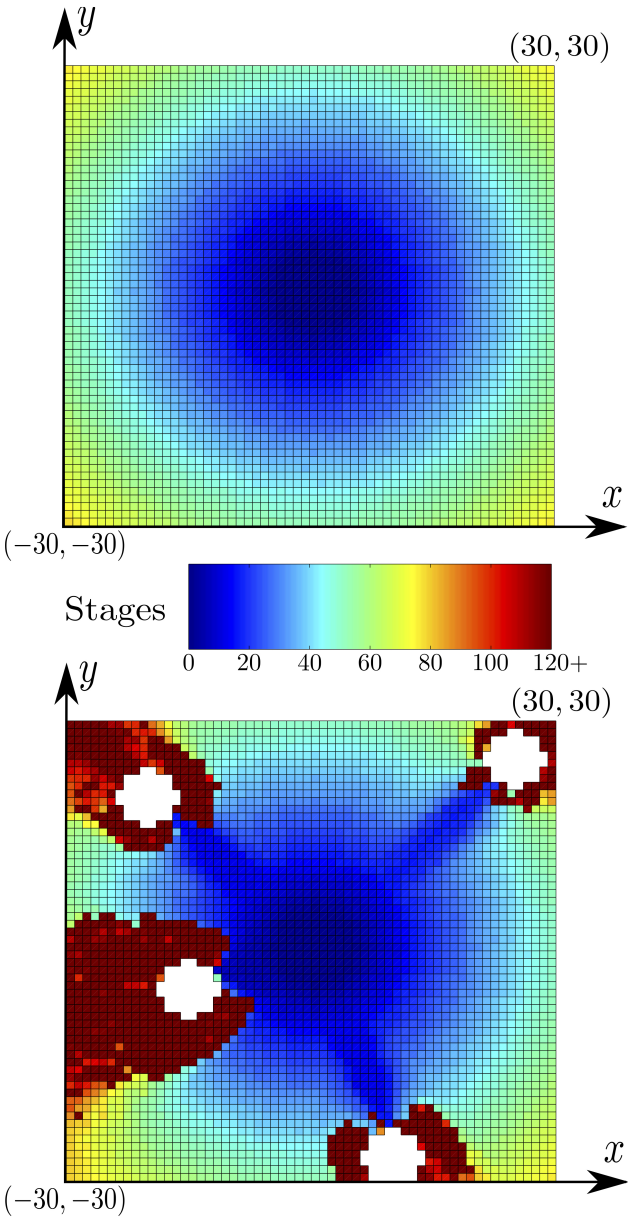

Figure 7.11: *Values of stages required for capture.*

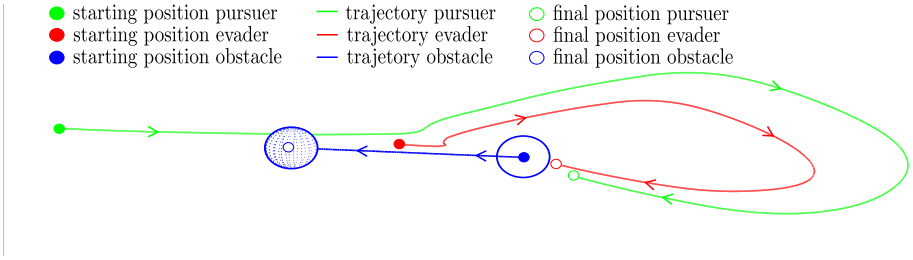

Figure 7.12: *Pursuit-evasion game with a moving obstacle.*

initial position is $(x^1, y^1, z^1) = (30, 28, 20)$, while the obstacle is moving on a straight line from $(40, 38, 30)$ to $(20, 18, 20)$ with a velocity of $v^o = 10$ in the inertial frame. In contrast to 74 stages in the obstacle-free case, the value of this game is 136 stages.

7.6.3 Conclusion

The convergence of the PEG in a three-dimensional environment with UAV agents having dynamical constraints is shown. Due to the collision avoidance behavior the convergence is also achieved in presence of previously unknown obstacles. Despite the value of the game increases in the most cases where the agents had to face an obstacle, the imminent collisions could always been prevented. Since [Ale+13] shows that for the same system configuration the proposed repulsion force mechanism could prevent the UAV of colliding in a general three-dimensional environment with static and dynamic obstacles, pursuit and evasion is applicable in an unknown environment.

7.7 Pursuit-Evasion Layer

The implementation of the two-player PEG with UAVs, defined in section 4.1.1, on an embedded computer is briefly described hereafter. For that reason, an algorithm is used, namely NPG, which enables the determination of Nash equilibria in mixed strategies of N-player games.

Algorithm for N-Player Nash-Equilibrium in Mixed Strategies

As described above, an optimal control action tuple $\mathbf{u}^{k*} = (\mathbf{u}_p^{k*}, \mathbf{u}_e^{k*})$ for the agents p and e in stage k of the PEG is derived by the determination of the Nash equilibrium. The MATLAB-function NPG [Cha10] is able to solve an *N-player finite non-cooperative game* by computing one Nash equilibrium in mixed strategies. Thereby, the optimization formulation of a *N-player non-cooperative game* according to [Cha09] is used for computation. The function uses the *sequential quadratic programming* based *quasi Newton method* to solve a non-linear minimization problem with non-linear constraints.

Since it is not feasible to generate C code of the NPG function automatically, the algorithm to compute one Nash equilibrium has been implemented from scratch in C to be applicable on the embedded computer. For that, the NLopt package [Ste13] is utilized to solve the non-linear minimization problem, more precisely the *SLSQP* [Kra88; Kra94] algorithm included there.

Moreover, an open-source C library for computing Nash equilibria in mixed strategies with the Lemke-Howson algorithm is used for the implementation of the MTPGD game.

7.7.1 Validation of the Algorithm

The correctness of the algorithm implementation is validated by running the same simulations on MATLAB and on the embedded computer and the comparison of the resulting value functions. Therefore, the two-player game from section 4.1.1 is chosen with the following set-up:

- Since the chosen optimal control actions represent a velocity change in three linear directions of p and e, a maximum velocity $\mathbf{v}_{max}$ with $\mathbf{v}_{max}^p = \begin{bmatrix}15 & 15 & 3.5\end{bmatrix}^T$ and $\mathbf{v}_{max}^e = \frac{\mathbf{v}_{max}^p}{1.5}$ and a maximal absolute value of $v_{maxA}^p = 15$ for the pursuer and $v_{maxA}^e = 10$ for the evader is defined.
- The numerical solution of the PEG is computed by solving it for each initial positions $(x^1, y^1, z^1) \in X \times Y \times Z$, while x^1 and y^1 take integer values in a $61x61$ grid, with $X = [-30, 30]$ and $Y = [-30, 30]$ in the pursuers' reference frame (pursuers' position is the origin). In each

simulation, the initial altitude of both UAVs is 20, i.e., $z^1 = -20$. This is necessary for the visualization of the value function.

- $s = 6$ is chosen, i.e., each player has 7^3 strategies available in each time step k.
- The stage duration is chosen to be $\Delta T = 0.1$, while the velocity control is sampled with $\Delta t = 0.005$. The real-time specification to be satisfied by the embedded computer is $\Delta T = 0.1$s for one stage k.
- A capture distance $d_\epsilon = 5$ is chosen, since it is the maximum change in distance, which can be achieved in $\Delta t = 1$ regarding $v^p_{maxA} = 15$ and $v^e_{maxA} = 10$.

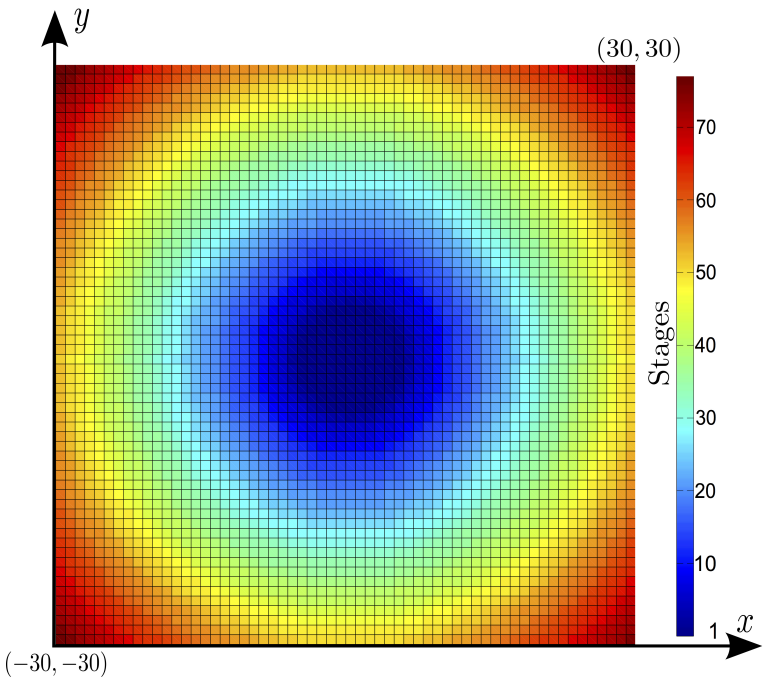

Figure 7.13: *Value of stages needed for capture (embedded computer).*

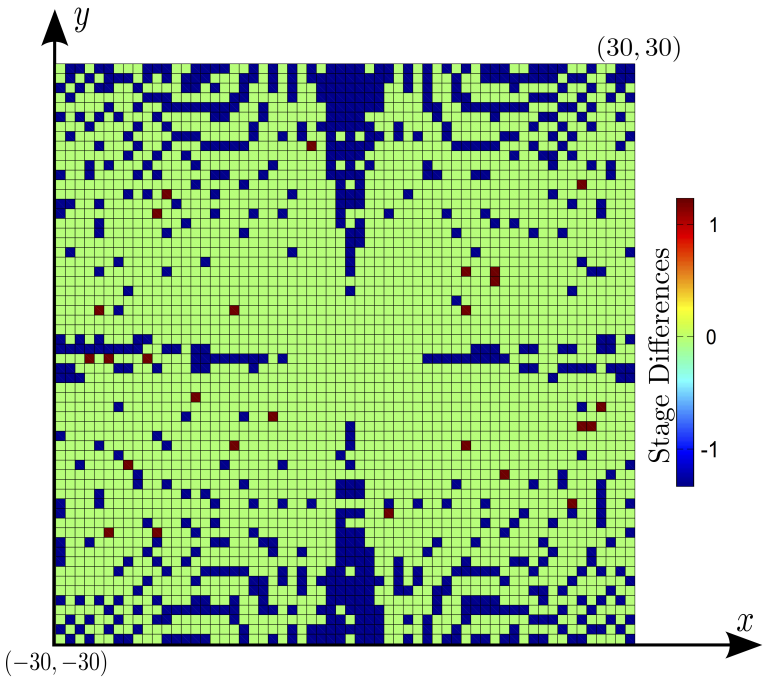

Figure 7.14: *Difference of value of stages needed for capture in* MATLAB *and on the embedded computer.*

Figure 7.13 depicts the value function over the regarded discretized state space computed by the embedded computer. Regarding this solutions the convergence of the PEG in three dimensions is given everywhere, meaning that in this configuration the evader can never avoid to be captured by the pursuer. Figure 7.14 depicts the difference of the value of stages between the MATLAB simulation and the simulation on the embedded computer. The differences are slightly in the whole state space. Moreover, due to the very small differences (caused by possible rounding errors and varieties in the minimization algorithm implementation) between the MATLAB and the embedded computer solution, the implementation on the BeagleBone Black is accomplished successfully. The next important point is to check the real-time applicability of the approach. The demanded computational time of $\Delta T = 0.1$s for one stage of the game is successfully satisfied. By configuring the algorithm for the saddle-point computation of one stage k, such that it stops after maximal 0.09s, the minimization algorithm is still able to maintain the demanded absolute tolerance of 10^{-6} for the minimum func-

tion value. The use of an RTOS assures that the algorithm yields a solution within $\Delta T = 0.1$s, thus the real-time specifications are satisfied.

7.8 Cooperative Behavior Assignment Layer

In this layer the behavioral strategy of each pursuer is determined. This layer demands the possible outcomes when playing all possible combinations of behaviors by the underlying layer, the pursuit-evasion game. Having those possible outcomes, a static cooperative game as described in chapter 5 has to be solved. The algorithm for the computation of e.g. the pareto efficiency solution is relatively trivial for static games and follows the description in chapter 3. A more crucial task of this layer is the data transfer and the synchronization between the pursuers. Therefore, the embedded computer responsible for the cooperative behavior assignment is equipped with an RF module. This module is used as communication interface between the pursuers, while one of the pursuers is declared as the master, and is responsible for the synchronization and the data transfer between all teammates. The master sends out a specific bit sequence in every time cycle. When this bit sequence is received by the other pursuers it triggers them to broadcast their information set, while the master broadcasts its own one. In real experiments, it is determined that the computation of one stage of a two-pursuer one-evader game with observation sharing over a serial RF connection takes about $120ms$.

8 Practical Real-Time Experiments

8.1 Experimental Set-Up

To prove the feasibility of the algorithms developed in this work, real system experiments are carried out. In particular, the single-action game decomposition and the MTPGD presented in chapter 4 is used to solve the non-cooperative games, while for the cooperation team-behavior game of the pursuers the Pareto Efficiency approach presented in chapter 5 is utilized. Five different two-pursuer one-evader PEGs is defined. The first three experiments are situations, where all players have perfect state information. Moreover, the pursuers are faster than the evader. Next, in experiment four, a situation with imperfect state information and observation sharing is regarded while all players have equal maximum speed. The initial positions of the players are chosen such that the evader is supposed to be caught as shown in section 5.3.1. In the last experiment the same configuration as in experiment four is assumed, but with a faster evader.

Due to the fact, that only one test platform is available (chapter 7), each experiment is carried out for each player separately. Therefore, a total number of 15 experiments with two emulated players are applied. Due to safety reasons the PEGs take place in the plane, while the altitude is held to a fixed value. The stage time Δt for the non-cooperative game is set to $0.2s$. Normally a stage duration of $\Delta t > 120ms$ is demanded, because of the observation sharing when using the communication hardware described above, but since two emulated players are used here, this is not the reason to raise

the stage duration. The stage duration is set to $\Delta t = 0.2s$ because the GNSS provides a position update with a frequency of $5Hz$ being the limiting factor here. The capture radius is set to $3m$ for each experiment.

Figure 8.1 depicts the experimental set-up. The GNSS base station, mounted on a tripod, is depicted on the left. The experiments take place within a radius of $30m$ (free space) to the base station. The differential measurements are carried out with data of 17-20 satellites for each experiment. While the middle of the figure shows the autonomous agent described in the last chapter, the right side depicts the wind measurement station. A simple compass, a windsock, and an anemometer (figure 8.2) enabled the measurement of the wind direction and the average wind strength during an experiment.

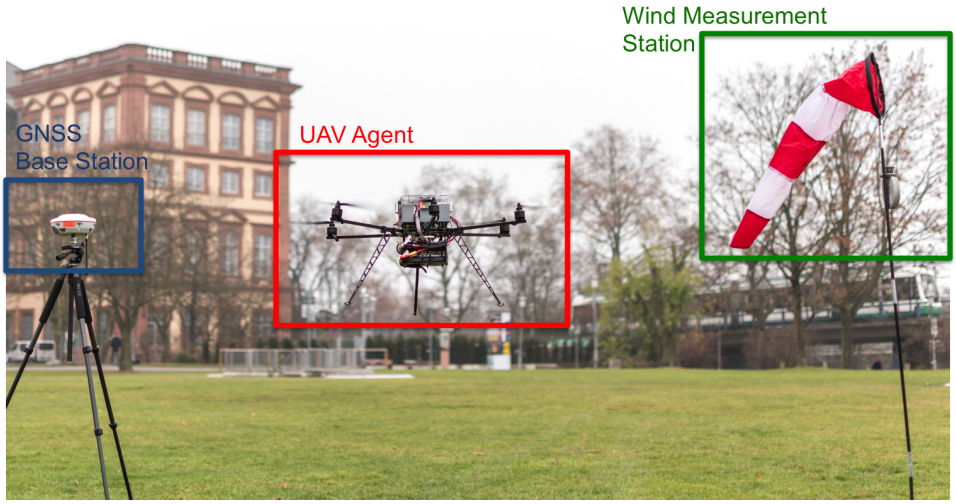

Figure 8.1: *Experimental set-up.*

The following tables summarize the configuration of each experiment:

Figure 8.2: *Anemometer Windmaster 2 [Wm2].*

Player	**Init. Position**	**Max. Speed**	**Visibility**	**Wind**
Evader	$(0, 0)$	$1.33\frac{m}{s}$	Inf	$1\frac{m}{s}$ SE
Pursuer 1	$(-5.5m, -5m)$	$2\frac{m}{s}$	Inf	$1.5\frac{m}{s}$ NW
Pursuer 2	$(-0.5m, -5m)$	$2\frac{m}{s}$	Inf	$0.5\frac{m}{s}$ NW

Table 8.1: *Experiment 1: set-up.*

Player	**Init. Position**	**Max. Speed**	**Visibility**	**Wind**
Evader	$(7.5m, 6m)$	$1.33\frac{m}{s}$	Inf	0
Pursuer 1	$(0, 0)$	$2\frac{m}{s}$	Inf	$1.6\frac{m}{s}$ NE
Pursuer 2	$(2m, 10m)$	$2\frac{m}{s}$	Inf	$1\frac{m}{s}$ N

Table 8.2: *Experiment 2: set-up.*

Player	**Init. Position**	**Max. Speed**	**Visibility**	**Wind**
Evader	$(8.5m, -7.5m)$	$1.33\frac{m}{s}$	Inf	0
Pursuer 1	$(15m, -15m)$	$2\frac{m}{s}$	Inf	0
Pursuer 2	$(0, 0)$	$2\frac{m}{s}$	Inf	0

Table 8.3: *Experiment 3: set-up.*

Player	**Init. Position**	**Max. Speed**	**Visibility**	**Wind**
Evader	$(0, 0)$	$2\frac{m}{s}$	4	$0.1\frac{m}{s}$ W
Pursuer 1	$(-3.5m, -3.5m)$	$2\frac{m}{s}$	4	$1\frac{m}{s}$ N
Pursuer 2	$(11.5m, 11.5m)$	$2\frac{m}{s}$	4	$0.5\frac{m}{s}$ NW

Table 8.4: *Experiment 4: set-up.*

Player	Init. Position	Max. Speed	Visibility	Wind
Evader	$(3.5m, 3.5m)$	$2\frac{m}{s}$	4	$0.2\frac{m}{s}$ W
Pursuer 1	$(-15m, -15m)$	$1.8\frac{m}{s}$	4	$0.5\frac{m}{s}$ E
Pursuer 2	$(-11.5m, -11.5m)$	$1.8\frac{m}{s}$	4	$1\frac{m}{s}$ S

Table 8.5: *Experiment 5: set-up.*

8.2 Experimental Results

In this section, the results of five selected experiments are presented. The player's trajectories of the remaining experiments can be found in the appendix.

Firstly, in figure 8.3 the trajectories of the three players for experiment one can be seen, while the evader is the actual player. Since both pursuers are initially positioned behind the evader, both are playing the "pursue" behavior through out the game and are able to catch the evader after 35 stages ($7s$). The real system had to deal with wind coming from south-east direction with a velocity of $1\frac{m}{s}$. Since the evader is heading in north-east direction it had to struggle with relatively strong upwind in east direction.

In the second experiment depicted on figure 8.4, the effect of the wind can be observed even more. Regarding the velocity vectors of pursuer one, being the actual UAV in this experiment, one can see that the demanded speed and direction cannot be maintained. During the experiment a strong upwind from north-east is blowing with a velocity of $1.6\frac{m}{s}$ causing, that unlike in the simulations, the other pursuer catches the evader. Moreover, it can be observed, that pursuer 2 is herding the evader most of the game, until catching him after 73 stages ($14.6s$).

While no wind is present in experiment three (figure 8.5), another reason affects the trajectory of the real UAV. In the other two runs of this experiments with the other players being the actual UAV, trajectories close to the simulation are achieved. Here, pursuer two is acting differently. At the beginning of the experiment, some measuring errors of the GNSS system cause a temporary position inaccuracy of about $1m$, causing the UAV to drift away in the first stages of the game. Nevertheless, pursuer two is able to catch the evader after 79 stages ($15.8s$).

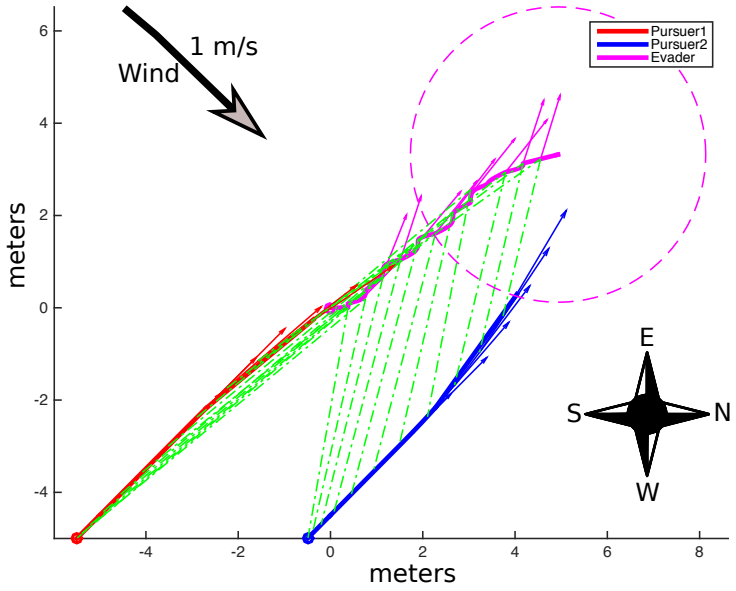

Figure 8.3: *Experiment 1: actual UAV evader.*

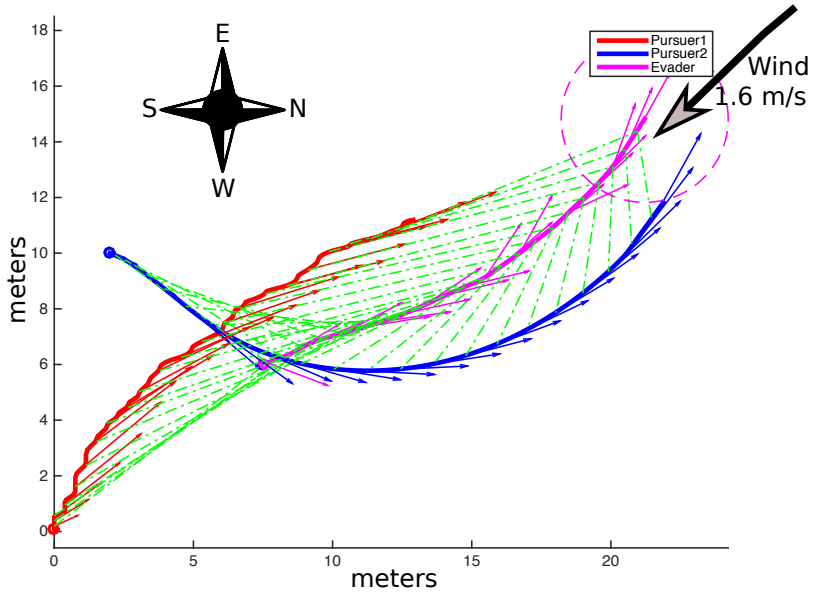

Figure 8.4: *Experiment 2: actual UAV pursuer* 1.

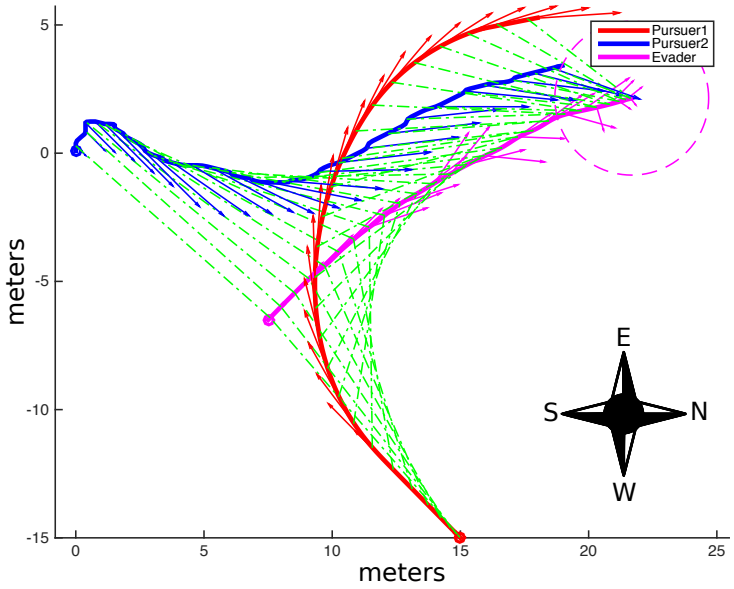

Figure 8.5: *Experiment 3: actual UAV pursuer* 2.

In the following experiment, the evader is initially only escaping from pursuer one. Since, a visibility radius of $4m$ for each player is assumed, the evader does not know about the second pursuer yet. The second pursuer knows about the position of the evader due to the observation sharing with pursuer one. In figure 8.6 it can be seen, that pursuer one is herding the evader towards the other pursuer. After entering the visibility radius of the evader, the latter has no chance to escape from pursuer two anymore, while all players have equal maximal speed. The capture of the evader occurs after 30 stages ($6s$). The south wind with a velocity of $0.1\frac{m}{s}$ has no noticeable effect on the UAV.

In the last experiment, depicted on figure 8.7, the biggest difference to the expected behavior of the UAVs can be regarded caused by wind. Here, the upwind (south wind) with a velocity of $1\frac{m}{s}$ slows down the second evader in such a way, that the evader, which is 10% faster than the pursuers, is able to escape both UAVs. In the other two runs, with the other players being the real UAV, the evader is caught as expected.

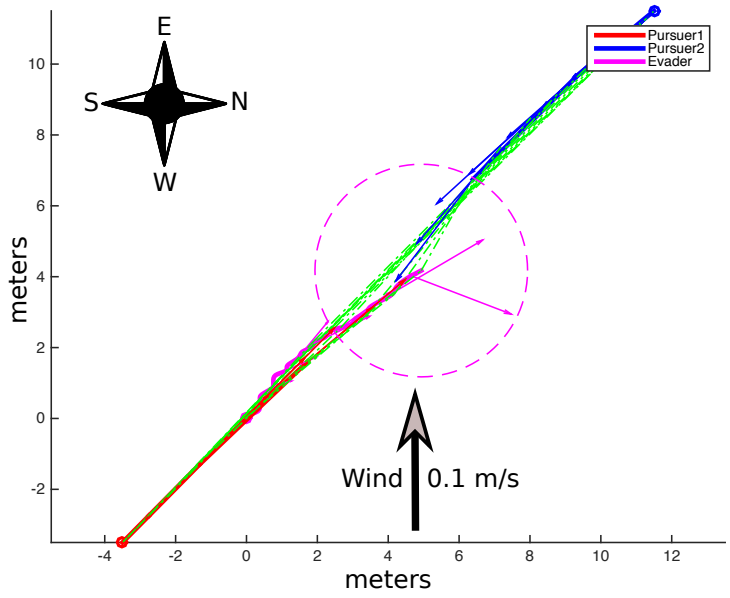

Figure 8.6: *Experiment 4: actual UAV evader.*

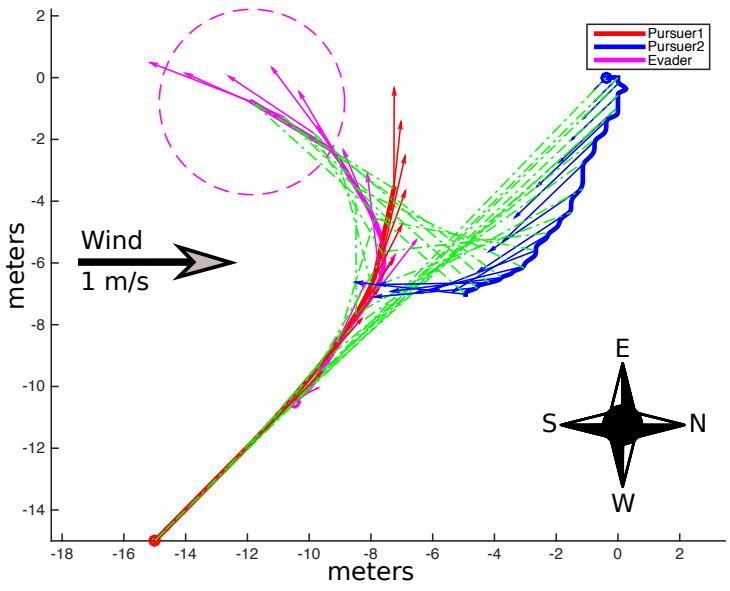

Figure 8.7: *Experiment 5: actual UAV pursuer* 2.

8.3 Conclusion

Regarding the experimental results, the applicability of the approaches presented in this work are shown. All 15 experiments are carried out successfully, meeting the expected trajectories from the simulations in almost every case. Despite the fact that the simulation model of the game does not meet the actual system with 100%, there are two factors that influence the reference trajectories for the actual game. Firstly, it is observed that to a certain degree, sensor uncertainties, e.g. a degraded satellite signal, are leading to inaccurate measurements and thus to a deviating behavior. The second and more severe reason is the disturbances caused by wind and wind gusts. While the average wind throughout an experiment is measured, the wind gusts are not registered. The observations during the experiments reveal, that relatively strong wind gusts with a velocity over $2\frac{m}{s}$ affect the behavior of the UAV strongly. As is evident from the experimental results, especially when the UAV is facing an upwind with an average velocity greater than $0.5\frac{m}{s}$, it is significantly decelerated.

9 Conclusion and Final Remarks

A framework enabling the application of multi-pursuer single-evader PEG with actual UAVs in real-time is presented. The problem is defined as a multi-player discrete-time deterministic dynamic game with non-fixed terminal time. The complexity of the solution process increases rapidly with the number of players and strategies. Therefore, a suitable method is sought, reducing the time-complexity such that a real-time implementation is feasible. The reduction takes place on two different levels of the game. While computing the result of a single stage of the game, the action set is decomposed into three distinct games. Hence, the optimal strategies for a motion in x-, y-, and z-direction are calculated independently. The decomposed single-stage game yields very close results to the full game, while significantly reducing the time-complexity. In other words, the difference between the results of the decomposed and the full game have an average value of 0.039 stages and a standard deviation of 0.37 stages which is negligibly small for all practical purposes. The time complexity of the full game, being $\mathcal{O}(s^6)$, is reduced to $3 \cdot \mathcal{O}(s^3)$, where s is the number of strategies for each degree of freedom. The reduction of the exponential complexity is, thus, significant. For instance, the solution calculation for a game with $s = 9$ is performed 171.17 times faster through decomposition of the action space. The second approach for time-complexity reduction is changing the game structure. Two different methods are introduced, namely the MTPGD and TSTPG, which are applied to a two-pursuer single-evader game. In the MTPGD, the original game is decomposed into many two-player games, for which less complex algorithms exist, while the cost functional of each player in the

two-player game has a constant term considering the actual state of each other players of the overall game. Due to a fusion of those intermediate results, a close solution to the solution of the full game is achieved. Depending on the cardinality of the action space the time-complexity could be decreased considerably (From $\mathcal{O}(S^{3N})$ to $\mathcal{O}(NS^6)$, where N is the number of players and S is the number of strategies per player)). Therefore, this decomposition is most beneficial for a larger number of players. The TSTPG encapsulates a team of players to one super pursuer, while the action set of this super player consist of each possible combination of the players of the team. While the results using this approach are even closer to those of the full game, the time-complexity of the utilized solution algorithm stays the same ($\mathcal{O}(NS^6)$). Nevertheless, other algorithms that are only available for two-player games, like the Lemke-Howson algorithm, enable a computation in PTIME for zero-sum games to be used. In simulation it is shown that using the Lemke-Howson algorithm results in a huge reduction of the run time for both the MTPGD and the TSTPG, being considerably faster (≈ 21 times faster with TSTPG and ≈ 70 times faster with MTPGD for $S = 9$) in opposition to the computational time of the full game. In total, the action space decomposition and the complexity reduction with the TSTPG result in a 3558.62 times faster calculation, and with the MTPGD a 12043.52 times faster calculation, with 9 strategies per axis and player.

Moreover, it is demanded that a team of pursuers is able to capture a single faster evader through cooperation. Therefore, a superordinate cooperation on top of the non-cooperative game is proposed, enabling the change of behavior for a pursuer. Two pursuing behaviors are defined, namely *pursuit* and *battue*. The first behavioral strategy is a conventional pursue, while the second one is a behavior where the pursuer is trying to drive the evader towards another pursuer. It is shown that due to the cooperation on a higher level, the outcome of the pursuing team is increased (up to 13.25%) for a game with perfect information structure and a slower evader. While the gain of the outcome of such a game is not significant, it is when it comes to games with imperfect information e.g. visibility-based PEGs. Due to the observation sharing between the pursuers, it is only necessary that one pursuer is able to sense the evader. This information superiority enables the capture of a faster evader. This is shown due to simulations. With increasing evader

speed the initial states for which the evader can be captured increasingly shrinks. Another issue is that because of the observation sharing, some information may be received by other pursuers with a delay of multiple stages. It is shown that when using a Kalman filter for the estimation of the actual state, a capture of the evader is still possible with a delay of 6 stages. Without an estimator, a delay of 4 steps makes a capture impossible.

The results obtained are demonstrated in practice using UAVs. The UAV agents behaviors are encapsulated in the RNBC structure. This architecture is a generalized cascaded control structure which arranges the fastest and more reflexive behaviors in the lower layers, while the slower and more complex behaviors are placed at the upper levels. Using this implementation for UAVs, a comparison to the PMA from [RT07] is performed. This work, focuses on a similar topic, namely the time-complexity reduction of the game and the capture of a faster evader with multiple pursuers. In order to compare the approaches in a fair way, some restrictions to the GTA are set, like the fixed and finite set of position states the game can reach. This impacts especially the termination condition. The game ends when the evader reaches the boundary of the position state. Moreover, a performance measure is introduced including the average of terminal stages, the variance of terminal stages, the diverged games and the operations needed for calculation of the regarded approach combination. The two methods are compared by running a large number of simulations for three different configurations: (i) faster pursuers, (ii) equally fast players, and (iii) a faster evader. The results show the advantage of the escaping and pursuing behavior of the GTA of the present work. The biggest advantage can be seen in the run-time of the GTA, which is about 21 times faster regarding the collectivity of the operations conducted in all simulations.

With the successful realization of an actual PEG UAV it is possible to run several experiments. The simulation results show the feasibility of a two-pursuer one-evader scenario. Sensor uncertainties and especially wind and wind gusts affect the results, leading partly to deviations in respect to the simulation results. Most notably, strong upwind could cause a completely different outcome in opposition to the simulation, while in one experiment the evader which is supposed to be caught, is able to escape.

Finally, few remarks are due. The action space decomposition does not take the non-holonomic constraints of the hex rotor into consideration. In this specific example, the impact on the dynamic behavior is acceptable. A motion of a multi rotor in $x-$ or $y-$direction is achieved by changing the pitch or roll angle and the thrust. For instance, a motion only in a positive x-direction demands a positive pitch angle and a proper additional positive thrust. Without a proper thrust, the multi rotor would also perform a motion in $z-$direction. The roll and pitch angles are independently controllable but to maintain the $z-$velocity an additional thrust must be provided. Due to this coupling, the maximal allowed velocity commands in $z-$direction in the PEGs are kept about four times less than those for x and y. The action space decomposition can only be performed for holonomic systems, or for special non-holonomic systems (like multi rotors) by proper adaptations on the action space.

Changing the game structure is a very critical task. The choice of suitable cost functionals is especially crucial. The procedure of how to constitute the cost functional for a TSTPG or MTPGD game structure varies with the application and needs a detailed analysis of the process to be regarded. When using the MTPGD a suitable fusion of the interim solutions must also be performed. The fusion procedure also varies with the application and demands extensive investigation of the specific problem.

The superordinate cooperation gives the members of a team the ability to negotiate about which behavioral strategy is the best one to use to maximize the outcome of the team. Thus, the superordinate cooperative provides more flexibility to a team regarding their behavior. It is shown that the results of a game with perfect state information can be improved, enabling the possibility to capture a faster evader in visibility-based PEGs. The drawback of this approach is the increased complexity to the solution process. The complexity increases by a factor of n^S with every additional behavioral strategy, S, with n as is the number of players in a team.

The cooperation between team-members demands a permanent communication between team-members. The present solution takes $120ms$ for one full data exchange between two players. With an increasing number of players the communication overhead gets even bigger. Thus, a delay of multiple stages can occur. To overcome this problem, a space-saving encoding could

be implemented or another communication structure or channel could be used.

Future Work

First and foremost the n-pursuer m-evader PEGs is still a little examined topic in current research. In 1-pursuer m-evader PEGs a similar approach as discussed in chapter 5 could be applied, a cooperative evasion against a faster pursuer for instance. The evaders could have several behavioral strategies, e.g. "evade", "hide" or "attract". A conceivable strategy could be that one evader tries to attract the pursuer, such that the other evaders are able to escape or to hide. When it comes to n-pursuer m-evader PEGs the problem gets much more complex. The complexity increases because both teams are able to cooperate. Especially the pursuers have to decide which evader to pursue. A suggested approach could be by adding an additional level to the RNBC structure of the agents superordinate to the cooperative layer, namely the "Assignment Layer" as depicted on figure 9.1. Analogical to the cooperative layer, a static cooperative game could be defined to determine which pursuer combinations is the optimal one against each evader. The result of this cooperative game would be to determine optimal single-pursuer multiple-evader or multiple-pursuer single-evader combinations, considering all behavioral strategies on the lower layer, to maximize the outcome of the team. Of course, with this additional layer the total game becomes on the one hand applicable on n-pursuer m-evader PEGs, while on the other hand the time-complexity increases exponentially with the number of possible player combinations. Therefore, an accurate analyzation has to be carried out. As a pre-step to the assignment game, an elimination of non-reasonable combinations could be carried out to minimize the combination space and thus the time-complexity of the solution process of the game.

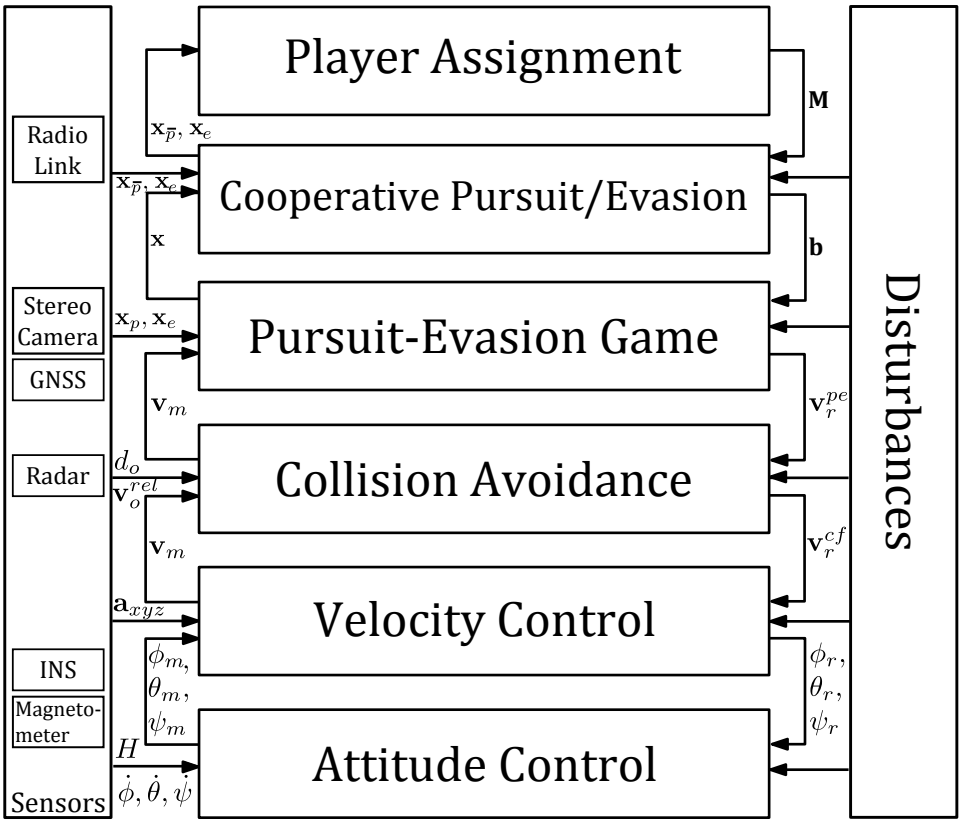

Figure 9.1: *Exemplary RNBC implementation for PEG UAV agents with player assignment.*

Bibliography

[CHI11] T. H. Chung, G. A. Hollinger, and V. Isler. „Search and pursuit-evasion in mobile robotics“. In: *Autonomous Robots* 31.4 (2011), pp. 299–316. ISSN: 1573-7527. DOI: 10.1007/s10514-011-9241-4.

[Mor+05] S. Morris et al. *Cooperative Tracking of Moving Targets by Teams of Autonomous Unmanned Air Vehicles*. Defense Technical Information Center, 2005.

[AGI08] F. Amigoni, N. Gatti, and A. Ippedico. „A Game-Theoretic Approach to Determining Efficient Patrolling Strategies for Mobile Robots“. In: *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*. WI-IAT '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 500–503. ISBN: 978-0-7695-3496-1. DOI: 10.1109/WIIAT.2008.324.

[BBH14b] S. Bhattacharya, T. Başar, and N. Hovakimyan. „On the construction of barrier in a visibility based pursuit evasion game“. In: *European Control Conference, ECC 2014, Strasbourg, France, June 24-27, 2014*. 2014, pp. 1894–1901. DOI: 10.1109/ECC.2014.6862391.

[ZL11] M. Zhang and H. Liu. „Persistent tracking using unmanned aerial vehicle: A game theory method“. In: *AIAA Guidance, Navigation, and Control Conference*. 2011.

[SS95] J. Shinar and G. Silberman. „A Discrete Dynamic Game Modelling Anti-missile Defense Scenarios“. In: *Dyn. Control* 5.1 (Jan. 1995), pp. 55–67. ISSN: 0925-4668. DOI: 10.1007/BF01968535.

[Pon11] M. Pontani. „Numerical Solution of Orbital Combat Games Involving Missiles and Spacecraft“. In: *Dynamic Games and Applications* 1.4 (2011), pp. 534–557.

[TS03] V. Turetsky and J. Shinar. „Missile guidance laws based on pursuit–evasion game formulations“. In: *Automatica* 39.4 (2003), pp. 607 –618. ISSN: 0005-1098. DOI: 10.1016/S0005-1098(02)00273-X.

[She02] S. Shedied. *Optimal Control for a Two Player Dynamic Pursuit Evasion Game: The Herding Problem*. University Libraries, Virginia Polytechnic Institute and State University, 2002.

[GTG06] B. P. Gerkey, S. Thrun, and G. Gordon. „Visibility-based Pursuit-evasion with Limited Field of View“. In: *Int. J. Rob. Res.* 25.4 (Apr. 2006), pp. 299–315. ISSN: 0278-3649. DOI: 10.1177/0278364906065023.

[SO14a] N. M. Stiffler and J. M. O'Kane. „A complete algorithm for visibility-based pursuit-evasion with multiple pursuers“. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 1660–1667. DOI: 10.1109/ICRA.2014.6907074.

[SO14b] N. M. Stiffler and J. M. O'Kane. „A sampling-based algorithm for multi-robot visibility-based pursuit-evasion“. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 1782–1789. DOI: 10.1109/IROS.2014.6942796.

[Bor21] Émile Borel. „La théorie du jeu et les équations intégrales à noyau symétrique“. In: *Comptes Rendus Hebdomadaires des Séances de l'Academie des Sciences* 173 (1921), 1304–8.

[Neu28] J. v. Neumann. „Zur Theorie der Gesellschaftsspiele“. In: *Mathematische Annalen* 100 (1928), pp. 295–320.

[NM07] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior (60th-Anniversary Edition)*. Princeton University Press, 2007, pp. I–XXXII, 1–739. ISBN: 978-0-691-13061-3.

[Nas50a] J. F. Nash. „Non-cooperative Games“. PhD thesis. Princeton, NJ: Princeton University, May 1950.

[Nas50b] J. Nash. „The Bargaining Problem“. In: *Econometrica* 18.2 (1950), pp. 155–162.

[Nas53] J. Nash. „Two-Person Cooperative Games“. In: *Econometrica* 21.1 (1953), pp. 128–140.

[Sha52] L. S. Shapley. „Notes on the n-Person Game, III: Some Variants of the Von Neumann-Morgenstern Definition of Solution“. In: *Research Meomoranda, Santa Monica, CA: RAND Corporation* RM-817 (1952).

[Sha53] L. S. Shapley. „A value for n-person games. Contribution to the Theory of Games“. In: *Annals of Mathematics Studies* 2 (1953). Ed. by H. Kuhn and A. Tucker, p. 28.

[Nah12] P. J. Nahin. *Chases and Escapes: The Mathematics of Pursuit and Evasion (Princeton Puzzlers)*. Princeton University Press, 2012. ISBN: 0691155011.

[Isa51] R. Isaacs. „Games of Pursuit“. In: *Santa Monica, CA: RAND Corporation* (1951).

[Isa65] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. New York: John Wiley and Sons, Inc., 1965. ISBN: 0486406822.

[Bel57] R. Bellman. *Dynamic Programming*. 1st ed. Princeton, NJ, USA: Princeton University Press, 1957.

[Mer72] A. W. Merz. „The game of two identical cars“. In: *Journal of Optimization Theory and Applications* 9.5 (1972), pp. 324–343. ISSN: 1573-2878. DOI: 10.1007/BF00932932.

[PT00] V. S. Patsko and V. L. Turova. *Numerical Study of Differential Games with the Homicidal Chauffeur Dynamics*. Tech. rep. IMM Ural Branch of RAS, Ekaterinburg, Russia, 2000.

[Mit01] I. Mitchell. *Games of Two Identical Vehicles*. Tech. rep. SU-DAAR740. Stanford University, Department of Aeronautics and Astronautics, 2001.

[PT11] V. S. Patsko and V. L. Turova. „Homicidal Chauffeur Game: History and Modern Studies“. In: *Advances in Dynamic Games: Theory, Applications, and Numerical Methods for Differential and Stochastic Games*. Ed. by M. Breton and K. Szajowski. Boston: Birkhäuser Boston, 2011, pp. 227–251. ISBN: 978-0-8176-8089-3. DOI: 10.1007/978-0-8176-8089-3_12.

[BO99] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory (Classics in Applied Mathematics)*. 2nd ed. Soc for Industrial & Applied Math, Jan. 1999. ISBN: 089871429X.

[KR05] S. Kopparty and C. V. Ravishankar. „A framework for pursuit evasion games in Rn“. In: *Information Processing Letters* 96.3 (2005), pp. 114 –122. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2005.04.012.

[BBH07] S. D. Bopardikar, F. Bullo, and J. Hespanha. „A cooperative homicidal chauffeur game“. In: *Decision and Control, 2007 46th IEEE Conference on*. 2007, pp. 4857–4862. DOI: 10.1109/CDC.2007.4434251.

[Ls10] R. Liu and C. ze su. „A novel approach based on evolutionary game theoretic model for multi- player pursuit evasion“. In: *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*. Vol. 1. 2010, pp. 107–110. DOI: 10.1109/CMCE.2010.5609628.

[LQT12] Y. LIU, N. QI, and Z. TANG. „Linear Quadratic Differential Game Strategies with Two-pursuit Versus Single-evader“. In: *Chinese Journal of Aeronautics* 25.6 (2012), pp. 896 –905. ISSN: 1000-9361. DOI: 10.1016/S1000-9361(11)60460-3.

[Liu+13] S. Y. Liu et al. „Evasion as a team against a faster pursuer“. In: *2013 American Control Conference*. 2013, pp. 5368–5373. DOI: 10.1109/ACC.2013.6580676.

[WF13] T. K. Wang and L. C. Fu. „A guidance strategy for multi-player pursuit and evasion game in maneuvering target interception“. In: *Control Conference (ASCC), 2013 9th Asian*. 2013, pp. 1–6. DOI: 10.1109/ASCC.2013.6606182.

[CSG09] Z.-s. Cai, L.-n. Sun, and H.-b. Gao. „A Novel Hierarchical Decomposition for Multi-player Pursuit Evasion Differential Game with Superior Evaders“. In: *Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation*. GEC '09. Shanghai, China: ACM, 2009, pp. 795–798. ISBN: 978-1-60558-326-6. DOI: 10.1145/1543834.1543945.

[RT07] J. Reimann and G. I. of Technology. *Using Multiplayer Differential Game Theory to Derive Efficient Pursuit-evasion Strategies for Unmanned Aerial Vehicles*. Georgia Institute of Technology, 2007. ISBN: 9780549112112.

[Pan+12] S. Pan et al. „Pursuit, evasion and defense in the plane“. In: *2012 American Control Conference (ACC)*. 2012, pp. 4167–4173. DOI: 10.1109/ACC.2012.6315389.

[LU06] D. Li and T. O. S. University. *Multi-player Pursuit-evasion Differential Games*. Ohio State University, 2006. ISBN: 9780542930645.

[LCS08] D. Li, J. B. Cruz, and C. J. Schumacher. „Stochastic multi-player pursuit–evasion differential games“. In: *International Journal of Robust and Nonlinear Control* 18.2 (2008), pp. 218–247. ISSN: 1099-1239. DOI: 10.1002/rnc.1193.

[Wei+07] M. Wei et al. „Multi-Pursuer Multi-Evader Pursuit-Evasion Games with Jamming Confrontation“. In: *JACIC* 4.3 (2007), pp. 693–706. DOI: 10.2514/1.25329.

[Ge+06] J. Ge et al. „Hierarchical decomposition approach for pursuit-evasion differential game with multiple players“. In: *2006 IEEE Aerospace Conference*. 2006, 7 pp.–. DOI: 10.1109/AERO.2006.1656027.

[FV13] A. Festa and R. B. Vinter. „A decomposition technique for pursuit evasion games with many pursuers“. In: *52nd IEEE Conference on Decision and Control*. 2013, pp. 5797–5802. DOI: 10.1109/CDC.2013.6760803.

[SV03] J. A. Sethian and A. Vladimirsky. „Ordered Upwind Methods for Static Hamilton–Jacobi Equations: Theory and Algorithms“. In: *SIAM Journal on Numerical Analysis* 41.1 (Jan. 2003), pp. 325–363. ISSN: 0036-1429. DOI: 10.1137/s0036142901392742.

[AS16] M. D. Awheda and H. M. Schwartz. „Decentralized learning in pursuit-evasion differential games with multi-pursuer and single-superior evader“. In: *2016 Annual IEEE Systems Conference (SysCon)*. 2016, pp. 1–8. DOI: 10.1109/SYSCON.2016.7490516.

[JQ10] S. Jin and Z. Qu. „Pursuit-evasion games with multi-pursuer vs. one fast evader“. In: *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*. 2010, pp. 3184–3189. DOI: 10.1109/WCICA.2010.5553770.

[RK15] M. V. Ramana and M. Kothari. „A cooperative pursuit-evasion game of a high speed evader“. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. 2015, pp. 2969–2974. DOI: 10.1109/CDC.2015.7402668.

[GRH10] C. Giovannangeli, E. Rivlin, and M. Heymann. *Pursiut-Evasion Games in Presence of Obstacles in Unknown Environments: Towards an Optimal Pursuit Strategy*. INTECH Open Access Publisher, 2010. ISBN: 9789533070629.

[BHB09] S. Bhattacharya, S. Hutchinson, and T. Başar. „Game-theoretic analysis of a visibility based pursuit-evasion game in the presence of obstacles“. In: *2009 American Control Conference*. 2009, pp. 373–378. DOI: 10.1109/ACC.2009.5160610.

[BBH14a] S. Bhattacharya, T. Başar, and N. Hovakimyan. „On the construction of barrier in a visibility based pursuit evasion game“. In: *Control Conference (ECC), 2014 European*. 2014, pp. 1894–1901. DOI: 10.1109/ECC.2014.6862391.

[BBF14] S. Bhattacharya, T. Başar, and M. Falcone. „Numerical approximation for a visibility based pursuit-evasion game“. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 68–75. DOI: 10.1109/IROS.2014.6942542.

[ZB16] R. Zou and S. Bhattacharya. „Visibility-Based Finite-Horizon Target Tracking Game“. In: *IEEE Robotics and Automation Letters* 1.1 (2016), pp. 399–406. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2521429.

[AX09] J. Annas and J. Xiao. „Intelligent pursuit & evasion in an unknown environment“. In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. 2009, pp. 4899–4906. DOI: 10.1109/IROS.2009.5354246.

[DZJ12] J. Dong, X. Zhang, and X. Jia. „Strategies of Pursuit-Evasion Game Based on Improved Potential Field and Differential Game Theory for Mobile Robots“. In: *Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2012 Second International Conference on*. 2012, pp. 1452–1456. DOI: 10.1109/IMCCC.2012.340.

[LJTH64] C. E. Lemke and J. J. T. Howson. „Equilibrium Points of Bimatrix Games“. In: *Journal of the Society for Industrial and Applied Mathematics* 12.2 (1964), pp. 413–423. DOI: 10.1137/0112033.

[Joh13] S. G. Johnson. *The NLopt nonlinear-optimization package*. http://ab-initio.mit.edu/nlopt. 2013.

[Cha09] B. Chatterjee. „An optimization formulation to compute Nash equilibrium in finite games“. In: *Methods and Models in Computer Science, 2009. ICM2CS 2009. Proceeding of International Conference on*. 2009, pp. 1–5.

[AB16] A. Alexopoulos and E. Badreddin. „Decomposition of multi-player games on the example of pursuit-evasion games with unmanned aerial vehicles“. In: *2016 American Control Conference (ACC)*. 2016, pp. 3789–3795. DOI: 10.1109/ACC.2016.7525503.

[ASB14b] A. Alexopoulos, T. Schmidt, and E. Badreddin. „A pursuit-evasion game between unmanned aerial vehicles“. In: *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*. Vol. 02. 2014, pp. 74–81.

[ASB16] A. Alexopoulos, T. Schmidt, and E. Badreddin. „Real-Time Implementation of Pursuit-Evasion Games Between Unmanned Aerial Vehicles“. In: *Informatics in Control, Automation and Robotics: 11th International Conference, ICINCO 2014 Vienna, Austria, September 2-4, 2014 Revised Selected Papers*. Ed. by J. Filipe et al. Cham: Springer International Publishing, 2016, pp. 147–163. ISBN: 978-3-319-26453-0. DOI: 10.1007/978-3-319-26453-0_9.

[AB12] A. Alexopoulos and E. Badreddin. „Multi-Agent Pursuit-Evasion Game with Unmanned Aerial Vehicles (UAVs)“. In: *Annual Report of the Institute for Computer Engineering* (2012).

[ASB14a] A. Alexopoulos, T. Schmidt, and E. Badreddin. „Pursuit and evasion in a recursive nested behavioral control structure for unmanned aerial vehicles“. In: *Control, Automation and Systems (ICCAS), 2014 14th International Conference on*. 2014, pp. 1175–1180. DOI: 10.1109/ICCAS.2014.6987737.

[Sch14] T. Schmidt. „Drei-Agenten Pursuit-Evasion-Spiel mit unbemannten Luftfahrzeugen“. Master Thesis. Heidelberg University, 2014.

[ASB15] A. Alexopoulos, T. Schmidt, and E. Badreddin. „Cooperative pursue in pursuit-evasion games with unmanned aerial vehicles“. In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. 2015, pp. 4538–4543. DOI: 10.1109/IROS.2015.7354022.

[McL08] M. McLure. „Vilfredo Pareto, 1906 Manuale di Economia Politica, Edizione Critica, Aldo Montesano, Alberto Zanni and Luigino Bruni (eds), (Milan: EGEA—Università Bocconi Editore, 2006) pp. XXII, 706, ISBN 88-8350-084-9“. In: *Journal of the History of Economic Thought* 30.01 (2008), pp. 137–140.

[Bad06] E. Badreddin. „Recursive behavior-based architecture for mobile robots.“ In: *Robotics and Autonomous Systems* 8.3 (Apr. 20, 2006), pp. 165–176.

[Bar+09] C. Bartolein et al. „Dependable Design for Assistance Systems: Electrical Powered Wheelchairs“. In: *DDCS*. Hamburg, Germany, 2009.

[Kan+10] A. A. Kandil et al. „Collision avoidance in a recursive nested behaviour control structure for Unmanned Aerial Vehicles.“ In: *SMC*. IEEE, 2010, pp. 4276–4281.

[Set98] J. Sethian. „Adaptive fast marching and level set methods for propagating interfaces.“ eng. In: *Acta Mathematica Universitatis Comenianae. New Series* 67.1 (1998), pp. 3–15.

[Pey09] G. Peyre. *Toolbox Fast Marching*. http://www.mathworks.com/matlabcentral/fileexchange/6110-toolbox-fast-marching. Accessed: 2016-02-15. 2009.

[Kir15] B. F. Kirsch. „Non-linear model based control of a hex rotor“. Master Thesis. Heidelberg University, Dec. 2015.

[Ras14] Raspberry Pi Foundation. *Raspberry Pi*. www.raspberrypi.org. 2014.

[Cub14] CubieTech Ltd. *Cubieboard – A series of open ARM miniPCs*. www.cubieboard.org. 2014.

[Tex14a] Texas Instruments Inc. *BeagleBoard.org*. www.beagleboard.org. 2014.

[Tex14b] Texas Instruments Inc. *BeagleBone Black*. www.beagleboard.org/Products/BeagleBoneBlack. 2014.

[Bbb] *BeagleBone Black*. http://beagleboard.org/static/uploads/BBB_Alt_View_Small.png. Accessed: 2016-12-30.

[Vn2] *Vectornav VN-200 Rugged*. http://www.vectornav.com/images/default-source/products/vn-200-rugged-(1).gif?sfvrsn=6. Accessed: 2016-12-30.

[K50] *ComNav K501 GNSS Board*. http://www.gpsworld.com/wp-content/uploads/2013/05/201342815202868161.jpg. Accessed: 2016-12-30.

[T30] *ComNav T-300 Base Station*. https://geo-matching.com/upload/2148-general.png. Accessed: 2016-12-30.

[Xbeb] *XBee-Pro*. https://www.digi.com/products/xbee-rf-solutions/modules/xbee-digimesh-2-4. Accessed: 2016-10-14.

[Xbea] *XBee-Pro S1 RF module*. https://www.electronic-shop.lu/media/catalog/product/9/6/964-02.jpg. Accessed: 2016-10-14.

[Mx2] *Graupner MX-20*. https://img.conrad.de/medias/global/ce/2000_2999/2000/2090/2099/209941_BB_00_FB.EPS_1000.jpg. Accessed: 2016-12-30.

[Voo09] H. Voos. „Entwurf eines Flugreglers für ein vierrotoriges unbemanntes Fluggerät (Control Systems Design for a Quadrotor UAV)“. In: *Automatisierungstechnik* 57.9 (2009), pp. 423–431.

[Bea06] M. Beatty. *Principles of Engineering Mechanics: Volume 2 Dynamics – The Analysis of Motion*. Mathematical Concepts and Methods in Science and Engineering. Springer, 2006. ISBN: 9780387237046.

[KKK95] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos. *Nonlinear and Adaptive Control Design*. 1st. New York, NY, USA: John Wiley & Sons, Inc., 1995. ISBN: 0471127329.

[BS05] S. Bouabdallah and R. Siegwart. „Backstepping and sliding-mode techniques applied to an indoor micro quadrotor“. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005. ICRA 2005.* IEEE. 2005, pp. 2247–2252.

[Ale+13] A. Alexopoulos et al. „A Comparative Study of Collision Avoidance Techniques for Unmanned Aerial Vehicles“. In: *SMC*. 2013, pp. 1969–1974.

[Cha10] B. Chatterjee. *N-person game*. www . mathworks . com / matlabcentral/fileexchange/27837-n-person-game. 2010.

[Ste13] Steven G. Johnson. *The NLopt nonlinear-optimizazion package*. http://ab-initio.mit.edu/nlopt. 2013.

[Kra88] D. Kraft. *A software package for sequential quadratic programming*. Tech. rep. DFVLR-FB 88-28. DFVLR, Cologne, Germany, 1988.

[Kra94] D. Kraft. „Algorithm 733: TOMP–Fortran modules for optimal control calculations“. In: *ACM Transactions on Mathematical Software* 20.3 (1994), pp. 262–281.

[Wm2] *Kaindl Electronic Windmaster 2*. https : / / img . conrad . de / medias/global/ce/8000_8999/8100/8130/8135/100464_BB_00_FB.EPS.jpg. Accessed: 2016-12-30.

A Experimental Results (cont.)

All remaining figures of the player's trajectories for all experiments described in chapter 8 can be found below. Note, that the initial position of the actual UAV is always assumed to be the origin. Thus, the initial positions of the other players change respectively to maintain the relative positioning to each other.

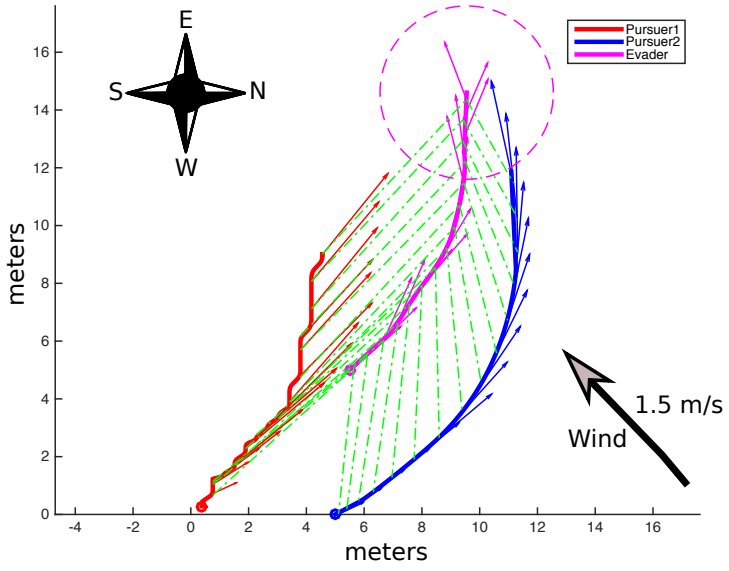

Figure A.1: *Experiment 1: actual UAV pursuer* 1.

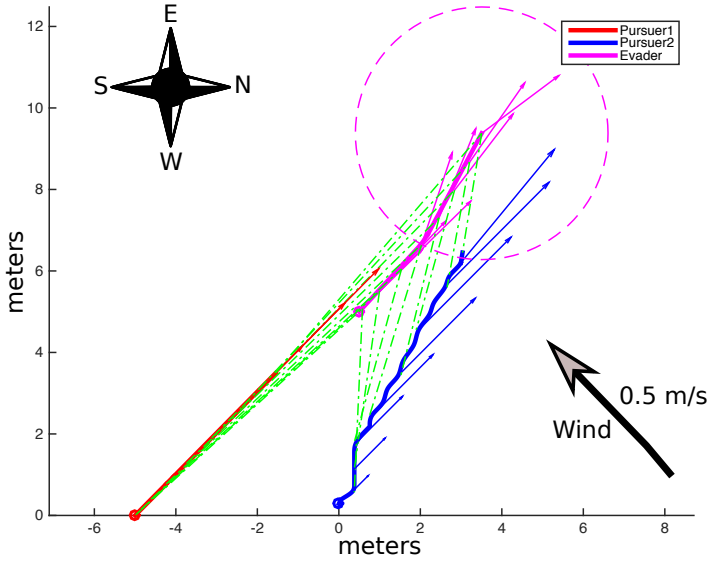

Figure A.2: *Experiment 1: actual UAV pursuer* 2.

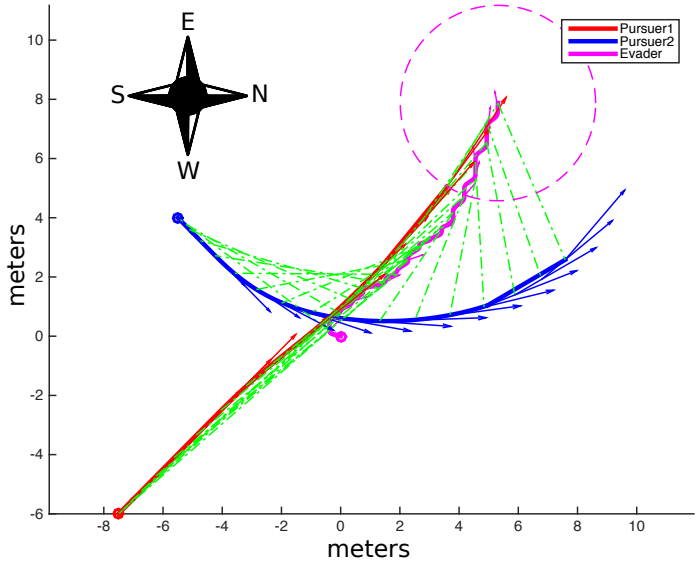

Figure A.3: *Experiment 2: actual UAV evader.*

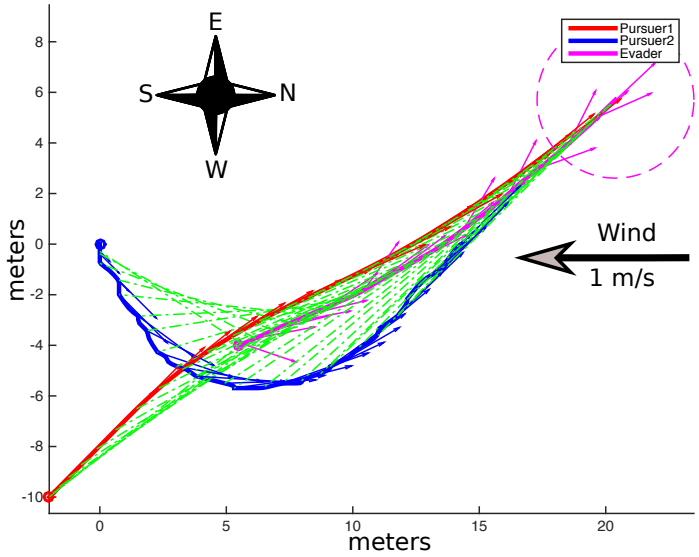

Figure A.4: *Experiment 2: actual UAV pursuer* 2.

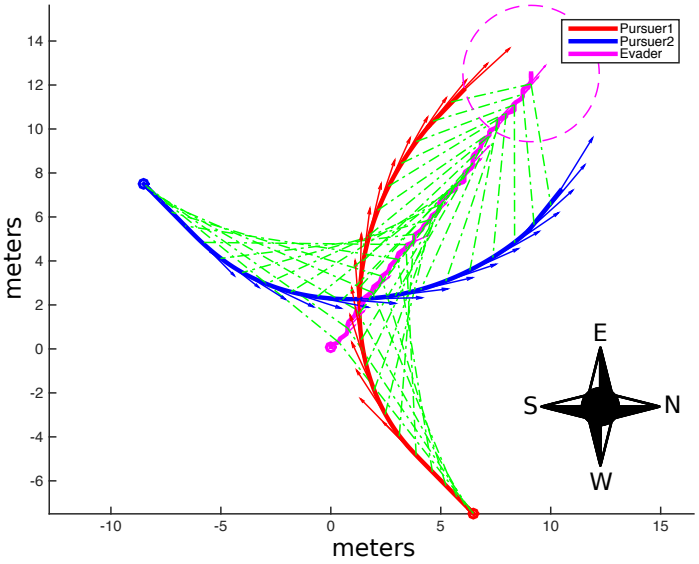

Figure A.5: *Experiment 3: actual UAV evader.*

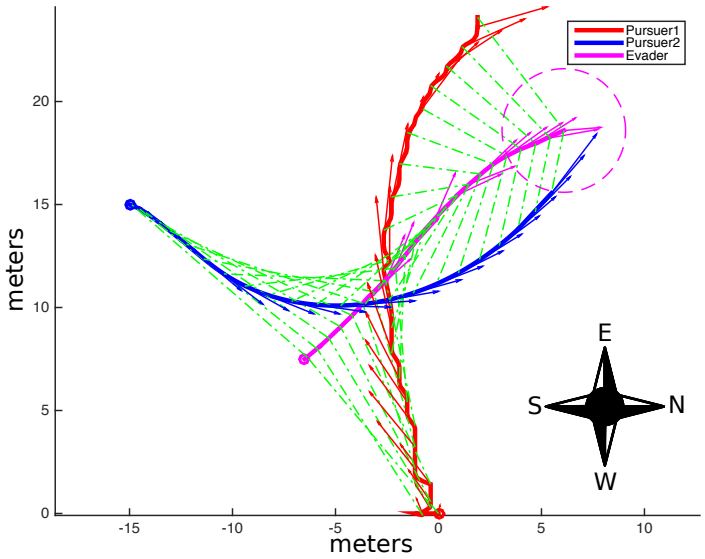

Figure A.6: *Experiment 3: actual UAV pursuer* 1.

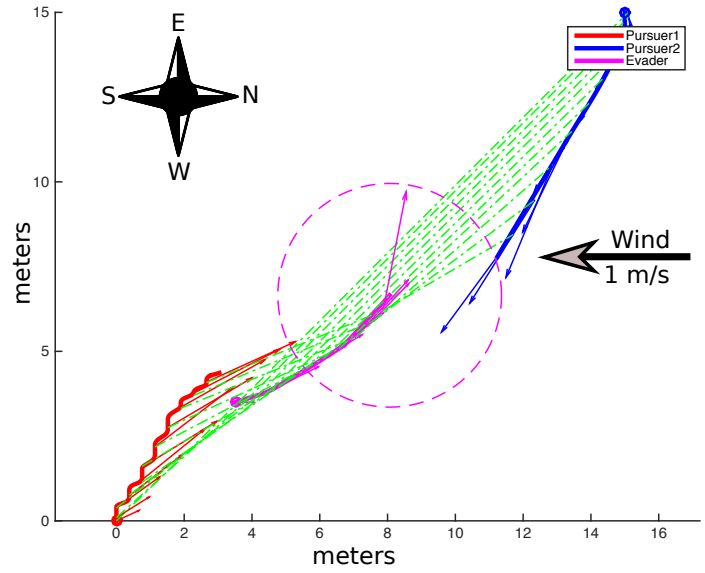

Figure A.7: *Experiment 4: actual UAV pursuer* 1.

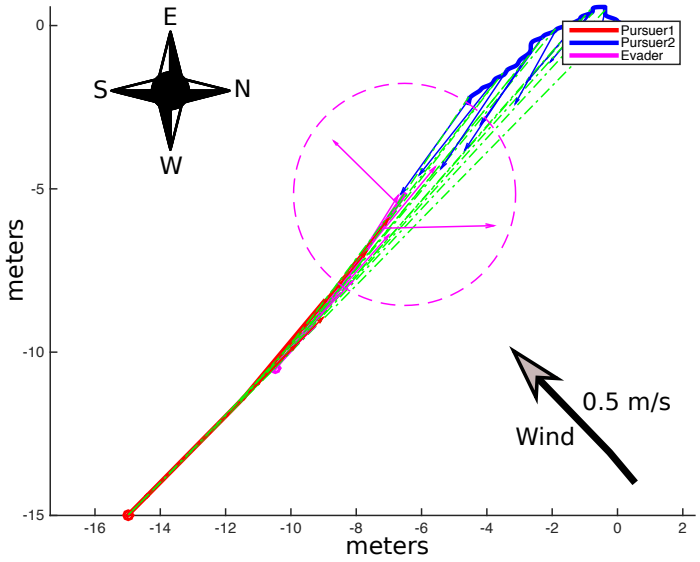

Figure A.8: *Experiment 4: actual UAV pursuer* 2.

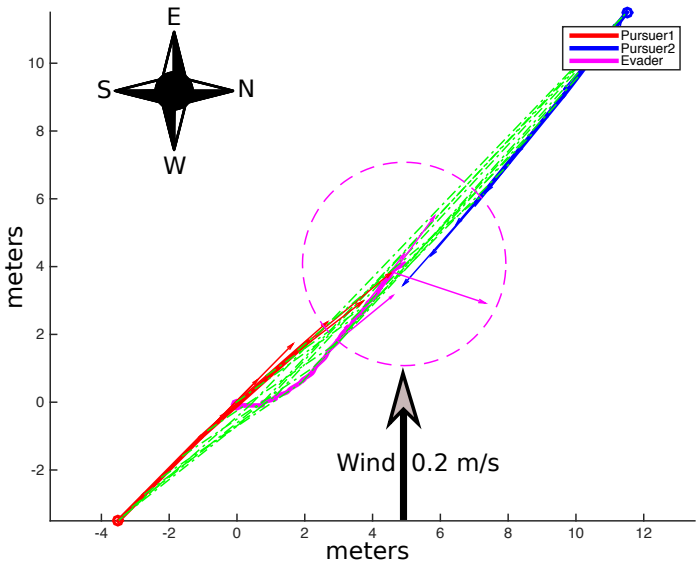

Figure A.9: *Experiment 5: actual UAV evader.*

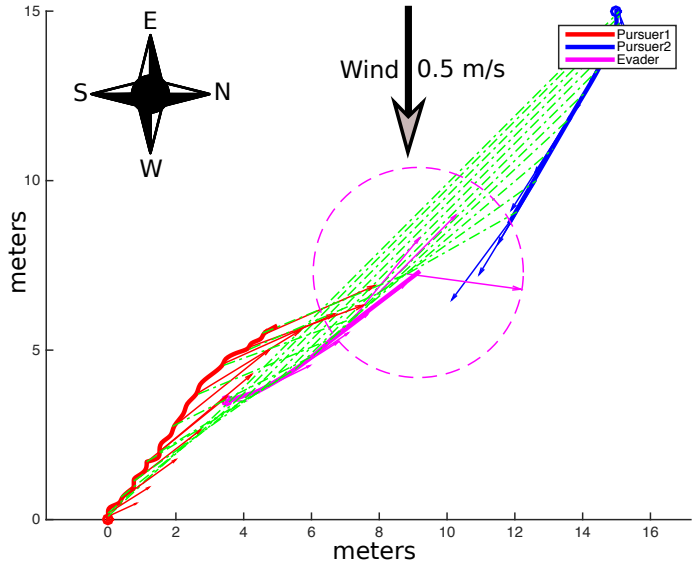

Figure A.10: *Experiment 5: actual UAV pursuer* 1.