Philipp Thoms

Anwendung von dreidimensionalem Context-Steering auf Quadcopter-Schwärme

INF FAKULTÄT FÜR INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Anwendung von dreidimensionalem Context-Steering auf Quadcopter-Schwärme

Master Thesis

Philipp Thoms

26. September 2023

Betreuender Professor: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Betreuer: Dr.-Ing. Christoph Steup

Philipp Thoms: *Anwendung von dreidimensionalem Context-Steering auf Quadcopter-Schwärme*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2023.

Abstrakt

Die Verwendung von Quadcoptern steigt und durch ihre Vielseitigkeit kommen sie in verschiedenen Anwendungsgebieten zum Einsatz. Diese sind zum Beispiel Überwachung oder Suchmissionen. In der Forschung gibt es dabei verschiedene Schwerpunkte. Einer davon ist die Wegfindung. Da dynamische Echtzeit-Szenarien langfristige Pfadplanung sehr erschweren und Quadcopter ein sehr kompaktes ressourcenarmes System sind, wird die Verwendung vieler klassischer Ansätze stark eingeschränkt.

Context-Steering ist ein lokaler Algorithmus, der reaktiv auf die Umgebung seinen Weg berechnet. Diese Arbeit zielt darauf ab, den Vorteil der Verwendung von Context-Steering gegenüber eines Attraction-Repulsion-Systems aufzuzeigen unter den Bedingungen, dass beide Ansätze sehr ähnlich implementiert werden und die gleichen Szenarien absolvieren. Um dieses Ziel zu erreichen, werden zwei Experiment-Durchläufe durchgeführt. Die dabei genutzte Karte ist ein Tunnel mit zwei gleichgroßen Räumen an beiden Enden. Die Größe ist dabei frei skallierbar. In verschiedenen Szenarien müssen die Kopter Interest-Punkte abfliegen. Nach dem ersten Durchlauf werden Optimierungen vorgenommen und auftretende Probleme gelöst. Große dabei auftretende Probleme waren die Stabilität, Robustheit und Parameterfindung, die teilweise behoben werden konnten.

Die Ergebnisse aus diesen Experimenten zeigen dabei, welche Vorteile die Verwendung von Context-Steering besitzt. Die Parametrisierung ist aufgrund der sehr entkoppelten Funktionsweise in Context-Steering viel einfacher als bei seinem Vergleichspartner. Das Attraction-Repulsion-System funktioniert basierend auf einem sehr ineinander verzahnten Kräftesystem, bei dem kleine Änderungen große und unvorhersehbare Auswirkungen haben können. Durch diese Struktur gestaltete sich eine Optimierung und Anpassung der Parameter sehr schwierig, weshalb sich die Ergebnisse im Gegensatz zu Context-Steering im zweiten Durchlauf verschlechtert haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ENU East-North-Up

GCS Ground Control Station

IP Interest-Punkt

KI Künstliche Intelligenz

MOOP Multi-Kriterielles-Optimierungs-Problem

MOP Multi-Kriterielles-Problem

SOP Einzelaufgaben-Problem

1. Einleitung

Einer der Schwerpunkte in der Schwarmrobotik ist die Herausforderung der Nutzung des Schwarmverhaltens zur Wegfindung. Hierbei handelt es sich um ein multidisziplinäres Forschungsfeld, das sich mit dem Kooperieren und Kommunizieren von autonomen Robotern beschäftigt, um gemeinsam komplexe Aufgaben, wie Überwachungs- oder Rettungsmissionen zu bewältigen.

Die Wegfindung des Roboteschwarms in einer dynamischen und unvorhersehbaren Umgebung ist für den erfolgreichen Abschluss der Aufgaben von größter Relevanz. Die Herausforderung dabei liegt nicht nur darin, dass sie Hindernissen ausweichen und den richtigen Weg finden, sondern auch gleichzeitig die Schwarm-Dynamik aufrecht erhalten. Diese Komplexität kann traditionelle Ansätze der Wegfindung übersteigen und vor allem der dynamische Anteil gestaltet die Verwendung von langfristiger Pfadplanung als sehr schwierig [10].

Die Wahl der korrekten Route hat einen direkten Einfluss auf die Sicherheit und Effizienz des gesamten Schwarms. Um diese korrekt zu berechnen, müssen viele Faktoren wie Echtzeitinformationen über Umgebung, Position des Schwarms und die Kommunikation untereinander zusätzlich zu den Informationen über die Ziele in Betracht gezogen werden. Je nach verwendetem Algorithmus ist die Parametrisierung eine weitere großes Herausforderung, die vor der Verwendung bewältigt werden muss. So wird immer wieder nach neuen Algorithmen gesucht, die eine einfachere und effizientere Anwendung bieten.

1.1. Motivation

Wie bereits erwähnt, ist die Wegfindung in der Schwarmrobotik eines der zentralen Themen. Die Pfadplanung ist aufgrund von großer Speicheranforderung und der herausfordernden Echtzeitplanung in dynamischen Systemen sehr schwierig. Diese Systeme umfassen beispielsweise Rettungs -und Suchmissionen oder andere Logistikanwendungen. Ein weiteres großes Anwendungsgebiet ist autonomes Fahren bei einfachen Robotern bis hin zu Kraftfahrzeugen. Wenn in diesen Gebieten das anwendende System stark ressourcenbegrenzt ist, zum Beispiel in Form eines Quadcopters, ist die Anwendbarkeit der rechenintensiveren Algorithmen, wie zum Beispiel A*, sehr stark begrenzt, da diese auf einem so sehr optimierten System nicht funktionieren würden.

Deshalb beschäftigt sich diese Arbeit mit einer alternativen Lösung der Wegfindung, dem Context-Steering, welche bereits in Videospielen erfolgreich genutzt wird [9][10][11]. Hier ist die Verwendung hauptsächlich durch zweidimensionale Anwendungen geprägt [11]. Als direktes Beispiel führt Andrew Fray [11] die Rennfahrer-Intelligenz in dem Formel-1-Rennspiel "F1 2010" an.

1.2. Ziele

Das Ziel dieser Arbeit ist es zu testen, ob Context-Steering eine sinnvolle und einfachere Alternative zu den herkömmlichen Attraction-Repulsion-Ansätzen in der Schwarmrobotik ist. Die Anwendung bezieht sich dabei auf Quadcopter-Schwärme, die in der "Paparazzi UAV"-Simulation emuliert werden sollen. Aus der Motivation, eine alternative Möglichkeit der Wegfindung zu finden, ergibt sich folgende Grundfrage:
Wie schneidet Context-Steering im Vergleich zu Attraction-Repulsion in der Schwarmrobotik ab?
Um die Frage zu klären, müssen folgende Teilfragen vorab bearbeitet werden:

- Kann die aktuelle Anwendung von Context-Steering von zweidimensionalen auf dreidimensionale Szenarien übertragen werden?
- Existiert eine Parametrisierung für beide Ansätze, die in mehreren verschiedenen Szenarien funktioniert und wenn ja, wie schwer ist es, eine geeignete zu finden?
- Ist es möglich, stabiles Schwarmverhalten in definierten dreidimensionalen Szenarien zu generieren?

Um diese Fragen zu klären, wird eine Reihe an Experimenten durchgeführt, um in typischen Schwarmszenarien quantitativ Context-Steering und Attraction/Repulsion zu vergleichen. Dabei werden die zu vergleichenden Algorithmen auf unterschiedliche Weise vor verschiedene Herausforderungen gestellt.

Wenn sich Context-Steering als geeignete Alternative herausstellt, kann zukünftig ein wesentlich einfacherer zu implementierender und zu parametrisierender Ansatz für die Wegfindung gewählt werden.

1.3. Struktur

Zunächst wird im Grundlagenkapitel 2 die Wegfindung in der Schwarmrobotik 2.1 erläutert. Daraufhin wird der aktuelle Stand 2.2 der Verwendung von Quadcoptern und verschiedenen Algorithmen, die in Künstlicher-Intelligenz-Bewegungs-Szenarien genutzen werden, erklärt. Daraus resultiert Context-Steering, welches in Hinblick auf seine Verwendung in Kapitel 2.3 erläutert wird. Die für diese Arbeit benötigten Programme und Hardwarekomponenten werden im Anschluss in Kapitel 2.5 und 2.6 dargestellt.

Die Anwendung der Grundlagen geschieht in Kapitel 3. Hier wird eine Übersicht über die Verwendungs- und Einstellungsmöglichkeiten von Context-Steering mit Quadcoptern in Kapitel 3.1 geben und die dazugehörenden Entscheidungsmöglichkeiten in Kapitel 3.4 beschrieben.

Wie genau die beiden Algorithmen, ihre Parameter und der Aufbau der Experimente aussieht, wird zu Beginn von Kapitel 4 festgelegt. Daraufhin werden die Experimente durchgeführt und ausgewertet. Die daraus resultierenden Ergebnisse werden noch einmal in Kapitel 5 zusammengefasst und deren Verwendung für zukünftige Projekte dargestellt.

2. Grundlagen

Das Grundlagenkapitel beginnt mit einer Einführung in die Wegfindung in der Schwarmrobotik 2.1. Hier wird zunächst die Definition eines Schwarms gegeben und die Funktionsweise der Bewegung einzelner Individuen dargestellt. Dazu zählen die Berechnungsansätze, die zur Lösung dieser Herausforderung verwendet werden können und Messverfahren zum Erfassen der Vergleichswerte für die Stabilität des Schwarms. Im Anschluss wird im Stand der Wissenschaft 2.2 die aktuelle Verwendung von Quadcoptern in Schwarmanwendungen erläutert. Dazu gehört, welche Wegfindungsalgorithmen benutzt werden sowie deren Funktionsweise. Daraus resultiert die Verwendung von Context-Steering 2.3. In dem dazugehörigen Kapitel wird die Funktionweise des Algorithmus beschrieben inklusive verschiedener Erweiterungen. Eine davon ist, Context-Steering als Multi-Kriterielles-Problem (MOP) aufzufassen und mithilfe der in Kapitel 2.4 besprochenen Ansätze zu lösen. Im Anschluss daran wird in den letzten beiden Kapiteln 2.5 und 2.6 auf die Simulation und die Quadcopter eingegangen, die für diese Arbeit verwendet werden.

2.1. Wegfindung in der Schwarmrobotik

Bevor auf die Funktionweise der Wegfindung in der Schwarmrobotik eingegangen wird, muss zuerst geklärt werden, was ein Schwarm ist. Ein Schwarm ist: "A collective behavior of simple entities having simple rules with ability of local interactions. The swarm produces a global and intelligent behavior which is unknown to the single entities: “The whole is more than the sum of its parts!” [20]. Als Inspiration werden komplexe Verhalten aus der Natur genommen, zum Beispiel Ameisen, Bienen oder Termiten. Der Begriff Schwarm wurde zuerst von Beni [3] im Kontext zellularer Robotersysteme verwendet. Hier wurde der Begriff in Bezug auf die Intelligenz und Unvorhersehbarkeit des Verhaltens benutzt [22]. Dieses Verhalten wird durch simple Regeln ausgedrückt, die von Reynolds [18] erstmalig erwähnt wurden. Diese besagen, dass

das eigene Verhalten basierend auf den benachbarten Individuen beschrieben werden kann: Diese sind:

1. Flock Centering/Attraction (Anziehung) in 2.1

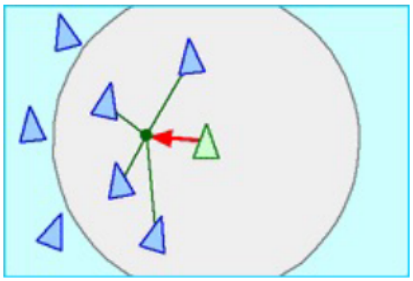


Abbildung 2.1.: Attraction im Schwarm [16]

2. Collision Avoidance/Repulsion (Abstoßung) in 2.2

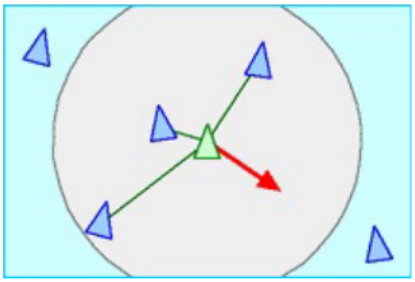


Abbildung 2.2.: Repulsion im Schwarm [16]

3. Velocity Matching/Allignment (Anordnung) in 2.3

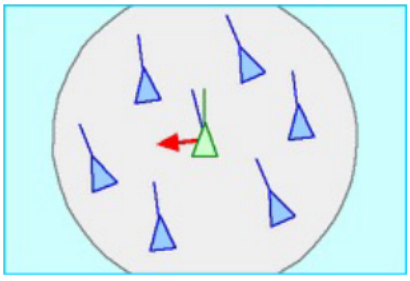


Abbildung 2.3.: Allignment im Schwarm [16]

Diese drei Regeln bilden zusammen ein Kräftesystem, mit dem jedes Manöver für jedes Individuum berechenbar ist und jedes Manöver von den benachbarten Individuen abhängt.

Attraction aus Abbildung 2.1 ist dabei die Kraft, die alle Individuen näher aneinander heranzieht. Repulsion aus Darstellung 2.2 ist das genaue Gegenteil, in dem die Kraft alle Individuen voneinander abstößt [16]. Die letzte Kraft ist Allignment aus Abbildung 2.3, die dafür sorgt, dass die Individuen in die gleiche Richtung mit der gleichen Geschwindigkeit ihrer Nachbarn steuern [16]. So lassen sich für die Kräfte Funktionen aufstellen, die diese beschreiben. Für Attraction und Repulsion sehen sie wie folgt aus:

Basisfunktion für Attraction und Repulsion Jede der nachfolgenden Ansätze berechnet die Anziehungskraft f_{Att} und Abstoßungskraft f_{Rep} auf ihre eigene Weise und nutzt dafür die Basisform 2.1

$$\vec{f_i}(i,j) = -(\vec{x}_i - \vec{x}_j)(f_{Att} - f_{Rep}) \tag{2.1}$$

Lineare Attraction aus Abbildung 2.4 lässt sich durch die Formel 2.2 beschreiben, wobei k_a die Stärke der Anziehung ist [16]. Wenn i und j weit voneinander entfernt sind, ist die Anziehung groß und je näher sie sich kommen desto geringer wird die Anziehung.

$$\vec{f_i}(i,j) = -k_a(\vec{x_i} - \vec{x_j}), k_a > 0 \tag{2.2}$$

Ein komfortabler Abstand wird durch die Formel 2.3 dargestellt [16]. Hier ist d die vom Benutzer gesetzte Distanz die das Individuum zu seinen Nachbarn erreichen und halten soll. Es wird je nach Abweichung stärker angezogen, wenn es weit entfernt ist oder stärker abgestoßen, wenn es nah an seinem Partner ist. Dies wird in Abbildung 2.5 veranschaulicht.

$$\vec{f_i}(i,j) = [-k(||\vec{x_i} - \vec{x_j}|| - d)](\vec{x_i} - \vec{x_j}), k > 0 \tag{2.3}$$

Abstoßen wenn nah dran beschreibt die Stärke der Repulsion-Kraft bei nahen Zielen mit der Formel 2.4. k_r ist dabei die Stärke der Repulsionskraft und r_s der Raum um das Individuum, in dem er alle anderen abstößt. Je

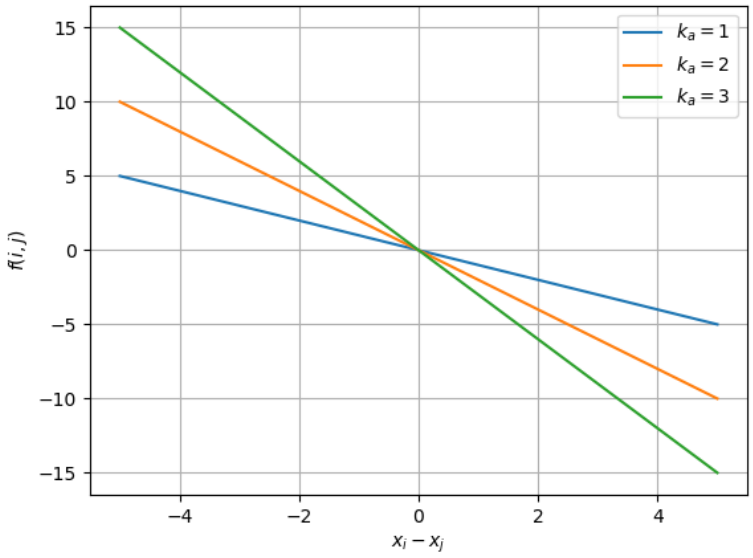


Abbildung 2.4.: Lineare Anziehung

näher es an einem Nachbarn ist, um so größer ist die Kraft, dargestellt in Abbildung 2.6.

$$\vec{f_i}(i,j) = k_r \exp\left(\frac{-\frac{1}{2}||\vec{x}_i - \vec{x}_j||^2}{r_s^2}\right)(\vec{x}_i - \vec{x}_j), k_r, r_s > 0 \tag{2.4}$$

Linieare begrenzte Attraction und unbegrenzte Repulsion ist die erste der drei Funktionen die Attraction und Repulsion miteinander vereint. Die Anziehung wird hier mit der Formel 2.5 festgelegt und wächst mit steigendem Abstand. Die Repulsion wird durch den Ausdruck 2.6 berechnet, wobei b der Abstoßungsfaktor ist. Dargestellt ist dieses Verhalten in Abbildung 2.7

$$f_{Att} = a \tag{2.5}$$

$$f_{Rep} = \frac{b}{||x_i - x_j||^2} \tag{2.6}$$

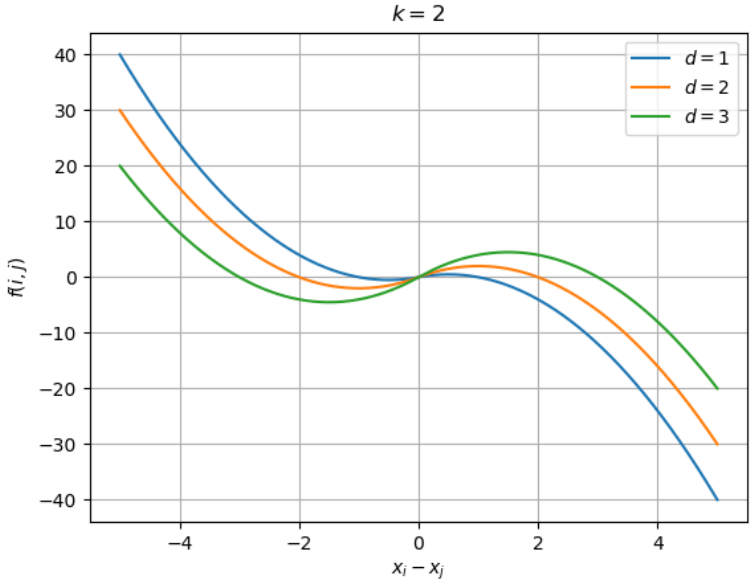


Abbildung 2.5.: Komfortable Distanz

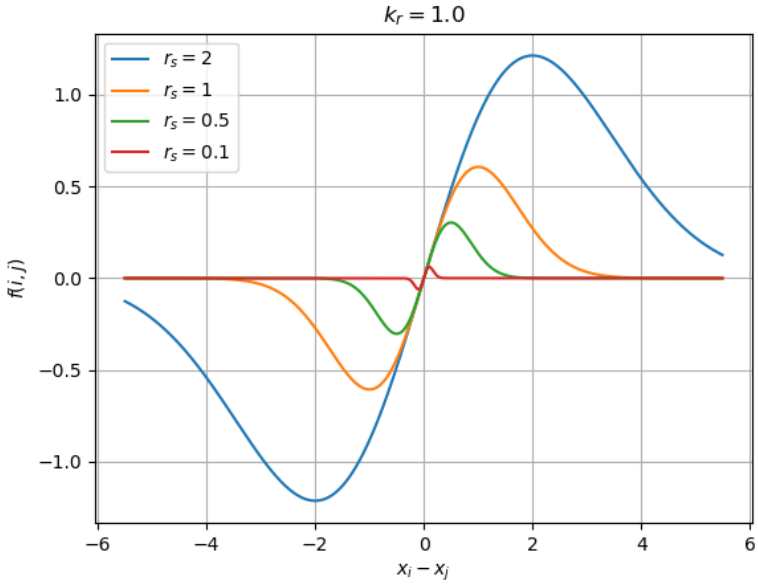


Abbildung 2.6.: Abstoßen wenn nah dran

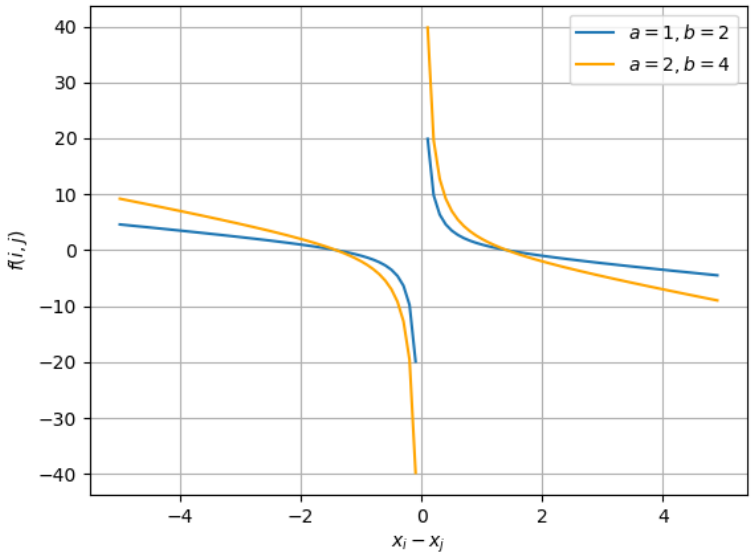


Abbildung 2.7.: Lineare begrenzte Anziehung und unbegrenzte Abstoßung

Fast konstante Attraction und unbegrenzte Repulsion ist die zweite der Funktionen und ändert nur die Berechnung der Anziehungskraft mit Formel 2.7. Die Abstoßungskraft wird wie beim Vorgänger mit dem Ausdruck 2.8 berechnet.

$$f_{Att} = \frac{a}{||x_i - x_j||} \tag{2.7}$$

$$f_{Rep} = \frac{b}{||x_i - x_j||^2} \tag{2.8}$$

Begrenzte Attraction und begrenzte Repulsion ist die letzte Form der Berechnung. In dieser Funktion 2.9 wird eine konstante Attraction a benutzt. Die begrenzte Repulsion wird wie in Formel 2.10 berechnet und ist im Grunde die gleiche Formel, die im "Abstoßen wenn nah"-Ansatz benutzt wird. Es sind die gleichen zwei frei wählbaren Parameter enthalten, wobei b die Stärke der Repulsion-Kraft verändert und c die Größe des Raums in dem eine Abstoßungskraft angewendet wird. Durch die-

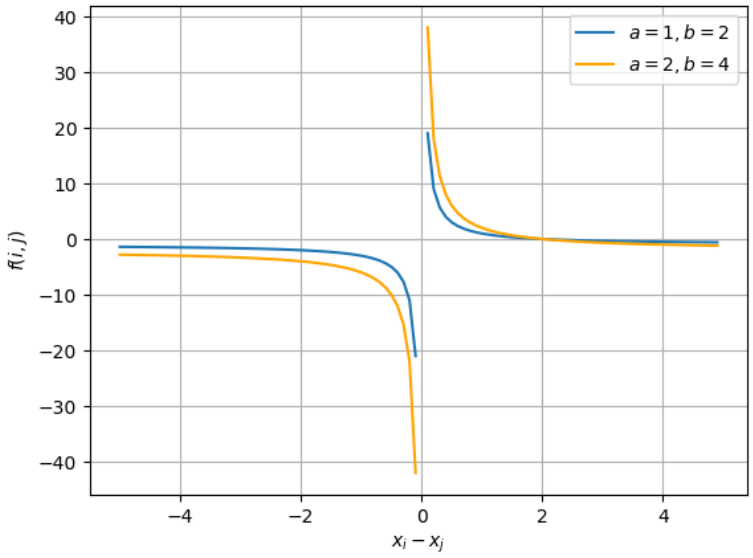


Abbildung 2.8.: Konstante Anziehung und unbegrenzte Abstoßung

se Verwendung ähneln sich die Graphen 2.9 und 2.6 im Repulsion-Teil stark.

$$f_{Att} = a \tag{2.9}$$

$$f_{Rep} = b \exp \frac{-||x_i - x_j||^2}{c} \tag{2.10}$$

Um die Ergebniskraft für die Richtung eines Individuums zu berechnen, müssen alle Einzelkräfte, die gegenüber allen anderen Schwarmteilnehmer wirken, addiert werden. Formel 2.11 stellt diese Aggregation dar.

$$F_i = \sum_{j=0}^{M} \vec{f_i}(i, j) \tag{2.11}$$

Falls eine Zielstellung integriert werden soll, kann dieser Formel noch ein weiterer Bestandteil hinzugefügt werden. Die daraus entstehende in 2.12 abgebildete Summe fügt der Schwarmberechnung eine neue Umgebungskraft hinzu für N

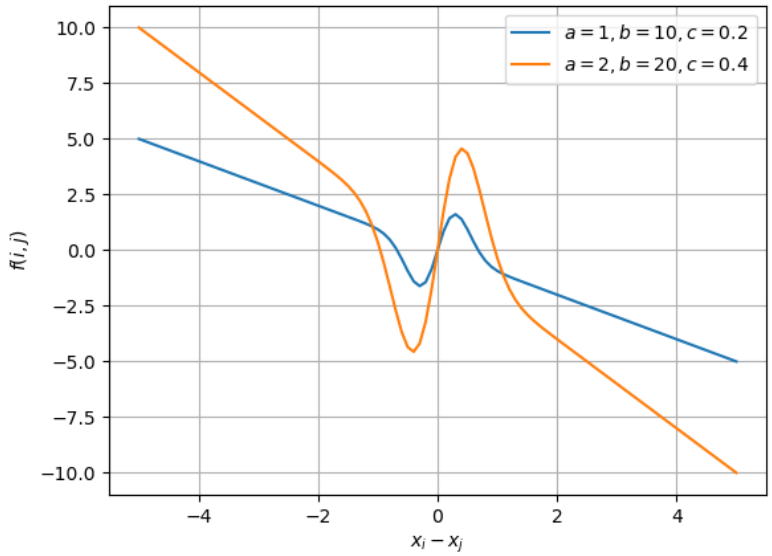


Abbildung 2.9.: Begrenzte Anziehung und begrenzte Abstoßung

viele verschiedene Objekte. Diese können, so wie die Individuen des Schwarms auch eine anziehende und abstoßende Komponente besitzen.

$$F_i = \sum_{j=0}^{M} \vec{f}_i(i,j) + \sum_{k=0}^{N} \vec{g}_i(i,k) \tag{2.12}$$

Um die Stabilität der Formation eines Schwarms zu berechnen, kann der Entropie -und Varianzwert berechnet werden. Die Entropie misst dabei die Verteilung des Schwarms in dem Raum, den er in dem Moment einnimmt. Je höher die Entropie ist, um so gleichmäßiger sind die einzelnen Individuen im Schwarm verteilt. Im speziellen Fall der Quadcopter wird sie mit dem Zentrum des Schwarms und dem Abstand jedes Individuums zu ihm berechnet. Das Zentrum berechnet sich mit Formel 2.13 und der Abstand zum Schwarm mit Formel 2.14.

$$\vec{x_c} = \frac{1}{M} \sum_{i=1}^{M} \vec{x_i}(t) \tag{2.13}$$

$$\vec{d_i} = ||\vec{x_c} - \vec{x_i}|| \tag{2.14}$$

Aus dem Verhältnis eines Abstands zum Abstand aller Individuen zum Zentrum zusammen (Formel: 2.15 kann mithilfe der Funktion 2.16 die Entropie berechnet werden.

$$p_i = \frac{d_i}{\sum_{j=0}^{M} d_j} \tag{2.15}$$

$$Entropie_i = -\sum_{i=1}^{M} p_i \log_2(p_i) \tag{2.16}$$

Zur Entropie kann zusätzlich die Varianz berechnet werden, indem der mittlere Abstand 2.17 aller Individuen zum Zentrum genutzt wird. Die Formel 2.18 berechnet daraus die Varianz des Schwarms zu diesem Zeitpunkt. Die Varianz gibt an, wie viel Raum der Schwarm einnimmt. Eine niedrige Varianz deutet darauf hin, dass die einzelnen Individuen sich sehr nah am Mittelpunkt befinden und der Schwarm so einen sehr kleinen Raum einnimmt.

$$m = \frac{\sum_{i}^{M} d_i}{M} \tag{2.17}$$

$$Varianz_i = -\sum_{i=1}^{M} \frac{(d_i - m)^2}{M} \tag{2.18}$$

2.2. Stand der Wissenschaft

Quadcopter finden immer mehr Verwendung in der Forschung. Sie sind eine populäre Plattform wegen ihrer Agilität, Einfachheit und Vielseitigkeit in ihrer Anwendung[17]. In der Forschung gibt es dabei verschiedene Schwerpunkte. Einige davon sind Unfallvermeidung, Kommunikation, Wegfindung und Schwarmverhalten. Die Anwendungsbereiche steigen stetig von Überwachung, Suchmissionen und Filmaufnahmen über Inspizierung von Gebäuden und Lagern bis hin zu militärischer Verwendung [19][17][5][13].

Vor allem in Rettungsmissionen, bei denen das Opfer erst gefunden werden muss, ist Zeit eine wichtige Komponente. Da der Suchraum sehr groß werden kann, zum Beispiel in einem Gebirge, sind für die Suche sehr viele Kräfte in sowohl personeller als auch finanzieller Natur nötig. Quadcopter-Schwärme sollen die Kosten einschränken und dabei die benötigte Zeit verkürzern [14].

In Künstliche Intelligenz (KI)-Bewegungs-Szenarien sind die zwei am meisten auftretenden Ansätze Steering und Path-Planing. Path-Planing setzt auf globale Wegfindungsansätze wie z.B A* oder Navigationsnetze [7]. Diese Ansätze sind durch ihre Funktionsweise nur sehr schwer auf komplexe dynamische Szenarien anwendbar. Wenn ein Szenario auf physikalischen Grundlagen basiert und so die Ansätze in ihrer Wegfindung auch noch die physikalischen Kräfte einbeziehen muss, ist die Verwendung noch schwieriger [10]. Der zweite Ansatz ist der in Videospielen sehr weit verbreitete Steering-Algorithmus. Steering ist ein lokaler Ansatz, in dem verschiedene Handlungsansätze (Behaviour) einen Richtungsvektor zurückgeben, welcher basierend auf dem jeweiligen Verhalten den optimalen Weg vorschlägt [9][10][11].

Häufig auftretende Behaviour sind Suchen (Seek) und Flüchten (Flee). Die durch die Behaviour berechneten Richtungsvektoren werden zu einem finalen Bewegungsvektor zusammengefasst. Jedoch können bei der Aggregation zum finalen Vektor Problemen auftreten, die zu nicht erwünschtem Verhalten führen können. Beispielweise kann es innerhalb von dynamischen Szenarien zu oszillierender Bewegung kommen, bei der der Agent ständig seine Richtung um 180° dreht. Zusätzlich ist reines Steering anfällig für Dead-Locks [10][11].
Anhand von Abbildung 2.10 ist erkennbar, dass der Agent sich nicht bewegen

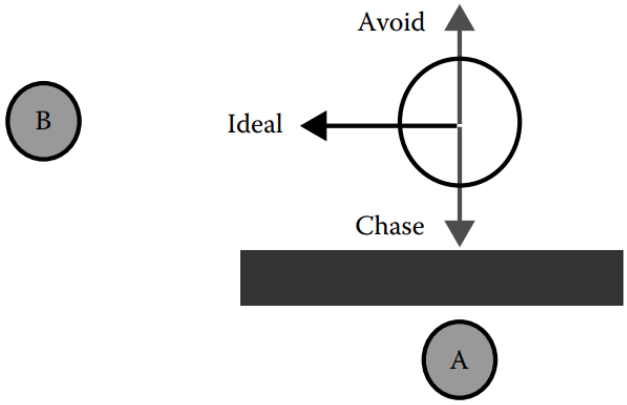


Abbildung 2.10.: Deadlock im Steering-Algorithmus [11]

würde, da der Fluchtvektor an dem Punkt genauso groß ist wie der Suchvektor. Dieses Verhalten kann je nach Anforderung durch Gewichte oder Prioritätsmethoden angepasst werden [10], um solche Probleme teilweise zu vermeiden.

Einen weiteren Lösungsvorschlag hat Andrew Fray mit der Erweiterung von Steering auf Context-Steering entwickelt [11].

2.3. Context-Steering

Wie bereits in Kapitel 2.2 erwähnt, ist Context-Steering eine von Fray [11] entwickelte Erweiterung des Steering-Verhaltens. Wie im Steering gibt es verschiedene Behaviour. Im Context-Steering wird jedem einzelnen Verhalten eine Context-Map zugewiesen, was die große Änderung bezüglich des normalen Steerings darstellt. Die speziellen Maps sind Arrays, wie in Abbildung 2.11 zu erkennen. Im Array steht jeder Index für eine festgelegte Richtung und der In-

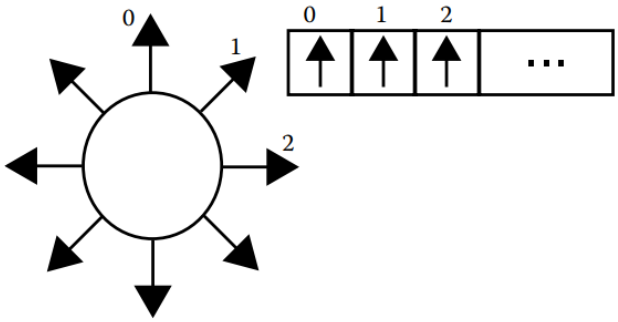


Abbildung 2.11.: Default-Context-Map [11]

halt für die Relevanz, die der Behaviour dieser Richtung zuweist. Dadurch ist es dem Agenten möglich, seine Umgebung wahrzunehmen. So gibt am Ende jedes Simulationsschritts der Behavior keinen Richtungsvektor sondern eine Map zurück, die bei einem Seek-Behaviour wie in Abbildung 2.12 aussehen kann [11]. Dabei weist der Behaviour der Richtung, die direkt auf Punkt A zeigt, einen höheren Interest zu als dem Index Richtung B, da dieser weiter entfernt ist.

Je größer dieses Array ist, um so größer ist die Auflösung der Umgebung. Da jede Map gleich groß ist und so an jedem Index die gleiche Richtung liegt, sind die verschiedenen Maps einfach miteinander zu vergleichen. Die finale Berechnung ist vom Anwender frei bestimmbar. Ist beispielsweise die Gefahr in eine Richtung zu groß, kann er sie blockieren, um den Agenten in seiner

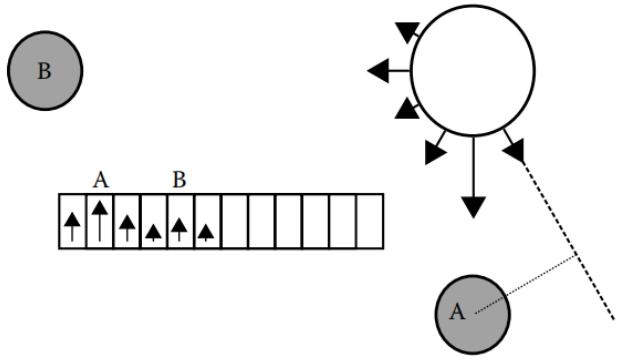


Abbildung 2.12.: Seek-Behaviour mit Context-Map [11]

Richtungswahl auf weniger gefährliche Routen zu begrenzen [11]. Für Context-Steering gibt es viele Ideen, den Algorithmus zu erweitern. So beschreibt Fray in seinem Paper [11] selbst, wie man verschiedene Herausforderungen, die in Context-Steering auftreten mithilfe von Post-Processing bewältigen kann. In einem System mit limitierten Ressourcen muss die Auflösung der Context-Map stark begrenzt werden. Mit einer kleineren Auflösung ist es viel wahrscheinlicher, dass die optimale Richtung zwischen zwei Context-Map-Einträgen liegt. Um trotzdem in die optimale Richtung zu fliegen, stellt Fray unter [11] vor, wie mithilfe von Subslot-Movement dieses Problem gelöst werden kann (Abbildung 2.13). Dabei wird der Gradient über die Context-Map berechnet. Dafür muss die Stärke der Context-Map als Funktion 2.19 dargestellt werden.

$$S = f(x, y) \tag{2.19}$$

Der Gradient ist dann der Vektor mit den partiellen Ableitungen als einzelne Bestandteile 2.20.

$$\nabla S = \left(\frac{\delta S}{\delta x}, \frac{\delta S}{\delta y} \right) \tag{2.20}$$

Der Vektor ∇S gibt daraufhin die Richtung des stärksten Anstiegs innerhalb der Kontextkarte an. Damit kann die Richtung zwischen zwei Context-Map-Einträgen bestimmt werden, wenn dort der Anstieg größer ist.

In anderen Erweiterungen wie dem Paper [10] wird beschrieben, wie Multi-Objective Optimization die Entscheidungsfindung verbessern kann. In Paper [21] wird der gesamte Ablauf von Context-Steering generalisiert und in

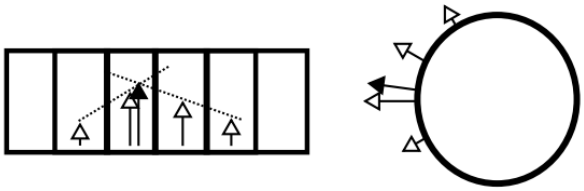


Abbildung 2.13.: Subslot Movement in Context-Steering [11]

verschiedene Blöcke aufgeteilt, die einzeln neu beschrieben werden können. Falls die Parametrisierung ein Problem darstellt, ist es nach Paper [9] möglich, diese innerhalb eines Projektes mit evolutionären Algorithmen herauszufinden.

2.4. Multikriterielle Entscheidungsfindung

Context-Steering kann als Multi-Kriterielles-Optimierungs-Problem (MOOP) aufgefasst werden [12]. Die einzelnen Context-Maps sind jeweils einzelne Kriterien. Eine Interest-Map zeigt die Richtungen an, in denen jeweils Interest liegt, wobei eine Danger-Map die Richtungen zeigt, die am stärksten vermieden werden sollen. Da beide die gleichen Richtungen anzeigen und der Wert in ihnen sich unterscheidet, können sie auch in Abbildung 2.14 dargestellt werden. Die Aufgabe der multikriteriellen Entscheidungsfindung ist es, die pareto-optimalen Lösungen zu finden [8], indem die besten Richtungen als Lösungen extrahiert werden. Um dies zu erreichen, gibt es eine Reihe von Maßnahmen, um die entstandenen Ergebnisse zu so aufzubereiten, dass die pareto-optimale Lösung gefunden werden kann. Dafür wird das MOPin Formel 2.21 mithilfe einiger Methoden wie zum Beispiel Gewichtete-Summe und ϵ-Einschränkung, in ein Einzelaufgaben-Problem (SOP) umgewandelt [8].

$$\min/\max\left(f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})\right),\ \vec{x}\ \epsilon\ \text{Solution-Space} \tag{2.21}$$

So kann jedes einzelne Kriterium seine eigene pareto-optimale Lösung finden [8]. Da Context-Steering aus verschiedenen Maps besteht und damit auch ein MOP darstellt [10], können diese Maßnahmen angewendet werden. Mit ihnen soll das Ergebnis der Flugrichtung, welche beim Zusammenführen der unterschiedlichen Maps entsteht, erheblich verändert werden können. Im folgenden wird auf einige dieser A-Priori-Methoden eingegangen.

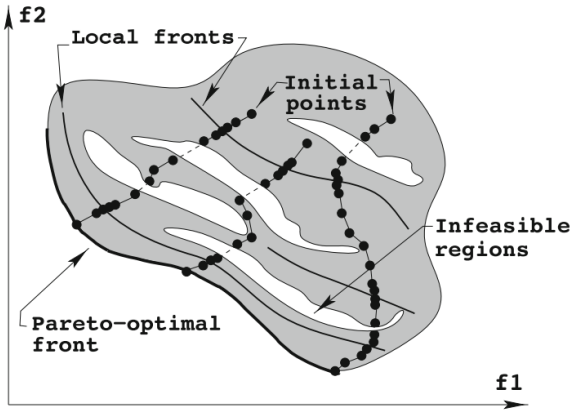


Abbildung 2.14.: Multi-Kriterielle Entscheidungsfindung [8]

Gewichtete-Summe Bei der Gewichteten-Summe-Methode wird das MOP aus Formel 2.21 in ein SOP mit dem Ausdruck 2.22 umgewandelt [15]. Diese eine Zielsetzung kann dann mithilfe der Gewichte minimiert oder maximiert werden.

$$\min/\max \left(\sum_{i=1}^{k} w_i f_i \right) \text{ mit } \sum_{i=1}^{m} w_i = 1 \tag{2.22}$$

Mit dieser Methode kann innerhalb von konvexen Problemen jede pareto-optimale Lösung gefunden werden. Ist das Problem allerdings konkav, sind einige Lösungen nur sehr schwer zu finden. Zusätzlich ist die Parametrisierung der Gewichte schwierig, da der Zusammenhang der verschiedenen Kriterien nicht eindeutig ist und sie nicht linear zueinander stehen. So können kleine Änderungen am Gewicht die Lösungen stark beeinflussen [15].

ϵ-Constraint Hierbei wird ein Kriterium ausgewählt, welches optimiert wird während alle anderen Funktionen in Einschränkungen umgewandelt werden [15]. Das Problem wird so zu einem SOP der Form 2.23.

$$\min\left(f_l(\vec{x})\right) \text{ unterliegt } f_j(\vec{x}) \leq \epsilon_j, \text{ für alle } j = 1, ..., m, j \neq l \tag{2.23}$$

Alles was bei einem Minmierungsproblem unterhalb dieser frei einstellbaren Schranke liegt, ist pareto-optimal. Lösungen die außerhalb dieser Einschränkung liegen nicht [15]. Ein Maximierungsproblem ist auch möglich, indem das Vergleichs-Symbol vom ϵ-Constraint umgedreht wird und

damit der Funktionswert aus $f_i(\vec{x})$ nur größer sein muss als die gewählte Grenze.

Hybrid Der hybride Ansatz vereint die beiden oberen Ansätze. Dadurch entsteht die Formel 2.24. Dabei werden zunächst die Lösungen, welche außerhalb des ϵ-Constraints liegen entfernt und auf Basis der Verbleibenden der Gewichtete-Summe-Ansatz ausgeführt. Der Gewichtete-Summe-Anteil der Funktion bleibt unverändert.

$$\min \left(\sum_{i=1}^{k} w_i f_i \right) \text{ solange } f_i(\vec{x}) \leq \epsilon_i \tag{2.24}$$

2.5. Paparazzi UAV

Paparazzi UAV (Unmanned Aerial Vehicle) ist ein in 2003 gegründetes Open-Source Projekt mit dem Schwerpunkt "Autonomes Fliegen" [2]. Es beinhaltet Hard -und Softwarelösungen und eine Liste an weiteren Features, die für das autonome Fliegen von Drohnen notwendig sind [2].

Die von Paparazzi genutzte Simulationsumgebung heißt JSBSim, ein "open source flight dynamics model". Es wird in vielen Anwendungen wie FlightGear oder als Plugin in der Unreal-Engine verwendet. JSBSim ist ein physikalisch-mathematisches Modell, das Bewegung von Luftfahrzeugen simuliert. Dabei werden die Kräfte und Flugeigenschaften des eigenen Fahrzeugs eingerechnet, wie zum Beispiel Gewicht, Flügelspanne und Aerodynamic. Für letzteres enthält JSBSim auch die Möglichkeit, Wind zu simulieren, um eine möglichst realistische Umgebung zu schaffen [4]. Innerhalb einer Simulation wird immer nur ein Flugobjekt emuliert. Wenn mehrere Quadcopter simuliert werden sollen, müssen mehrere Anwendungen parallel gestartet werden. So existiert jeder Kopter in seiner eigenen Welt und kommuniziert mit Agenten aus anderen Simulationen. Die Kommunikation findet dabei über Paparazzi statt. Dadurch ist es nicht möglich, Unfälle zu erzeugen, da keine physikalische Interaktion zwischen Koptern simuliert werden kann.

Die einzelnen Bestandteile der Paparazzi-Architektur, die während der JSBSim-Simulation verwendet werden, werden in der Abbildung 2.15 gezeigt [1]. Damit verschiedene viele Agenten Informationen austauschen können, existiert der Data-Link, welcher alle Hardware Komponenten miteinander verbindet. Die gesendeten Nachrichten werden so zwischen dem Server und

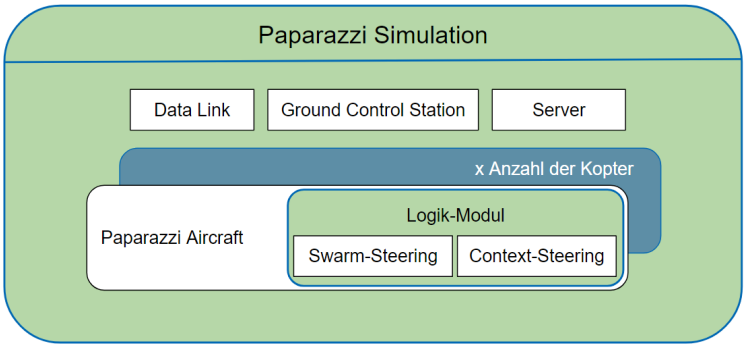


Abbildung 2.15.: Paparazzi-Architektur

den Drohnen verteilt. Der Server speichert, vor-verarbeitet und verteilt jede Nachricht an die richtige Stelle. Die Ground Control Station (GCS) erhält ihre vor-verarbeiteten Nachrichten vom Server und stellt diese im grafischen Overlay dar. Zusätzlich steuert die GCS den Data-Link. Die in der Simulation fliegenden Drohnen werden dabei von Paparazzi als Paparazzi-Aircraft dargestellt, welches verschiedenste Module enthalten kann [1]. Wie das im Speziellen aussehen kann, wird in Kapitel 3 genauer erklärt.

2.6. Quadcopter

Als Grundlage für die Quadcopter dient die 2016 angelegte FINken-Platform [6], welche in Abbildung 2.16 dargestellt ist. Sie diente dazu, Schwärme von Indoor-Robotern zu evaluieren [6].

Die Kopter haben die folgende Konfiguration:

- X-frame mit 200 mm diagonalem Motorabstand
- Li-Po Batterie für 10 min Flug (3 Zellen, 900 mAh)
- Motor: MN1804-20: 2400 kV, max 10 A, 5×3” Propeller
- Gesamtgewicht von 350 g
- Integrierter Autopilot mit 10-Axis-IMU (Paparazzi Lisa/MX 2.1)
- Fernsteuerung mit 2.4 GHz Spektrumprotokoll
- Kommunikation basiert auf 802.15.4

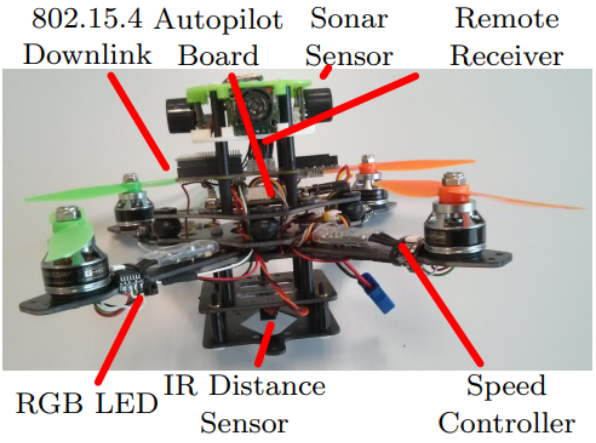


Abbildung 2.16.: Quadcopter-Architektur [6]

- SD-Card logging via SPI
- Infrarot-Höhensensor (Sharp GP2Y0A60SZLF)
- Ultraschall-Objektsensoren (Max-Botix MB1232)

Für die Programmierung wurde von Anfang an Paparazzi verwendet [6]. Innerhalb von Paparazzi existierte deshalb bereits ein OVGU-Aircraft. Wie an Abbildung 2.17 zu erkennen ist, besteht innerhalb von Paparazzi ein Aircraft aus sechs verschiedenen Modulen. Das erste Modul ist das Airframe. Hier wird

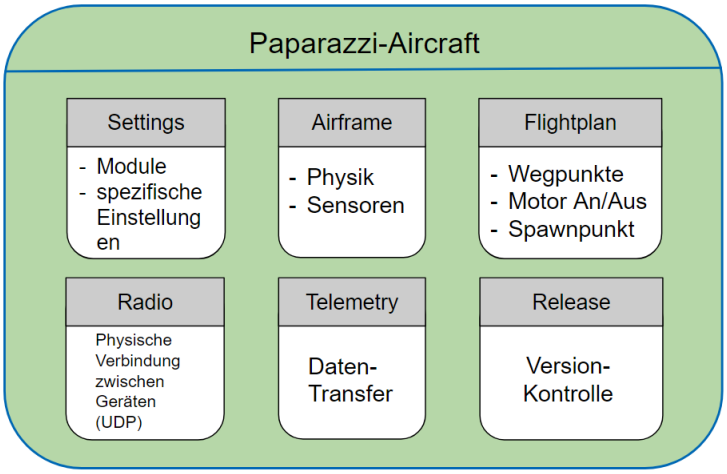


Abbildung 2.17.: Quadcopter-Architektur

festgelegt, welche Sensoren und Module wie initialisiert werden und um was

für einen Typ Aircraft es sich handelt. Wenn ein neues Modul hinzugefügt werden soll, muss es hier initialisiert werden. Ein weiterer Bestandteil der Aircrafts sind die Flightplans (Flugpläne). Diese enthalten die Startparameter wie Höhe und Ort und die verschiedenen Wegpunkte, die der Kopter kennen soll. Zusätzlich können hier Flugrouten und erste Logiken implementiert werden. Innerhalb der Simulation kann mit dem Flightplan auch das Flugobjekt manuell gesteuert werden. Um gewisse Voraus-Einstellungen tätigen zu können, existiert das Settings-Modul. Dort können verschiedene Features oder auch Notwendigkeiten, wie zum Beispiel der Stabilisator, hinzugefügt werden. Um mit anderen Koptern kommunizieren zu können, benötigt das Aircraft das Telemetrie-Modul. Diese Verbindung wird mit dem Radio-Modul aufgestellt. Innerhalb der Simulation findet der Nachrichten-Austausch standardmäßig mit UDP-Paketen statt. Innerhalb des eigenen Netzwerks können so keine Pakete verloren gehen. Innerhalb eines Realwelt-Szenarios ist diese Art der Kommunikation nicht möglich und es kann zum Verlust von Paketen kommen. Die speziellen Module die vom OVGU-Aircraft verwendet werden können in Tabelle 2.1 eingesehen werden.

Modul	Beschreibung
ahrs_int_cmpl_quat	Höhen/-Richtungs Referenz System
air_data	Physikalische Eigenschaften der Luft (Temperatur,Druck)
geo_mag	Berechnung zum normalisierten geomagnetischen Feld-Vektor
gps	Initialisierung von GPS
gps_ubx_ucenter	Treiber für u-blox GPS-Modul
guidance_rotorcraft	Generieren der Flugkommandos
imu_common	Initialisierung der Trägheitsmesseinheit
imu_nav_basic_rotorcraft	Standard Navigations-Pattern und Flugplan-Handhabung
imu_stabilization_int_quad	Stabilisator für das Flugobjekt/ Umsetzung der Flugkommandos

Tabelle 2.1.: Module innerhalb der Paparazzi-Drohne

3. Copter-Swarm-Steering

In diesem Kapitel wird über die verschiedenen Konfigurationsmöglichkeiten der beiden Ansätze Context -und Swarm-Steering gesprochen. Dabei wird zuerst im nächsten Abschnitt 3.1 eine Übersicht über alle Möglichkeiten gegeben. Darüber hinaus wird besprochen, wie Context-Steering von 2D auf 3D übertragen werden kann. In Kapitel 3.2 wird erklärt, wie sich Kopter innerhalb eines Schwarms gezielt bewegen und im Folgeabschnitt 3.3 welche Context-Maps möglich sind. Des Weiteren wird der Entscheidungsprozess in Kapitel 3.4 für den gewählten Richtungsvektor erörtert und die finale Implementierung in der Sektion 3.5 kurz erklärt. Die für die Formeln notwendigen Variablen werden in Tabelle 3.1 definiert und beschrieben.

3.1. Übersicht

Um der Forschungsfrage nachzugehen, mussten zunächst einige Entscheidungen bezüglich der Projekt -und Implementationsgrundlagen getroffen werden. Da Paparazzi im Lehrstuhl bereits in Projekten verwendet wurde, war es naheliegend, das Programm mit der JSB-Simulation zu verwenden. In Paparazzi sind viele Grundlagen bereits vorhanden wie zum Beispiel das im Lehrstuhl verwendete Quadcoptermodell und ein Kommunikationsmodul welches den Koptern ermöglicht, Informationen auszutauschen. So mussten neben kleinen Anpassungen nur noch die beiden Logik-Scripte designed und implementiert werden. Um die Effektivität von Context-Steering zu testen, muss ein Vergleich aufgestellt werden. Da innerhalb des Projekts mehrere Kopter fliegen und diese sich als Schwarm verhalten sollen, ist es naheliegend, den Ansatz der Grundsteuerung der Schwarmrobotik gegenüber zu stellen. Somit wird parallel zum Context-Steering das Kräftesystem mit Attraction-Repulsion (Swarm-Steering) implementiert, welches als sinnvolle Vergleichsbasis dient. Für den Ansatz wird die Formel 2.12 verwendet, die aus dem Swarm-

Variable	Beschreibung
N	Anzahl der Vektoren im größten Ring
n	Anzahl der Vektoren in einem Ring
Θ	Polarwinkel
Φ	Azimutwinkel
$\|S\|$	Anzahl der Vektoren
r	Radius des Kreises
$\|\Phi\|$	Anzahl der Vektoren in einem bestimmten Kreis
s	Ein bestimmter Vektor
$\|H\|$	Anzahl an Vektoren insgesamt
k_a	Skalierungsfaktor-Attraction
k_r	Skalierungsfaktor-Repulsion
c	Formvariable der Abstoßungskurve
$\vec{b_i}$	Basis-Richtungsvektor der Contextmap an Stelle j
d_i	Die i-te Drohne
d_{max}	Maximale Anzahl an Drohnen
q_{max}	Maximale Anzahl an Interest-Points
w_{max}	Maximale Anzahl an Wänden
p_0	Position der Drohne für die die Berechnung ausgeführt wird
p_j	Position der j-sten Drohne
q_j	Position des j-sten Interest-Points
w_j	Position der i-ten Wand
$\vec{w}_{ij}$	Schnittpunkt zwischen Wand j und Vektor i
$\vec{t}_{i,j}$	Normiertes Skalarprodukt von i und j
$\vec{v}$	Geschwindikeitsvektor der Drohne für die Berechnung
$\vec{v}_{max}$	Vom Ansatz vorgegeben maximale Geschwindigkeit
$\vec{r}_{ij}$	Richtungsvektor zwischen i und j
M	Basis-Richtungs-Map
P	Menge der Drohnenpositionen

Tabelle 3.1.: Variablen-Definition

Attraction-Repulsion-Teil aus Formel 2.1 und dem Umwelt-Teil besteht. Der Umwelt-Anteil wird in 4.1 genauer definiert.

Mit der Verwendung von Context-Steering müssen einige Fragen geklärt werden. Bevor der Ansatz von 2D auf 3D übertragen wurde, musste entschieden werden, wie groß die Context-Map in 2D sein soll. Anhand dessen kann mit verschiedenen Umwandlungsansätzen aus dem Sichtkreis eine Kugel konstruieren werden. Ein Ansatz ist es, die Kugel in Ringe aufzuteilen und Vektoren an diesen Ringen zu verteilen. Dieser wird im folgenden erklärt.

Für eine Umwandlung von 2D zu 3D wird die 2D-Context-Map als Äquator genutzt. Damit würden sich die Positionen der neuen Ringe mithilfe von Formel 3.1 berechnen lassen.

$$\Theta_i = \frac{\pi i}{N-1} \tag{3.1}$$

Θ_i ist dabei der Winkel, den alle Richtungsvektoren voneinander haben, um gleichmäßig auf dem Ring verteilt zu sein. Formel 3.2 berechnet, wie viele Vektoren $|Phi|$ pro Ring benötigt werden.

$$|\Phi| = \max(1, \lceil N \sin \Theta \rceil) \tag{3.2}$$

Das Maximum ist dabei direkt in der Mitte mit $|\Theta| = N$ und an den beiden Polen ist $|\Theta| = 1$ das Minimum. Natürlich kann auch pro Ring die gleiche Anzahl an Vektoren benutzt und so dieser Schritt übersprungen werden. Wenn die Anzahl an Vektoren pro Ring gleich sein soll, wird dieser einzelne Winkel $\Phi_{i,j}$ pro Ring mit der Formel 3.3 ermittelt.

$$\Phi_j = \frac{2\pi j}{|\Phi|} \tag{3.3}$$

Anhand von Θ und Φ kann jetzt jeder einzelne Vektor, der auf der Kugel liegt, mit 3.4 kalkuliert werden.

$$\begin{bmatrix} r \sin \Theta \cos \Phi \\ r \sin \Phi \sin \Theta \\ r \cos \Theta \end{bmatrix} \tag{3.4}$$

Um die genaue Menge an Vektoren zu errechnen, die die Anwendung benötigt, kann die Formel 3.5 genutzt werden.

$$|H| = \sum_{i=0}^{N-2} \max\left(1, \lceil N \sin \frac{\pi i}{N-2} \rceil\right) \tag{3.5}$$

So kann für die Simulation eine beliebige Context-Map-Größe verwendet werden. Die Wahl der Auflösung spielt eine große Rolle, denn je größer sie ist, um so größer ist die Abtastung der Umgebung und es können Objekte viel genauer eingeordnet werden. Allerdings bedeutet das auch einen wesentlich höheren Speicher -und Rechenaufwand. Anhand von Abbildung 3.1 ist zu erkennen, dass der Anstieg von Vektoren bei gleicher Anzahl pro Ring (Orange) wesentlich größer ist als bei abnehmender Anzahl zum Pol (Blau). Dieser Unterschied, der vor allem die Berechnungszeit in einer zeitkritischen Anwendung beeinflusst führt dazu, dass in dieser Arbeit pro Ring eine unterschiedliche Menge an Vektoren benutzt wird.

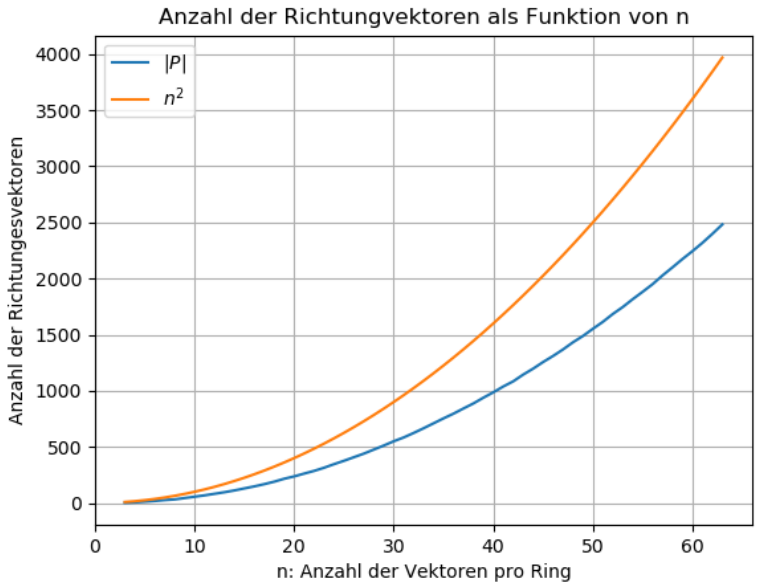


Abbildung 3.1.: Anzahl der Vektoren für gleiche und unterschiedliche Ringe

Ein weiterer Ansatz wäre es, die Richtungsvektoren in jedem Simulationsschritt neu, dynamisch und zufällig zu berechnen. Durch die ständige Neuberechnung können Objekte, die bei sonst statischen Context-Maps zwischen den Richtungsvektoren liegen und somit nicht betrachtet werden, auch erfasst werden. Allerdings bringt das unerwünschten Zufall in das System, sodass Objekte über mehrere Zeitschritte nicht betrachtet werden können, weil kein Vektor in ihrer Richtung generiert wurde. Zusätzlich bedeutet eine Neuberechnung in jedem Simulationsschritt viel größeren Rechnungsaufwand. Außerdem können einige Logik-Ansätze, die beispielsweise das Speichern der Context-Map

aus dem letzten Simulationsschritt beinhalten, nur schwer angewendet werden. Bei der dynamischen Berechnung haben alle Context-Maps in jedem Simulationsschritt andere Richtungsvektoren und sind damit mit den alten nur sehr schwer vergleichbar. Demnach eignet sich das dynamische Neuverteilen der Richtungsvektoren einer Context-Map nicht für diese Anwendung. Zusätzlich zur Auflösung muss geklärt werden, welche und wie viele Context-Maps es geben soll. Im Groben werden diese in drei verschiedene Typen unterteilt. Behaviour die das Schwarmverhalten bewerten, die Umgebungsinteraktion beobachten und die Physik betrachten. So muss aus diesen verschiedenen Behaviours eine Result-Map erstellt werden, die für die Richtungsfindung genutzt werden soll.

Zusammenfassend werden in Abbildung 3.2 beide Ansätze verglichen. Dabei sind der Ausgangspunkt für beide die Rohdaten, die von der Simulation inform von Position der Kopter, Interestpoints und Wände bereit gestellt werden. Diese werden je nach Algorithmus benutzt, um die Context-Maps oder die jeweiligen Kräfte zu berechnen. Auf diesen Ergebnissen werden die aus dem Entscheidungsfindungskapitel 3.4 besprochenen Ansätze verwendet, um daraus die Endergebnisse der Steering-Algorithmen zu berechnen. Beim Context-Steering wird dabei die Result-Map berechnet aus der die finale Richtung ausgewählt wird. Swarm-Steering überspringt diesen Schritt, indem der berechnete finale Kraftvektor aus Umwelt-Kraft und Attraction-Repulsion direkt die Richtung darstellt, in die die Kopter fliegen sollen. Wie diese Kräfte im Schwarm-System berechnet werden, muss noch festgelegt werden. Innerhalb von Kapitel 2.1 wurden verschiedene Ansätze besprochen, wie die Kräfteberechnung zwischen den Koptern stattfinden soll. In Kapitel 4.1 wird besprochenn welche von ihnen benutzt werden und wie genau die Umgebungskraft sinvoll mit eingerechnet wird.

3.2. Bewegungssteuerung

Um einen Kopter innerhalb von Paparazzi automatisiert zu steuern, gibt es für jedes Flugobjekt einen Flight-Plan. Hier können beliebig viele Wegpunkte aneinander gereiht werden, die der Kopter abfliegen soll. Um einen Flugobjekt dynamisch fliegen zu lassen, müssen diese Wegpunkte dynamisch verschoben werden. Paparazzi bietet die Möglichkeit, in Echtzeit Wegpunkte zu verschie-

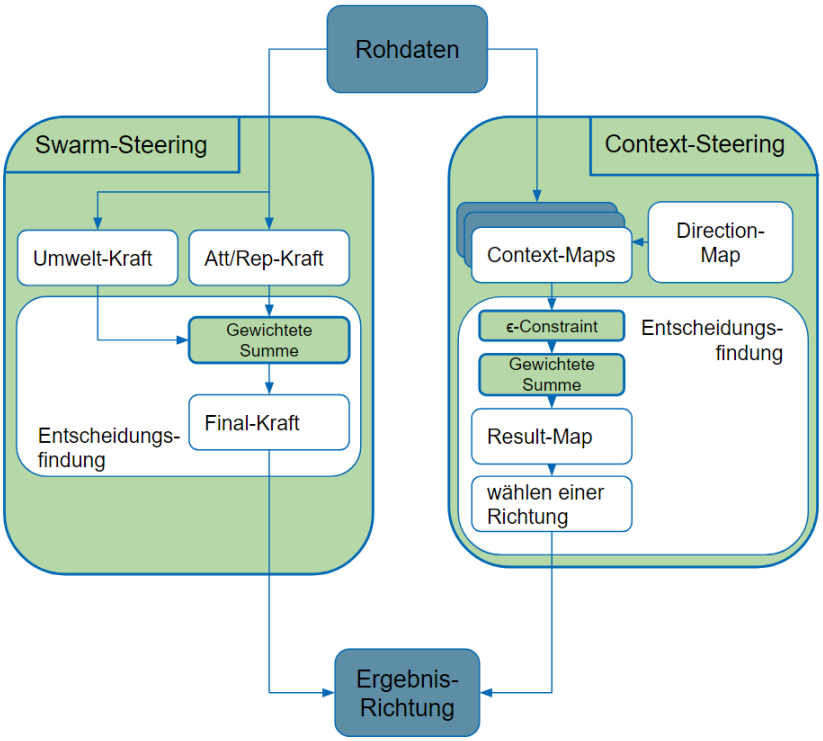


Abbildung 3.2.: Vergleich zwischen Context -und Swarm-Steering

ben und zu erfassen. Wenn beispielsweise dem Flugobjekt gesagt wird, es soll Wegpunkt 1 anfliegen und dieser befindet sich auf einer kreisförmigen Umlaufbahn, fliegt der Agent einen Kreis. Dabei legt das Logik-Modul fest, wohin der Wegpunkt verschoben wird. Je entfernter der Wegpunkt vom Kopter gesetzt wird, um so schneller fliegt der Kopter. Innerhalb des Context-Steering-Moduls wird ein Ergebnis-Vektor berechnet, indem die gewählte Richtung auf die eigene Position addiert und letztendlich mit einem Geschwindigkeits-Skalar multipliziert wird:

$$\vec{R}_{Context-Steering} = p_0 + \vec{r_i} \cdot V_{max,i} \tag{3.6}$$

Innerhalb des Swarm-Steerings wird der Geschwindigkeitsskalar nicht benötigt, da in der Addition der verschiedenen Richtungskräfte automatisch die gewählte Geschwindigkeit im Richtungsvektor enthalten ist.

$$\vec{R}_{Att/Rep} = p_0 + \vec{r_i} \tag{3.7}$$

Die Informationen für die Logik-Berechnungen liefert Paparazzi in Form eines Nachrichtensystems. Hier können die Kopter untereinander Nachrichten verschicken. Aufgrund einiger vorangegangener Tests konnte ermittelt werden, dass dieses System eine Verzögerung von bis zu sechs Sekunden vom Senden bis zum Empfangen der Nachricht hat. Bei einem Flug in einem Schwarm, bei dem die Kopter teilweise einen Abstand von weniger als einem Meter haben, sind sechs Sekunden fatal für die Sicherheit der Kopter und deren Umgebung. Da dieser Versatz in einer dynamischen Echtzeitsimulation viel zu hoch ist, wurde die Positionsverarbeitung der Kopter ausgelagert.

In einem separat implementierten System besitzt jeder Kopter eine einzigartige ID und eine Position. Somit kann in Echtzeit erfasst werden, wie viele Kopter existieren und wo sie sich befinden, falls dies sich in Laufzeit verändert. Dadurch wird allerdings der Grad des Realismus gesenkt, da innerhalb der Realwelt keine Kommunikation ohne Verzögerung möglich ist. Dies muss innerhalb der Auswertung in Betracht gezogen werden.

3.3. Erläuterung verschiedener Context Maps

Im Folgenden werden alle Context-Maps erläutert, die in dieser Arbeit verwendet wurden. Sie wurden drei unterschiedlichen Typen zugeordnet, die ihren

Handlungsdirektiven entsprechen. Für einige Context-Maps wird ein normiertes Skalarprodukt t einem Contest-Map-Richtungsvektor $\vec{b_i}$ und einem weiteren Context-Map abhängigen Vektor berechnet, das in Formel 3.8 dargestellt ist.

$$t_{i,j} = \frac{\vec{b_i} \cdot \vec{r_j}}{|\vec{b_i}| \cdot |\vec{r_j}|} \tag{3.8}$$

3.3.1. Dronen -und Schwarmverhalten

In dieser Kategorie befinden sich die Context-Maps, die das Schwarmverhalten der Drohnen simulieren sollen. Dafür können die Grundsätze, die in Kapitel 2.1 beschrieben wurden genutzt werden.

Attraction-Map Dronen haben eine Anziehung untereinander, damit sie als Schwarm zusammen bleiben. Je weiter die Drohnen voneinander weg sind, um so stärker ist diese Anziehung. So verstärkt jede Drohne den Wert der Context-Map in der jeweils liegenden Richtung. Der zugewiesene Wert, der auf die Context-Map addiert wird, wird abgeschwächt, wenn die Richtung der Drohne nicht genau auf einem der Basis-Richtungsvektoren liegt. Eine mögliche Lösung zur Berechnung ist die in Kapitel 2.1 erwähnte lineare Anziehung; dargestellt in Formel 3.9. Hier ist die Attraktion noch mit dem normierten Skalarprodukt aus Formel 3.8 multipliziert. Dieser Faktor verändert die Wertigkeit der Anziehung. Falls t kleiner ist als 0.7, was ungefähr ein Abweichung von mehr als 45° darstellt, wird in dieser Richtung der Context-Map kein Wert für die Anziehung der Nachbar-Drohne zugewiesen.

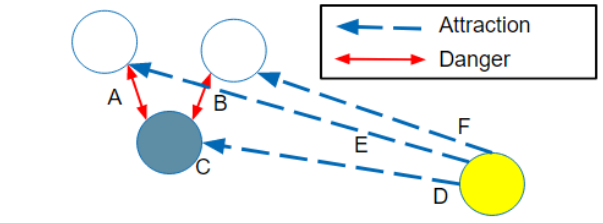
$$A_i = \sum_{j=1}^{d_{max}} t_{i,j} k_a ||p_j - p_0|| \text{ für } t_{i,j} > 0.7 \tag{3.9}$$

DangerDrone-Map Im Schwarm ist das Gegenstück zur Anziehung die Abstoßung. Um zu verhindern, dass die Drohnen ineinander fliegen, wird zusätzlich zur Anziehungskraft jedem Kopter eine Gefahr zugewiesen. Je näher einzelne Nachbarn einem Kopter sind, um so größer ist die Gefahr in deren Richtung. Somit ist die Repulsion-Kraft in kleineren Entfernungen größer als die Anziehungskraft und der Kopter wird abgestoßen. Die in Kapitel 2.1 erläuterten Ansätze können hier auch verwendet werden.

In Formel 3.10 wird eine Version der Formel 2.10 begrenzten Abstoßung verwendet.

$$D_i = \sum_{j=1}^{d_{max}} k_r \exp\left(\frac{-||p_j - p_0||^2}{c}\right) \tag{3.10}$$

Abbildung 3.3 zeigt, wie der Schwarm aufgrund dieser beiden Behaviour sich verhalten soll. Die Drohne, die weiter entfernt ist, soll sich dem Schwarm wieder annähern. Gleichzeitig wird verhindert, dass die anderen Kopter sich zu nahe kommen, da ihr Danger-Wert genauso groß ist wie der Attraction-Wert.



	A	B	C
Attraction:	4	4	2
Danger:	4	4	0

	D	E	F
Attraction:	2	2	1
Danger:	0	0	0

Abbildung 3.3.: Attraction und Danger im Schwarm

3.3.2. Umgebungsverhalten

Diese Kategorie enthält die Context-Maps, welche das Verhalten des Kopter mit seiner Umwelt simulieren sollen.

Interest-Map Jeder Kopter soll eine Anzahl von Zielen erreichen. Diese Interest-Punkt (IP) werden mithilfe der Interest-Map verarbeitet. Je näher der Punkt liegt, um so stärker ist der zugewiesene Wert. Allerdings gibt es ein Minimalinteresse, um zu garantieren, dass jeder Punkt bei der Richtungswahl immer in Betracht gezogen wird. Wie auch in der Attraction-Map wird in der Interest-Map der zugewiesene Wert aus der

Summe der Abstände der einzelnen Punkte und ihrer Winkelabweichung t berechnet.

$$I_i = \sum_{j=1}^{q_{max}} \frac{t_{i,j}}{|\vec{r}_{p_0 q_j}|} \text{ für } t_{i,j} > 0.7 \tag{3.11}$$

DangerWall-Map Der Platz über den die Drohnen in der Simulation fliegen dürfen ist begrenzt. So wird jeder Richtung der Context-Map ein Gefahren-Wert zugewiesen, der von der Entfernung der umliegenden Wände abhängt.

$$W_i = \sum_{j=1}^{w_{max}} \frac{1}{|\vec{r}_{p_0 w_{p_0 j}}|} \tag{3.12}$$

Abbildung 3.4 zeigt, wie der Kopter sich für den weiter entfernten IP entscheidet. Der näher liegende Punkt wird durch eine Wand blockiert und die DangerWall-Map blockiert diese Richtung.

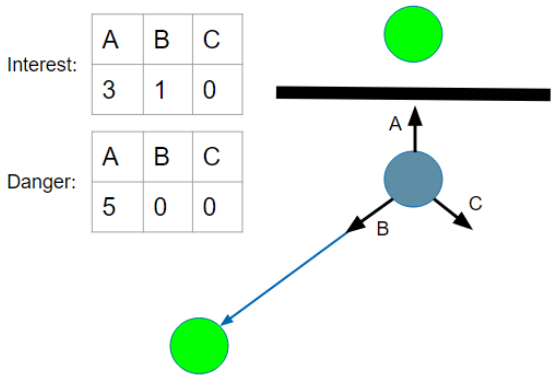


Abbildung 3.4.: Interest und Danger in der Umgebung

3.3.3. Physikalisches Verhalten

In der letzte Kategorie werden Context-Maps abgebildet, die die physikalischen Vorgänge der Simulation darstellen.

Velocity-Map Die zugewiesenen Velocity-Map-Werte 3.13 berechnen sich aus der momentanen Geschwindigkeit multipliziert mit dem normierten Skalarprodukt aus der Basisrichtung und der Richtung der Geschwindigkeit. Solange das normierte Skalarprodukt größer als Null ist und damit beide Vektoren einen kleineren Winkel als 180° haben, werden die Werte in der Karte gespeichert. Mit der Map soll verhindert werden, dass der Kopter nicht zwischen zwei Punkten hin und her fliegt, ohne einen der beiden zu erreichen. Damit gibt es immer eine Gewichtung in die aktuelle Flugrichtung, die größer wird, wenn die Drohne schneller fliegt.

$$V_i = t_{i,v_{norm}} |\vec{v}| \text{ für } t_{i,v_{norm}} > 0 \tag{3.13}$$

MaxVelocity-Map Diese Map wird benötigt, um eine Maximalgeschwindigkeit aus Formel 3.14 zu berechnen. Anhand der Danger-Werte aus Wand und Drohne wird beschlossen, wie schnell die Drohne am Ende in die ausgewählte Richtung fliegen kann.

$$V_{max,i} = v_{max} - D_i - W_i \tag{3.14}$$

In Abbildung 3.5 ist zu sehen, wie die Drohne sich für eine Richtung entscheidet in der weniger Interest besteht, sie aber ihre eigene Richtung nur minimal ändern muss.

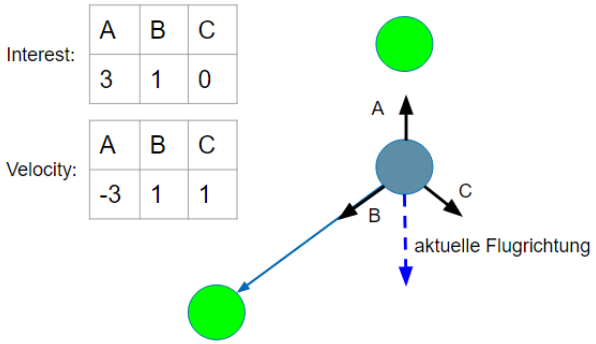


Abbildung 3.5.: Einfluss von Geschwindigkeit auf die Richtungswahl

3.4. Entscheidungsfindung

Um am Ende einen Richtungsvektor für die Flugrichtung zu wählen, muss beim Context-Steering zunächst eine Result-Map aus den einzelnen besprochenen Context-Maps erzeugt werden. Bevor die einzelnen Context-Maps mit der Formel 3.15 addiert werden, müssen sie vor-verarbeitet werden.

$$R_i = A_i + I_i + V_i - D_i - W_i \tag{3.15}$$

Wie bereits im multikriteriellen Entscheidungsfindungs-Kapitel 2.4 besprochen, kann beispielsweise jede Map gewichtet werden, um einzelne Behaviour hervorhebend oder abschwächend in das Gesamtergebnis einfließen zu lassen. Der Vorteil darin ist, dass die Ergebnisse so wesentlich stärker feinjustiert werden können, allerdings zu kosten von mehr Parametern, die eingestellt werden müssen. Das Einstellen dieser Gewichte kann zu einer sehr zeitintensiven Aufgabe werden. Neben dieser Feinjustierung kann es sein, dass ein Wert eine so ausschlaggebende Stärke erreicht, dass er extra bearbeitet werden muss. Beispielsweise ist die Gefahr zu hoch, sodass die Richtung in der sie liegt gesperrt werden muss. Dazu kann das ϵ-Constraint aufgestellt werden, das sobald die Schwelle erreicht wird, den überschreitenden Wert für die Auswertung löscht.

Dadurch entsteht die Möglichkeit, Extremwerte extra zu erfassen und zu löschen. Wie genau diese Schwelle eingestellt wird, damit nur diese Extremwerte behandelt werden, ist genau wie bei den Gewichten eine große Herausforderung und kann bei falscher Einstellung zu unerwünschtem Verhalten führen. Wenn beispielsweise die Gefahren-Grenze zu klein eingestellt wird, kann es sein, dass der Schwarm nicht mehr durch enge Räume fliegen kann. Trotzdem ist es sinnvoll, beide angesprochenen Ansätze zu verwenden, da sie das Nachjustieren der einzelnen Funktionen ermöglichen. Dadurch kann das kontinuierliche Ändern der Funktionen verhindert werden. Mit beiden im Entscheidungsfindungskapitel 2.4 besprochenen Ansätzen kann eine bessere Result-Map berechnet werden. Aus dieser wird die finale Richtung ausgewählt. Im einfachsten Fall wird die Map durchiteriert und der beste Wert genommen.

Für die Richtungswahl beim Swarm-Steering muss entschieden werden, inwiefern die beiden Kräftesysteme Attraction/Repulsion und Umgebung zusammen geführt werden. Wenn die Grundformel für Attraction/Repulsion aus der Basisformel 2.1 betrachtet wird, kann sie auch auf das Umgebungssystem angewendet werden, indem anstatt der Nachbardrohnen die Wände und Interest-Points betrachtet werden. Um beide Systeme zu verrechnen können sie wie

in Formel 3.16 gezeigt addiert werden, um einen finalen Ergebnis-Vektor zu erhalten.

$$\vec{f} = \vec{f}_{Att/Rep} + \vec{f}_{Umgebung} \tag{3.16}$$

Da das Kräftesystem sehr stark ineinander greift, wird für jede der Teilkräfte der Gewichtete-Summe-Ansatz verwendet. Die Herausforderung mit dem ϵ-Constraint ist hier, dass die einzelnen Kräfte zu stark ineinander greifen, um bei den Formeln gezielt eine Richtung zu löschen. Dadurch ist das präzise Setzen einer Grenze zu aufwändig und das Risiko von unvorhersehbarem und ungewolltem Verhalten viel zu hoch.

3.5. Implementierung

Im Folgenden wird kurz auf das Designsetup der Experimente eingegangen. Zusätzlich findet eine Erläuterung der genauen Implementierung der zwei Steering-Ansätze statt. Die genutzte Paparazzi-Version ist mit allen erstellten Skripten in GitHub[1] hinterlegt.

Setupdesign

Der Aufbau des Programmieranteils dieser Arbeit ist in Abbildung 3.6 dargestellt Innerhalb des Kreises wird gezeigt, welche Prozesse innerhalb von Paparazzi und seinen Akteuren stattfinden. Das Run-Script startet den gesamten Ablauf der Simulation beziehungsweise eines Experiments und beendet sie am Ende. Hier werden alle Parametereinstellungen vorgenommen und die Ergebnisse im jeweiligen Ordner gespeichert. Bevor ein Experiment gestartet wird, wird jedem Kopter eins der zwei Logik-Module (Context -oder Swarm-Steering) zugewiesen. In diesen findet die Berechnung der Flugrichtung statt, die in JSBSim simuliert wird. Zusätzlich werden hier die Ergebnis-Werte aufgenommen und als Datei abgespeichert.

Damit die Kopter untereinander wissen, welche Interest-Points angeflogen wurden, verteilt und verarbeitet das Communication-Script die geschickten Daten zwischen den Koptern und Paparazzi. Zusätzlich berechnet es am Start eines Experiments zufällige Interest-Points für die Kopter. Sind die Ergebnisse in

[1] `https://github.com/ovgu-FINken/paparazzi/tree/ContextSteering_Phil`

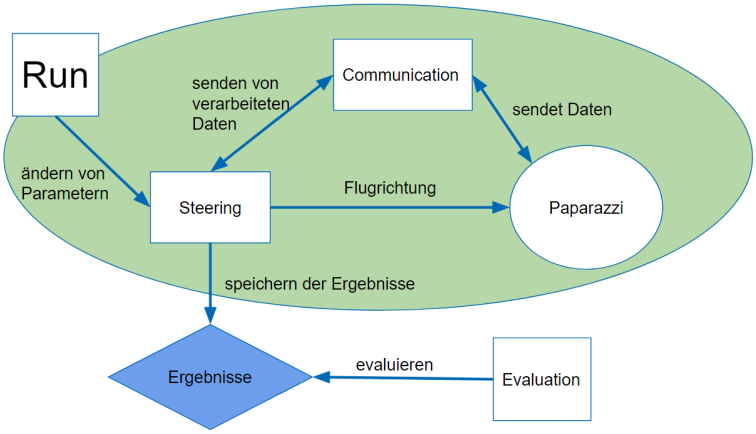


Abbildung 3.6.: Implementierungsdesign

den richtigen Ordnern abgespeichert und die Simulation beendet, kann das Evaluations-Script diese verarbeiten.

Implementierung der Logiken

Um die verschiedenen Context-Maps aus 3.3 zu implementieren, wurde zunächst eine Basis-Richtungs-Map aufgestellt, in der die einzelnen Richtungsvektoren enthalten sind. Daraufhin wurde jede Map als Funktion dargestellt, die jeweils durch die für sie relevanten Entitäten durchiteriert und vergleicht, ob das jeweilige Simulationsobjekt auf einem oder mehreren der Richtungsvektoren liegt. Der Einzugsbereich eines Richtungsvektors wird jeweils mit einem Abweichwinkel berechnet. Falls das Objekt innerhalb dieses Bereiches liegt, wird es bei den Interest/Danger-Maps auf den jeweiligen Slot der Context-Map addiert. Bei der Velocity-Map wird anstatt eines Objektes direkt der Geschwindigkeitsvektor betrachtet. Bevor die Context-Maps zu einer Result-Map miteinander verrechnet werden, werden die verschiedenen Entscheidungsfindungs-Maßnahmen aus Kapitel 2.4 angewendet. Die erste verwendete Maßnahme ist Gewichtete-Summe, bei der jede einzelne Context-Map ihr eigenes Gewicht bekommt. Der ϵ-Constraint wird für beide Danger-Maps benutzt. Hierbei wird ab einem zu großen Danger-Wert die Richtung bei der Entscheidung wohin geflogen werden soll blockiert. Die finale Richtung in die der Kopter fliegen soll wird aus der Result-Map ausgewählt.

Das Swarm-Steering wurde implementiert, indem die beiden Kräfte-Systeme Attraction/Repulsion und Umwelt unabhängig voneinander berechnet werden. Für das Attraction-Repulsion-System wurden dafür die Kräfte der Kopter untereinander berechnet, wie in Kapitel 2.1 beschrieben. Die Berechnung der Umweltkraft wird aus der Attraction/Repulsion-Berechnung abgeleitet. So werden in der Grundformel 2.1 anstatt der Drohnen-Positionen die Interestpoint-Positionen als Referenzpunkte genommen. Die dafür benötigten Danger -und Interestwerte werden mithilfe einer der Grundformeln aus Kapitel 2.1 berechnet. Die dadurch entstehende Umweltkraft wird mit der Attraction/Repulsion-Kraft in Formel 2.12 verrechnet. Das Ergebnis ist die finale Richtungskraft. Die beiden folgenden Pseudocode-Algorithmen 1 und 2 fassen dies noch einmal am Beispiel der Interest-Map-Berechnung und der Attraction-Repulsion-Kraft zusammen.

Algorithm 1 Berechnung der Interest-Map

Require: Q, p_0, M

$I = \vec{0}$
for q_j **in** Q **do**
    $\vec{r_j} = \vec{q_j} - \vec{p_0}$
    for m_i **in** M **do**
        $t_{r_j,m_i} = \frac{\vec{m_i} \cdot \vec{r_j}}{|\vec{m_i}| \cdot |\vec{r_j}|}$
        if $t_{r_j,m_i} \geq 0.7$ **then**
            $I_i = I_i + \frac{t_{r_j,m_i}}{|r_j|}$
        end if
    end for
end for
return I

Algorithm 2 Berechnung von der Attraction-Repulsion-Kraft

Require: P, k_a, k_r, c

$\vec{f}_{Arr/Rep} = \vec{0}$

for p_j **in** P **do**

  if $p_j \neq p_0$ **then**

    $\vec{f} = (p_j - p_0)\left(k_a - k_r \exp\left(\frac{-||p_j - p_0||^2}{c}\right)\right)$

    $\vec{f}_{Att/Rep} = \vec{f}_{Att/Rep} + \vec{f}$

  end if

end for

return $\vec{f}_{a/r}$

4. Experimente und Evaluation

In diesem Kapitel werden die durchgeführten Experimente erläutert. Das dafür notwendige Setup wird direkt nach dieser Einleitung in Abschnitt 4.1 erklärt. Aus dem Aufbau ergeben sich Hypothesen, die mit den Experimenten geklärt werden sollen, um die Forschungsfrage zu beantworten. Um einen Vergleich zwischen den einzelnen Steering-Ansätzen und den jeweiligen Szenarien zu schaffen, wurden drei Hypothesen aufgestellt.

1. Context-Steering erreicht mehr Durchläufe, bei denen alle IPs eingesammelt werden.
2. Sieben Bots sammeln die IPs am schnellsten ein.
3. Je weniger Bots existieren, um so stabiler ist das System bezüglich Entropie/Varianz und Kollision.

Im Anschluss an das Setup werden zwei Durchläufe 4.2 und 4.3 mit ihrer Vorbereitung, Durchführung und Auswertung ausgeführt. Das finale Ergebnis der Experimente wird in der Gesamtauswertung 4.4 zusammengefasst und die Hypothesen ausgewertet.

4.1. Aufbau und Szenarien

Im Folgenden wird der finale Aufbau der beiden Schwarmverhaltensmethoden mit allen getroffenen Entscheidungen erklärt. Anschließend wird festgelegt, wie die verschiedenen Szenarien aussehen sollen, die die Kopter in den Experimenten durchlaufen sollen.

Aufbau

Um die Experimente durchführen zu können müssen zunächst die einzelnen Parameter, die in den vergangenen Kapiteln besprochen wurden, sinnvoll gesetzt werden. Zusammengefasst sind das:

1. Wahl der Attraction-Repulsion -und Umgebungs-Funktionen
2. Berechnung für Context-Map-Werte
3. Berechnung der Result-Map
4. Auflösung Context-Map
5. Entscheidungsfindung
6. Anzahl der zu verwendenden Kopter
7. Daten/Aussehen der Simulationskarte
8. Startposition der Kopter/Interestpoints
9. Die in den Experimenten zu messenden Werte

Um das Swarm-Steering zu implementieren, wurde aus den im Grundlagenkapitel 2.1 erwähnten Formeln Formel 2.9 und 2.10 gewählt. Die daraus enstehende *Begrenzte Attraction und Repulsion*-Formel 4.1 bietet neben einer klar definierten Attraction den großen Vorteil, dass die Funktionsweise des "Abstoßen wenn nah dran"-Ansatzes enthalten ist.

$$f_{Attraction/Repulsion} = \sum_{j=1}^{d_{max}} (p_j - p_0) \left(a - b \exp\left(\frac{-||p_0 - p_j||^2}{c} \right) \right) \tag{4.1}$$

Somit soll gewährleistet werden, dass die Kopter immer in der gleichen Entfernung voneinander fliegen. Für die Berechnung des Umweltanteils wird eine Version des „Fast konstante Attraction und unbegrenzte Repulsion“-Ansatzes verwendet. Als Grundlage wird wieder die Basisformel 4.2 genutzt, wobei der Richtungsvektor aus der Position des berechnenden Kopters und der verschiedenen IPs I_j entsteht.

$$f_{Umwelt} = \sum_{j=1}^{q_{max}} (q_j - p_0)(f_{j,Interest} - f_{j,WallDanger}) \tag{4.2}$$

Für die Berechnung der Context-Maps werden für eine bessere Vergleichbarkeit die gleichen Formelansätze aus Kapitel 2.1 wie beim Swarm-Steering verwendet. Demnach werden für den für die Swarm-Maps "Begrenzte Attraction und Repulsion," und für die Umgebungs-Maps "Fast konstante Attraction und unbegrenzte Repulsion"" aus den Swarm-Bewegungsansätzen 2.1 benutzt. So können bereits bestehende Parameter wie a, b und c übernommen werden. Zusätzlich wird beim Context-Steering der Attraction/Interest-Wert immer mit

dem normierten Skalarprodukt 3.8 multipliziert, das aus dem Richtungsvektor der Context-Map und dem Richtungsvektor zum zu betrachtenden Punkt entsteht. Je größer der Winkel dieser beiden Vektoren ist, desto kleiner wird auch der resultierende Wert in der Context-Map. Nach der Berechnung der einzelnen Maps werden diese in einer Result-Map aus Formel 3.15 zusammengeführt, indem die einzelnen Werte addiert werden. So kann aus der Result-Map der höchste Wert als finale Richtung extrahiert werden.

Nach der Wahl der verschiedenen Berechnungsansätze musste noch festgelegt werde, wie groß die einzelenen Context-Maps werden sollen. Da die vorhandenen Realwelt-Kopter acht Sensoren enthalten, ist acht auch die gewählte Größe in 2D. Wenn, wie in Kapitel 3.1 besprochen, das 2D-Array auf 3D erweitert wird, indem die erklärten Formeln genutzt werden, entsteht ein 32 Stellen großes 3D Array, welches in Abbildung 4.1 dargestellt ist. Wie eine Result-Map nach der Aggregation aller Context-Maps aussehen kann ist in Abbildung 4.2 dargestellt.

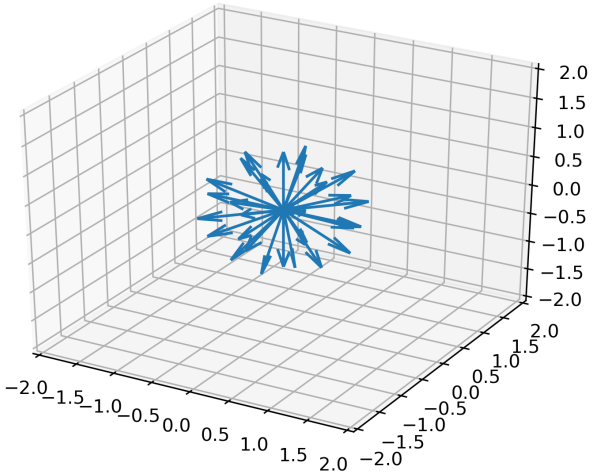


Abbildung 4.1.: Von 2D auf 3D umgewandelte Context-Map

Die implementierten Entscheidungsfindungs-Möglichkeiten für die Steering-Ansätze sind die aus Kapitel 3.4 besprochenen Gewichtete-Summe und ϵ-

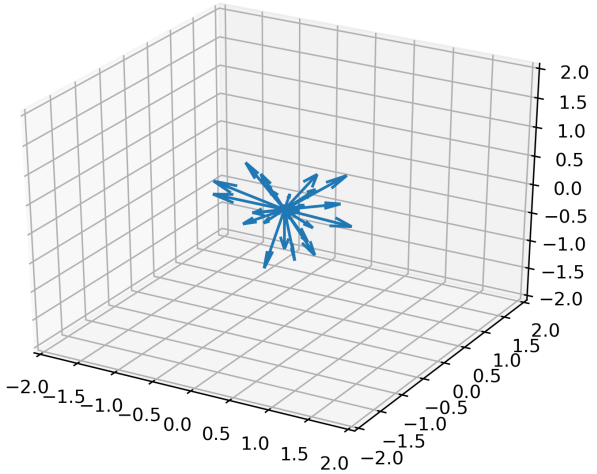


Abbildung 4.2.: Mögliches Aussehen einer Result-Map

Constraint. Bei der Gewichtete-Summe wird bei beiden Wegfindungs-Ansätzen bei der Berechnung des Endergebnisses (Result-Map oder Kraftvektor) jedem einfließenden Summand ein zusätzliches Gewicht α_i zugeordnet. Mit der Möglichkeit spezielles Verhalten hervorzuheben, entsteht für das Swarm-Steering die Endformel 4.3.

$$f_{Result} = \alpha_1 f_{Attraction/Repulsion} + \alpha_2 f_{Umwelt} \tag{4.3}$$

Beim Context-Steering werden die Danger-Maps mit einem ϵ-Constraint begrenzt. Befindet sich bei den Danger-Maps ein Wert über der gesetzten Grenze, wird die spezifische Richtung blockiert. In dem Sonderfall, dass alle Richtungen blockiert sind, verbleibt der Kopter solange an seiner Position, bis eine Richtung wieder frei wird. Die Funktionsweise des ϵ-Constraints kann auch in abgewandelter Form bei der Interest-Map verwendet werden. Dabei wird geprüft ob der berechnete Wert ein gesetztes Minimum unterschreitet und falls ja, wird er auf dieses festgelegte Minimum gesetzt.

$$I_i = \sum_{j=1}^{q_{max}} \max(\text{minInterest}, \frac{\vec{b}_i \cdot \vec{r}_{p_0 q_j}}{|\vec{r}_{p_0 q_j}|}) \tag{4.4}$$

Damit soll verhindert werden, dass IPs ab einer bestimmten Entfernung irrelevant für die Ergebniswahl werden. Die neue Form der Interest-Map-Berechnung

wird in Formel 4.4 dargestellt. Die Anwendung aller Entscheidungsfindungs-Algorithmen führt beim Context-Steering zu der Endformel 4.5.

$$R_i = \alpha_1 A_i + \alpha_2 I_i + \alpha_3 V_i - \alpha_4 D_i - \alpha_5 W_i \text{ solange } D_i, W_i < \text{Grenzwert} \quad (4.5)$$

Bei der Anzahl der fliegenden Kopter wurden jeweils die Grenzwerte des möglichen gewählt und der dazugehörige Mittelwert. Zwei Kopter sind der kleinste Schwarm, sieben sind die maximale Anzahl an Kopter die Paparazzi effektiv unterstützt und vier die Mitte zwischen beiden.

Die Karte, die für die Experimente implementiert wurde, ist ein Tunnel mit zwei gleich großen Räumen an beiden Enden, welcher in Abbildung 4.3 dargestellt wurde. Diese Art von Karte ermöglicht es, verschiedene Schwerpunkte des Schwarmverhaltens und der Wegfindung zu testen. So können IPs auf einem großen Raum in verschiedenen Quadranten verteilt werden. Den richtigen Weg zu finden, wenn die Punkte im Nachbarraum verteilt sind, kann je nach Tunneldurchmesser eine große Herausforderung darstellen. Beide genutzten Wegfindungsansätze haben keine Information darüber, wo sich der Tunneleingang befindet. Context-Steering kann allerdings die Wände und damit den Eingang mit der Walldanger-Map wahrnehmen. Swarm-Steering muss alleine aus den Danger und Interest-Werten einen geeigneten Weg durch den Tunnel finden. Dabei muss der Schwarm, um im Tunnel zu bleiben, seine Ausdehnung an die Breite des Tunnels anpassen. Zusätzlich können die verschieden wirkenden Kräfte im Swarm-Steering einen Deadlock auslösen, wenn sich der Schwarm zwischen zwei gleich weit entfernten IPs befindet. Je nach Tunnel-

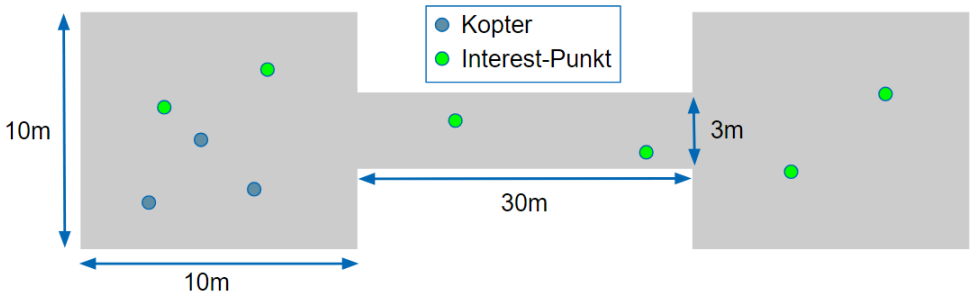


Abbildung 4.3.: Die für die Experimente verwendete Basis-Karte

durchmesser und Kartengröße können verschiedene Szenarien dargestellt und

getestet werden. Die Größe der Karte ist frei skalierbar mit einer Grundgröße von 50×10 m und einem 30 m langen Tunnel in der Mitte. Besagter Tunnel hat einen frei wählbaren Durchmesser. Die Startposition der Kopter befindet sich im linken Raum. Aus Gründen der Einfachheit wurden die Steering-Ansätze basierend auf dem East-North-Up (ENU) implementiert. ENU ein lokales System, bei dem jeder Kopter beim Start sich selbst im Koordinatenursprung sieht. Deshalb muss pro Kopter keine Koordinatensystemverschiebung stattfinden, wenn alle Kopter am gleichen Punkt starten. Dadurch kann gleichzeitig jeder den gleichen Flighplan benutzen. Zwar ist der Start in der gleichen Position innerhalb einer Realwelt-Anwendung nicht möglich, erleichtert aber in der Simulation an vielen Stellen die Berechnung. Direkt nach dem Start trennen sich die einzelnen Kopter durch die Steering-Ansätze und die Voraussetzungen der Simulation sind gleich der Realwelt.

Zusätzlich werden auf der Karte zufällig IPs verteilt. Damit diese sich pro Experiment nicht verändern, wurden sie mithilfe eines wählbaren Seeds generiert. Um zu testen, ob Context-Steering erfolgreich in 3D übertragen wurde, werden die IPs in verschiedenen Höhen generiert. Der Kopter hat einen IP-Sammelradius von 0.5 m. Dadurch muss der Schwarm seine Höhe an die Punkte anpassen, um sie abzufliegen. Wenn eine nachvollziehbare Höhendifferenz basierend auf der Position der IPs nachgewiesen werden kann, wurde Context-Steering erfolgreich im 3-Dimensionalen-Raum angewendet. Mit diesen Punkten sind die Experimente durchführbar. Die Werte die während der Simulation aufgenommen werden, sind in Tabelle 4.1 zusammengefasst.

Kopter-Unfälle werden hier nur aufgenommen, wenn Kopter untereinander zusammenstoßen würden. Für die Interaktion mit den Wänden werden die Out-Of-Map Werte aufgenommen. Out-Of-Map-Counter bezieht sich dabei auf die Anzahl, wie oft die Karte verlassen wird und Out-Of-Map-Percentage wie lange ein Kopter pro Experiment die Karte verlässt. Diese drei Parameter sind Teil der Robustheits-Auswertung. Mit diesem Stand muss noch der Rahmen der Experimente und deren Szenarios festgelegt werden.

Szenario

Innerhalb der Experimente werden neun verschiedene Kartenskalierungen durchlaufen, die als Basis Abbildung 4.3 benutzen. Wie genau diese Karten aussehen, ist in Tabelle 4.2 zu sehen. Dabei wird jede Kopter-Kombination

Parameter	Begründung
Position	Darstellung der Kopter-Position
ID	Differenzierung der gespeicherten Inhalte
Zeit	Notwendig für den Vergleich aller Experimente
Besuchte Interest-Punkte	Wie gut wurde das Experiment abgeschlossen
Out-of-Map-Counter	Wie Robust ist das System
Out-of-Map-Percentage	Wie Robust ist das System
Kopter-Unfälle	Wie Robust ist das System

Tabelle 4.1.: Zu speichernde Expererimentwerte

Index	Kartengröße in m^2	Tunnellänge in m	Tunneldurchmesser in m
1	25 × 5	15	1
2	25 × 5	15	3
3	25 × 5	15	5
4	50 × 10	30	1
5	50 × 10	30	3
6	50 × 10	30	5
7	100 × 20	60	1
8	100 × 20	60	3
9	100 × 20	60	5

Tabelle 4.2.: Mapgrößen pro Szenario

jedes Szenario durchlaufen. Jedes Szenario besitzt sechs IPs die zufällig verteilt werden. Es sind sechs, damit pro Kartenabschnitt zwei Punkte generiert werden können. Jedes Experiment hat eine Dauer von zweieinhalb Minuten und wird zehn mal wiederholt. Für die Auswertung der Szenarien wird eine Szenario-ID SX.Y.Z festgelegt. Das X steht dabei für den Index des Szenarios, Y für die Anzahl an geflogenen Quadcoptern und Z für die ID des Experiments. S3 steht damit für das dritte Szenario, S3.7 für das dritte Szenario bei dem sieben Kopter geflogen sind und S3.7.2 für das zweite Experiment im dritten Szenario mit sieben Koptern.

4.2. Erster Durchlauf

4.2.1. Vorbereitung

Um die Experimente durchführen zu können, müssen die bereits besprochenen Parameter festgelegt werden, die innerhalb des Aufbaus noch nicht genau spezifiziert wurden. Die folgende Tabelle 4.3 bildet diese mit ihren zugewiesenen Werten ab.

Die in den Attraction/Repulsion-Formel 4.1 eingesetzten a, b und c Parameter werden in Abbildung 4.4 dargestellt. Durch diese Belegung versucht der Kopter einen Abstand von ungefähr 1 m zu seinen Nachbarn zu halten. Das ist ein sicherer Abstand für die teilweise träge Koptersteuerung der Simulation. Da die Kopter innerhalb der Simulation keinen Körper haben, wurde ein Durchmesser von $0,2$ m festgelegt. Ein Unfall wird aufgenommen, wenn die Entfernung der Kopter zueinander kleiner als die **Unfall-Distanz** ist, die aus dem doppelten Durchmesser der Drohnen berechnet wurde. Eine Abweichung von den Realwelt-Drohnen ist möglich, da sich über die Zeit deren Durchmesser immer wieder verändert und der genaue Durchmesser für die Simulation keine Rolle spielt solange keine Unfälle passieren. Die einzelnen **Gewichte** sind durch erstes Austesten entstanden, wobei zuerst passende Gewichte für das Swarm-Steering gesucht wurden. Normalerweise ist das Ergebnis der Addition aller Gewichteten-Summen-Teile 1. Durch einen großen Werte-Unterschied der einzelnen Bestandteile der Ergebnis-Kraft und dem Hauptziel, dass die Kopter alle Interest-Punkte abfliegen sollen, muss das **Interesse** wesentlich höher eingestuft werden. Falls dieser Parameter zu niedrig war, haben sich die Kopter nicht zu den weiter entfernten Zielen bewegt. Dies ist vor allem

Parameter	Gesetzter Wert
a	1
b	10
c	$0,5$
Unfall-Distanz	$0,2$
Swarm-Steering	
$w_{Interest}$	4
$w_{DangerWall}$	2
$w_{AttRep-Force}$	$0,5$
$w_{Environment-Force}$	1
Context-Steering	
$w_{Interest}$	4
$w_{Attraction}$	$0,5$
$w_{Velocity}$	$0,2$
$w_{DangerDrones}$	$0,5$
$w_{DangerWall}$	2
$\epsilon - Danger$	5
$\epsilon - Wall$	7
$Minimal - Interest$	$0,05$

Tabelle 4.3.: Parameter-Konfiguration

bei höheren Bot-Mengen aufgetreten, da dort die schwarmeignenen Kräfte viel größer waren. Im Gegensatz führte zu hohes Interest dazu, das die **Wände** ignoriert wurden, was durch ein relativ hohes Gewicht an der WallDanger gelöst wurde. Trotz der hohen Gewichte bei den **Umwelt-Komponenten** war das Attraction-Repulsion-System mit seinen Kräften so dominant, dass es immer die Umwelteinflüsse überskaliert hat. Deshalb wurde ein niedriges Gewicht festgelegt und somit der Abstand verschoben, den die Kopter voneinander haben können, bis jeweils Attraction oder Repulsion die Kräfte der Umwelt übersteigen.

Um passende Parameter für Context-Steering zu finden, wurden die bereits getesteten Gewichte aus dem Swarm-Steering kopiert. So mussten nur noch das Gewicht für die aktuelle Geschwindigkeit und die jeweiligen ϵ**-Grenzen** gefunden werden. Nach einigen Tests, bei denen die Geschwindigkeit genauer untersucht wurde, ist ein vergleichsweise sehr niedriges Gewicht festgelegt worden. Sonst hätte es passieren können, die Velocity-Map einen zu starken

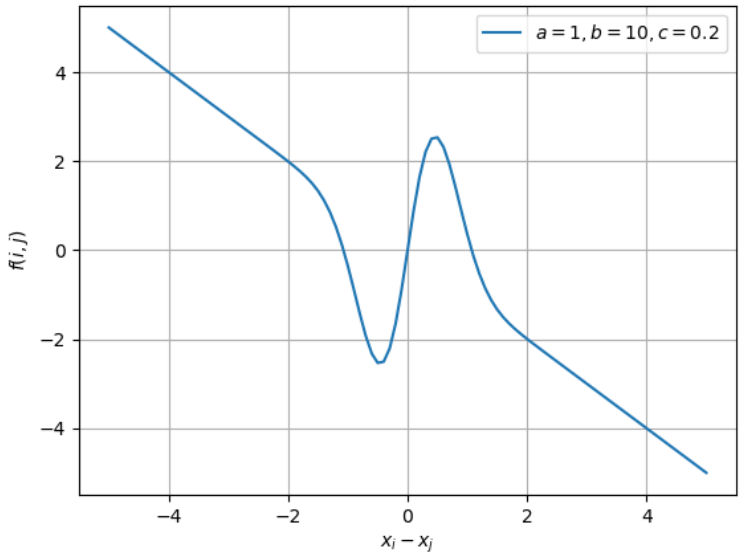


Abbildung 4.4.: Die verwendeten Attraction-Repulsion-Funktion

Einfluss auf das Gesammtergebnis hat und die Drohne nur noch in die gleiche Richtung fliegt. Für die ϵ-Grenzen wurden die Werte teilweise berechnet indem analysiert wurde, inwiefern ein logisches Verhalten erzeugt wird. Für die **Danger-Drone-Grenze** welche den Abstand der Drohnen einbezieht wurde ein Wert von 5 gesetzt. Dieser Wert blockiert die Richtung, falls eine benachbarte Drohne näher als $0,6\,\text{m}$ fliegt. Die Grenze für die **Wall-Danger** liegt mit 6 so, dass der Kopter nicht näher als circa $0,4\,\text{m}$ an eine Wand fliegen soll.

Der Abstand zur Wand kann kleiner sein als zu benachbarten Drohnen, da die Wand kein dynamisches Objekt innerhalb der Echtzeit-Berechnung ist. Zusätzlich dürfte der Abstand nicht größer als $0,5\,\text{m}$ werden, da dies der Minimalabstand der generierten IPs von den Wänden ist. Als Letztes wurde der **Minimal-Interest** gesetzt auf $0,05$. Bei einem Abstand von mehr als $20\,\text{m}$ wir der Interest-Map dieser Minimalwert zugewiesen. Es wurde ein Wert gesucht der so niedrig ist, dass Punkte in der Nähe einen wesentlich höheren Wert haben. Die nahen Punkte sind meistens die, die am Anfang direkt im gleichen Raum generiert worden. Bei den ersten Tests wurde eine Schwelle von $0,1$ verwendet, was dazu führte, dass die Kopter teilweise direkt in den Tunnel flogen, da das Interesse in Richtung rechtem Raum viel höher war. In dieser Richtung befanden sich vier Interest-Punkte, die mit ihrem Abstand ein

Mindest-Interesse von $0,4$ erzeugten. Der im gleichen Raum wie die Drohne existierende Punkt hatte einen Abstand von 5 m, wodurch ein Interesse von $0,2$ entstand. Das war wesentlich kleiner war als das Interesse in Richtung Tunnel. Deshalb wurde das Minimal-Interesse verkleinert, das zur gewünschten Handlungsdirektive bei den Kopter führte. Die Festlegung dieser Werte führte zu den verschiedensten Funktionen, die in Abbildung 4.5 inklusive ihrer Gewichte dargestellt sind.

Zusätzlich ist zu erwähnen, dass die **Experimente zur Festlegung dieser Parameter** auf der kleinsten Map (S1) durchgeführt wurden. Es bestand die Annahme, dass der Schwarm, wenn er S1 erfolgreich durchläuft, auch die größeren "einfacheren" Karten erfolgreich abschließt. Diese Gewichte ändern die

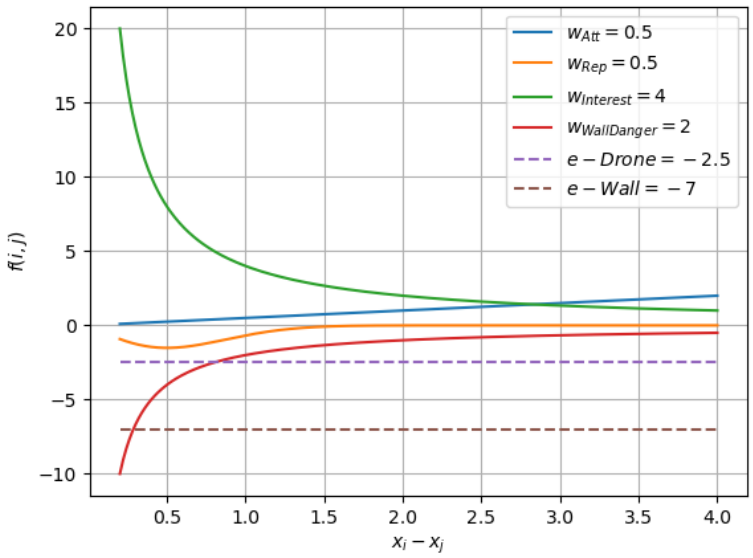


Abbildung 4.5.: Die aus den Gewichten entstehenden Funktionen

genutzten Funktionen wie in Abbildung 4.5 dargestellt. Negative Werte stehen dabei für eine abstoßende Kraft. Dadurch entstand innerhalb der ersten kleinen Tests für beide Ansätze ein Parametersatz, der sich in vielen Werten überschnitt und sehr robust wirkte, was in den folgenden Experimenten validiert werden soll.

4.2.2. Durchführung

Die nachfolgenden Experimente wurden auf einem eigenen PC durchgeführt. Sie wurden innerhalb einer virtuellen Maschine auf einem Ubuntu-Betriebssystem durchgeführt. Der PC besitzt die in Tabelle 4.4 aufgeführte Ausstattung. Die bei den Experimenten berechneten IP-Positionen wurden in Tabelle A.1 im Anhang zusammengefasst.

Bauteil	Typ
CPU	Intel(R) i5-6500 @ 3,2 GHz
GPU	Intel HD Graphics 530
RAM	24 GB @ 2133 MHz
Betriebssystem	Ubuntu 2004

Tabelle 4.4.: PC-Spezifikation

4.2.3. Auswertung

Um die Auswertung sinnvoll aufzubauen, wird jedes Szenario einzeln betrachtet. Dabei wird zunächst auf die generelle Performance der Ansätze in Hinblick auf ihren gesammelten Interest eingegangen. Innerhalb der Szenarien wird der Einfluss der Anzahl von Quadcoptern auf die jeweilige Kartengröße heraus gearbeitet und eine Begründung für die Unterschiede zwischen den beiden Algorithmen gesucht.

Die beiden Tabellen 4.5 und 4.6 dienen dabei als erste Grundauswertung. Hier ist ein grober Unterschied zwischen verschiedenen Szenarien und Schwarmverhalten leicht abzulesen. Dazu kommen verschiedene Box-Whisker-Plots, die im Anhang abgebildet sind. Diese haben verschiedene Metriken, die pro Szenario gemessen wurden. Abbildung A.2 stellt die gesammelte Interest-Menge pro Zeitslot dar. In diesen Abbildungen wurde die Experiment-Dauer in vier gleich große Zeitslots eingeteilt, aus denen die Menge der abgeflogenen IPs veranschaulicht wird. Die bereits angesprochene Simulations-Message-Verzögerung tritt hier am stärksten auf. Wenn zwei Kopter einen Punkt fast zeitgleich erreichen, der Punkt aber nur noch einmal erreicht werden muss, nimmt das Ergebnis trotzdem beide Besuche auf. Die Verzögerung entsteht dadurch, dass wie in Kapitel 3.2 besprochen, die Nachrichtenverteilung zwischen den einzelen Schwarmmitgliedern sehr lange dauern kann. Sobald ein IP abgeflogen

Szenario-Index	maximal erreichte Interest-Punkte	Ergebnis
		Zwei Kopter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	6	Drei Experimente haben alle Punkte erreicht
5	6	Alle Experimente haben alle Punkte erreicht
6	6	Alle Experimente haben alle Punkte erreicht
7	4	Punkt I-0 und I-4 wurden nie erreicht
8	5	Punkt I-0 wurde nie erreicht
9	5	Punkt I-0 wurde nie erreicht
		Vier Kopter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	6	Drei Experimente haben alle Punkte erreicht
5	6	Alle Experimente haben alle Punkte erreicht
6	6	Alle Experimente haben alle Punkte erreicht
7	4	Punkt I-0 und I-4 wurden nie erreicht
8	5	Punkt I-0 wurde nie erreicht
9	5	Punkt I-0 wurde nie erreicht
		Sieben Kopter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	6	Nur ein Experiment erreicht alle Punkte
5	5	Punkt I-0 wurde nie erreicht
6	5	Punkt I-0 wurde nie erreicht
7	3	Punkte I-0, I-4 und I-5 wurden nie erreicht
8	4	Punkte I-0 und I-5 wurden nie erreicht
9	5	Punkt I-0 wurde nie erreicht

Tabelle 4.5.: Context-Steering-Ergebnisse für den ersten Durchlauf

Szenario-Index	maximal erreichte Interest-Punkte	Ergebnis
		Zwei Kopter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	6	I-0 und I-1 sind die einzigen erreichten Punkte
5	5	Experimente mit hohen Punkten erreichen I-0 nicht
6	5	Punkt I-0 wurde nie erreicht, sonst immer alles
7	1	Jedes Experiment hat nur I-1 erreicht
8	3	Alle Experimente haben I-1, I-2 und I-3 erreicht
9	3	Alle Experimente haben I-1, I-2 und I-3 erreicht
		Vier Kopter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	2	I-0 und I-1 sind die einzigen erreichten Punkte
5	5	Experimente mit hohen Punkten erreichen I-0 nicht
6	5	Punkt I-0 wurde nie erreicht, sonst immer alles
7	1	Jedes Experiment hat nur I-1 erreicht
8	3	Alle Experimente haben I-1, I-2 und I-3 erreicht
9	3	Alle Experimente haben I-1, I-2 und I-3 erreicht
		Sieben Kopter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	2	I-0 und I-1 sind die einzigen erreichten Punkte
5	2	I-0 und I-1 sind die einzigen erreichten Punkte
6	5	Punkt I-0 wurde nie erreicht, sonst immer alles
7	1	Jedes Experiment hat nur I-1 erreicht
8	2	Alle Experimente haben I-1 und I-2 erreicht
9	2	Alle Experimente haben I-1 und I-2 erreicht

Tabelle 4.6.: Swarm-Steering-Ergebnisse für den ersten Durchlauf

wird, speichert der jeweilige Kopter das für sich ab und schickt dieses Array an erreichten Punkten an den Server. Dieser verteilt die Nachricht an das Kommunikations-Script, welches die Auswertung aller Kopter-Interest-Arrays durchführt. Die ausgewertete Liste mit allen erreichten IPs wird wieder auf alle Kopter verteilt. Diese Verteilung dauert bis zu sechs Sekunden, wodurch die Kopter teilweise den Punkt häufiger erreichen als notwendig. Dadurch entsteht innerhalb dieser Plots teilweise eine höhere Sammelrate pro Zeitslot als vorgesehen. Innerhalb der Simulation kann es deshalb zu kleinen Zeitverzögerungen kommen, da die Kopter noch versuchen, den IP zu erreichen, obwohl er bereits ausreichend angeflogen wurde. Bis auf diesen maximal sechs Sekunden großen Zeitverzug hat die Nachrichtenverzögerung keinen Einfluss auf die Ergebnisse, da der Schwarm eine sehr kompakte Struktur aufweist. Sobald ein IP besucht wird, ist er auch immer in ausreichender Zahl erreicht worden.

In Boxplot A.3 wurde extrahiert, zu welchem Zeitpunkt im Experiment der letzte IP eingesammelt wurde. Es wird allerdings nicht dargestellt, ob dies auch der Letzte ist, der noch fehlt, sondern lediglich der Letzte der angeflogen wurde. Die beiden Abbildungen A.4 und A.5 befassen sich mit der Robustheit der Experimente. Wichtig ist, dass die Kopter in allen Experimenten keinen einzigen Unfall untereinander hatten, weshalb die Koper-Unfall-Metrik nicht weiter analysiert wird.

Für die Stabilität des Schwarms werden für verschiedene auffällige Experimente die Varianz -und Entropie-Graphen über die Zeit dargestellt. Diese erklären, wie flächenmäßig verteilt der Schwarm war und wie konstant seine Verteilung war. Da die Kopter alle am selben Punkt starten sind bei null Sekunden in diesen Graphen auch Extrem-Werte abgebildet, die ignoriert werden können, da es sich um erwartete Anomalien handelt. Falls weitere sichtbaren Besonderheiten innerhalb der Experimente auffallen, wurden zusätzlich Heatmaps für die Szenarien aufgestellt, anhand derer man sowohl den Flugverlauf als auch den kleinsten Abstand des Schwarms untereinander ablesen kann. Im Anhang sind unter Abbildung A.1 die leeren Heatmaps inklusive der Positionen der eingezeichneten IPs dargestellt. Dies ermöglicht weitere Aussagen über das Verhalten des Schwarms.

Szenario 1-3

Da Anhand dieser Szenarien die Gewichte festgelegt wurden, ist das Ergebnis sehr gut ausgefallen. Wie an den Tabellen zu erkennen ist, ist sowohl für Swarm -als auch für Context-Steering der maximale Interest-Wert innerhalb aller Bot-Mengen erreicht worden. Die großen Unterschiede liegen hier in der Zeit und in der Robustheit/Stabilität. Im Context-Steering unterscheiden sich die Ergebnisse mit verschiedenen Bot-Mengen nur darin, wie lange die Kopter außerhalb der Karte waren, was in Abbildung 4.6 sichtbar wird. Je mehr Bots

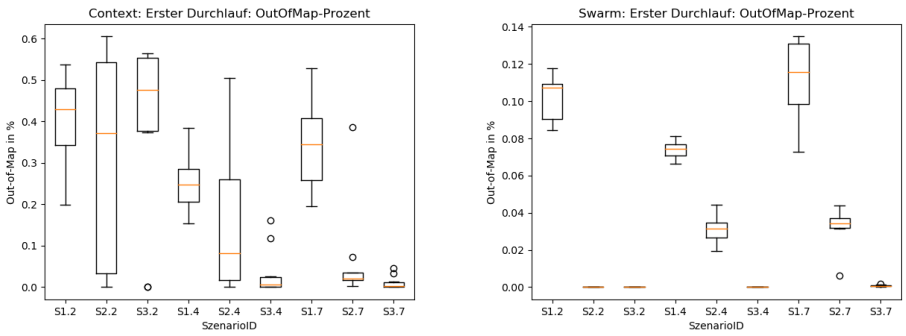


Abbildung 4.6.: Out-Of-Map Zeit (in %) pro Szenario in S1-3

geflogen sind, um so weniger befand sich der Schwarm außerhalb der Karte. Anhand von Abbildung 4.7 ist zu erkennen, dass Context-Steering um die $50 - 70\,\mathrm{s}$ gebraucht hat, wobei mehr Kopter das Experiment minimal langsamer bewältigt haben als weniger Kopter. Im Swarm-Steering brauchten die

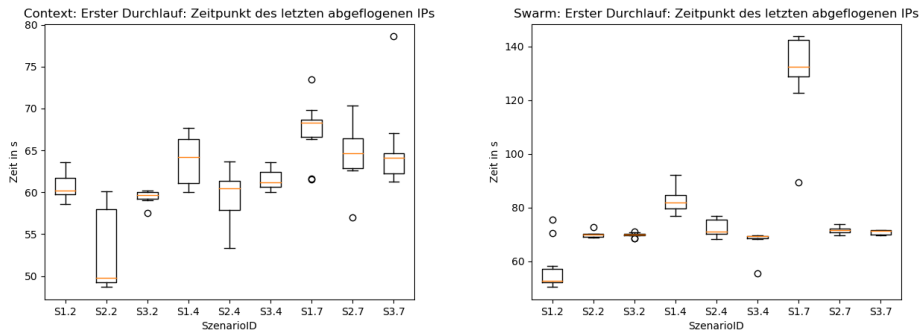


Abbildung 4.7.: Letzter Zeitpunkt der Interest-Erhöhung S1-3

Experimente allerdings im besten Fall genauso lange. Im schlechtesten Fall

benötigten sie mit mehr als 120 s bis zum Erreichen des letzten IPs in S1.7. In S1, dem Szenrio mit dem kleinsten Tunneldurchmesser sind die Experimentzeiten drastisch schlechter, wenn mehr Kopter benutzt werden. Dies liegt an der Funktionsweise des Swarm-Steerings. Je mehr Kopter vorhanden sind, desto größer ist die Kraft welche der Interessen-Verfolgung entgegen wirkt.

Zusätzlich ist zu erwähnen, dass teilweise die Schwärme in den Experimenten den IP I-0 zuletzt eingesammelt haben, weil sie erst durch den Tunnel geflogen und im rechten Raum die Punkte besucht haben, bevor sie zu I-0 in den Startraum zurückgekehrt sind. Dadurch ist die Experimentzeit länger, als wenn der Schwarm schon am Anfang I-0 eingesammelt hätte. Dieses Verhalten führt zu der ersten Vermutung, dass die Interest-Funktion weiter optimiert werden sollte, wenn nähere Punkte einen kleineren Interest aufweisen als mehrere Punkte mit einem größeren Abstand. Wie bereits erwähnt, ist der größte Unterschied in S1-3 die Robustheit und die Stabilität des Schwarms. Swarm-Steering weist in beiden Feldern wesentlich bessere Ergebnisse auf. Innerhalb der drei Szenarien wurde die Map maximal 13% der Zeit verlassen und auch weniger oft. Zusätzlich verlässt Swarm-Steering ab S3 die Karte praktisch nicht mehr. Im Weiteren wird der Vergleich der Entropie- und Varianzgraphen von jeweils einem Experiment in Abbildung 4.8 gezeigt. Die beiden Ansätze haben einen ähnlichen Graphverlauf in sowohl Entropie als auch Varianz. Das Tal in beiden Entropie-Graphen zeigt sehr gut das Schwarmverhalten am Tunneleingang bei etwa 18 s für Swarm-Steering und 20 s für Context-Steering. Die dafür notwendige Verteilung der Kopter erzeugt auch einen starken Anstieg der Varianz an der Stelle an dieser Stelle. Trotzdem ist klar ersichtlich, wie viel stabiler der Schwarm sich beim Swarm-Steering fortbewegt, da der Graph viel weniger oszilliert.

Das beim Context-Steering der Schwarm die Karte verlässt liegt vor allem daran, dass sobald alle IPs abgeflogen wurden, die Kopter sich ohne klares Ziel weiter bewegen. Der Fall, dass alle Punkte eingesammelt wurden, ist nicht extra behandelt wurden. Deshalb tritt in dieser Zeit undefiniertes Verhalten auf. Das ist in der Abbildung 4.9 von S2.2.0 sehr gut ersichtlich. Hier verlässt der Schwarm die Karte extrem in Richtung Süden, nachdem er alle IPs eingesammelt hat. Dieses undefinierte Verhalten muss innerhalb der Implementierung auftreten, da kein Interest in dieser Richtung aufgenommen wird und damit auch der Schwarm die Karte nicht verlassen sollte. Das führt generell dazu, dass die Kopter einen sehr hohen Out-Of-Map Anteil haben. Dies ist teilweise auch der Grund in den folgenden Szenarios, warum ein erheblicher Prozent-

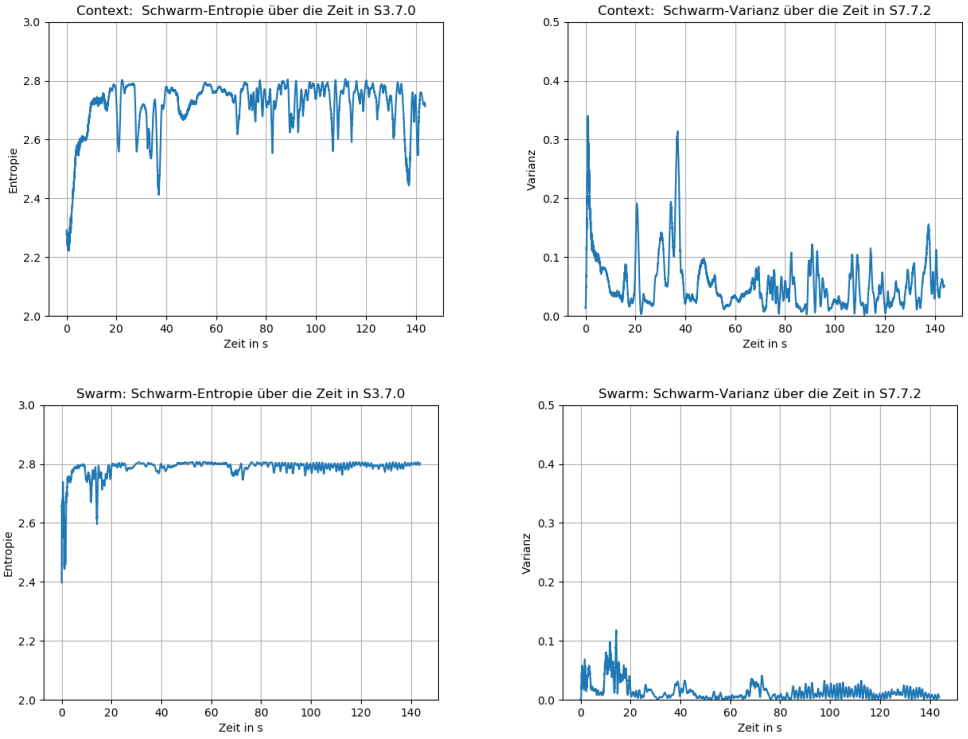


Abbildung 4.8.: Vergleich zwischen der Entropy/Varianz in beiden Ansätzen

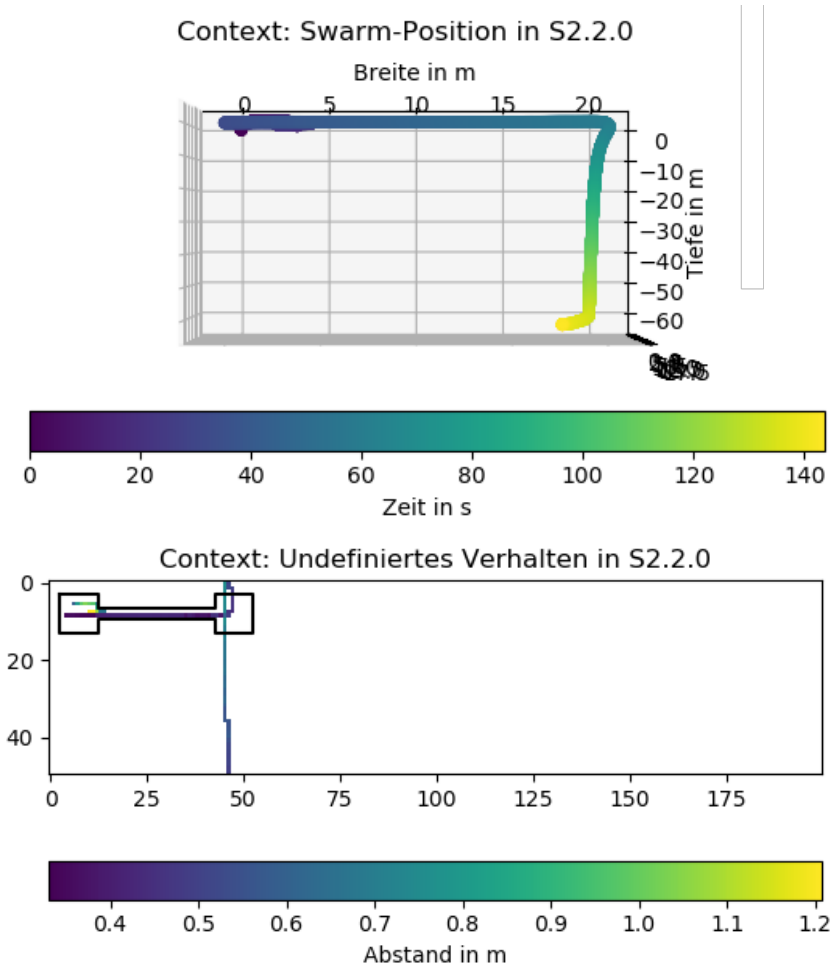


Abbildung 4.9.: Undefiniertes Verhalten eines Swarms

satz der Zeit außerhalb der Karte verbracht wird. Im Swarm-Steering tritt dieses Verhalten nicht auf, weshalb der Schwarm, wenn er die Map, verlässt nur sehr kurz außerhalb ist. Weiterhin unterscheiden sich die beiden Ansätze im Interest-Gain, welcher in Abbildung 4.10 dargestellt ist. Dabei hat Swarm-Steering bei größeren Botmengen ein kleineres Maximum, was auch die längere Experimentzeit erklärt, bis alle IPs erreicht wurden. Beide Ansätze haben sehr viele Zeitslots, in denen der Interest-Gain null ist. Dies liegt daran, dass die ersten drei Szenarios sehr schnell mit dem Abfliegen aller IPs fertig waren. Dadurch entstanden in der Restzeit viele Zeitslots mit null Interest-Gain, die ignoriert werden können. Eine der Teilfragen dieser Arbeit, ob Context-Steering in 3D übertragbar ist konnte hier bereits beantwortet werden. Da

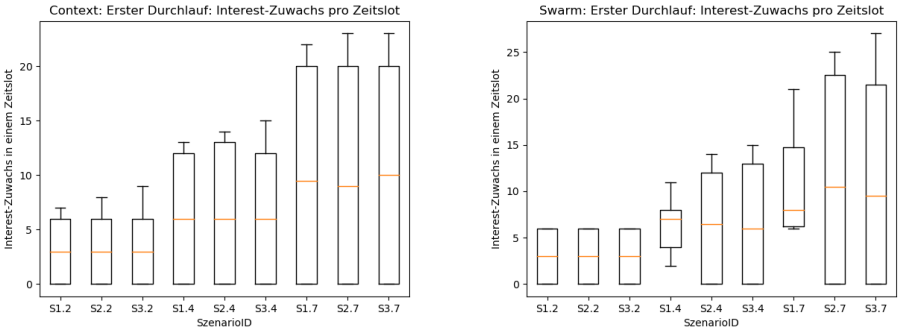


Abbildung 4.10.: Interest-Gain pro Zeitslot in S1.7

S2.2.0 bereits in Abbildung 4.9 betrachtet wurde, eignete sich das Experiment für eine Höhenuntersuchung. Anhand von Abbildung 4.11 ist eine Anpassung der Flughöhe erkennbar. Diese Veränderung basiert auf der Grundlage der je-

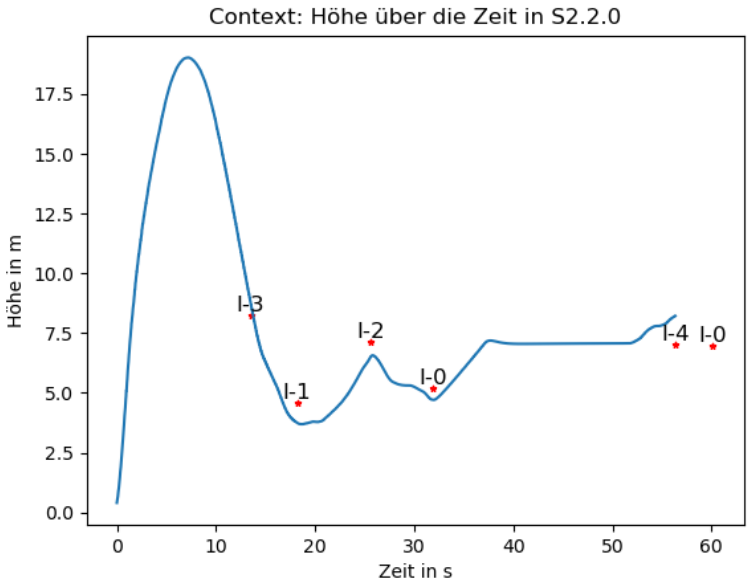


Abbildung 4.11.: Flughöhe eines Kopters

weiligen Positionen der Interestpoints und realisiert damit das 3D-Verhalten des Schwarms. In diesem Beispiel sammelt der Schwarm die Punkt in der Reihenfolge I-3 (7.14 m), I-1 (5.2 m, I-2 (8.24 m), I-0 (4.58 m, I-4 (7.02 m) und I-5 (6.96 m) ein.

Szenario 4

In S4 sind die ersten großen Unterschiede zwischen Schwarm-und Context-Steering zu erkennen, die nicht Teil der Stabilität und Robustheit waren. Während Context-Steering mindestens einen vollständigen Durchlauf basierend auf den abzufliegenden Punkten hatte, erreichten alle drei Kopter-Mengen beim Swarm-Steering nur die ersten beiden IPs I-0 und I-1. Der Grund warum Context-Steering nicht alle Experimente erfolgreich absolviert hat ist, dass häufig IP I-0 nicht abgeflogen wurde und der Schwarm am Ende keine Zeit hatte, zurück zu fliegen.

Das Verhalten vom Swarm-Steering, dass nur I-0 und I-1 abgeflogen wurden, ist darauf zurück zu führen, dass sich die Kartengröße im Vergleich zu S1-3 verdoppelt hat (Vergleich Tabelle 4.2). Wenn der Vergleich mit S1 gezogen wird, liegt der Kopter bei I-0 in einem schlechteren Winkel für den Tunnel. Somit tritt eines der großen in Kapitel 3.1 besprochenen Probleme mit Swarm-Steering auf. Der Kopter kann keinen direkten Weg zu den IPs im Tunnel oder dahinter finden, der nicht von einer Wand blockiert wird. Da diese wesentlich näher am Kopter liegt übt sie eine stärkere Abstoßungskraft aus als die Anziehungskraft die von den IPs generiert wird. Dadurch, dass das Swarm-Steering keine Informationen über seine eigene Umwelt erhält und sie nicht wahrnehmen kann, weiß der Schwarm nichts von der Existenz des Tunnels und verbleibt im linken Raum der Karte, was an der dritten Heatmap 4.16 zu erkennen ist. Dadurch ist in S4 auch kein großer Interest-Gain aus Abbildung 4.12 erkennbar. Ab-

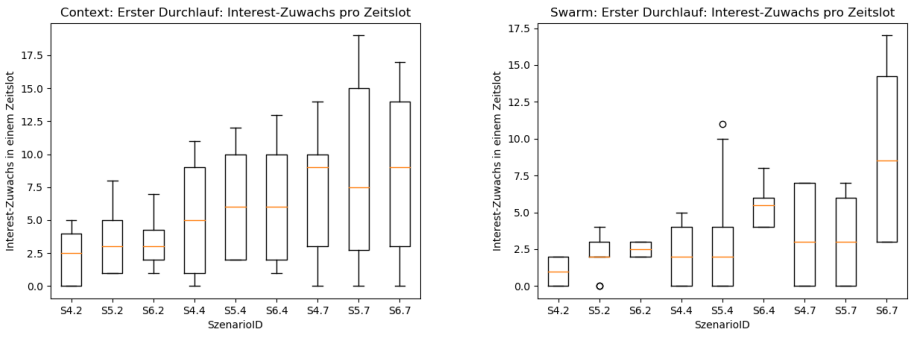


Abbildung 4.12.: Interest-Gain pro Zeitslot in S4-6

bildung 4.13 zeigt, dass das I-0 und I-1 bereits nach etwa 30 s abgeflogen sind und nicht mehr Interest generiert wurde.

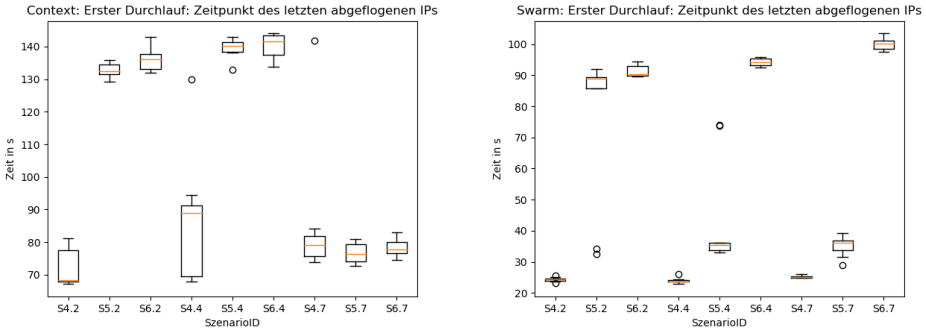


Abbildung 4.13.: Letzter Zeitpunkt der Interest-Erhöhung S4-6

Context-Steering kann dieses Problem lösen, da es durch die verschiedenen in den Maps enthaltenen Richtungsvektoren seine Umwelt wahrnehmen kann. Deshalb existieren zwei mal drei vollständige Durchläufe bei zwei und vier Koptern und einer bei sieben fliegenden Drohnen. Hierzu wird die Stabilität betrachtet, indem ein Vergleich zwischen einem Experiment mit vollem Interest (S4.7.7) und einem mit unvollständigem Interest (S.4.7.0) afugestellt wird. Die Varianz- und Entropiegraphen sind in Abbildung 4.14 dargestellt und zeigen zwei Unterschiede. Der Entropyabfall und Varianzanstieg am Ende basiert auf dem undefinierten Verhalten was bereits in S2.2.0 in Abbilding 4.9 erklärt. Da S4.7.0 nicht alle IPs abgeflogen hat ist hier dieses Verhalten nicht vorgekommen. Ein weiterer Unterschied befindet sich in der Mitte der Graphen zwischen 30 und 70 m. Da der Schwarm von S4.7.7 ein viel stärkere Varianz aufweist muss er hier viel stärker verteilt sein, was darauf schließen lässt das die Map weniger verlassen, da er sich dem Tunnel viel stärker anpasst.

Bei vielen Experimenten fehlt nur I-0 für einen vollständigen Durchlauf. Wie in S1-3 beschrieben, fliegt der Kopter erst in den Tunnel. Der Interest von einem einzelnen IP, der 30 bis 40 m entfernt liegt, ist aber nicht ausreichend, damit der Schwarm zurück fliegt. Zusätzlich tritt das Problem auf, dass der Schwarm, welcher sich im rechten Raum aufhält, keinen sicheren Zugang zum Tunnel findet. Der einzige relevante Zielpunkt ist zum Zeitpunkt I-0 und dazwischen liegt eine Wand. Auf dem Hinweg zeigen IPs innerhalb des Tunnels, auf welcher Position sich der Eingang befindet. Es kann also sein, dass sobald kein IP im Tunnel generiert werden würde, Context-Steering diesen auch nur betritt, wenn das Interest die Walldanger übersteigt und er somit die Map verlässt. Wie auch in den ersten drei Szenarien ist der Ursprung für das Verhalten wieder

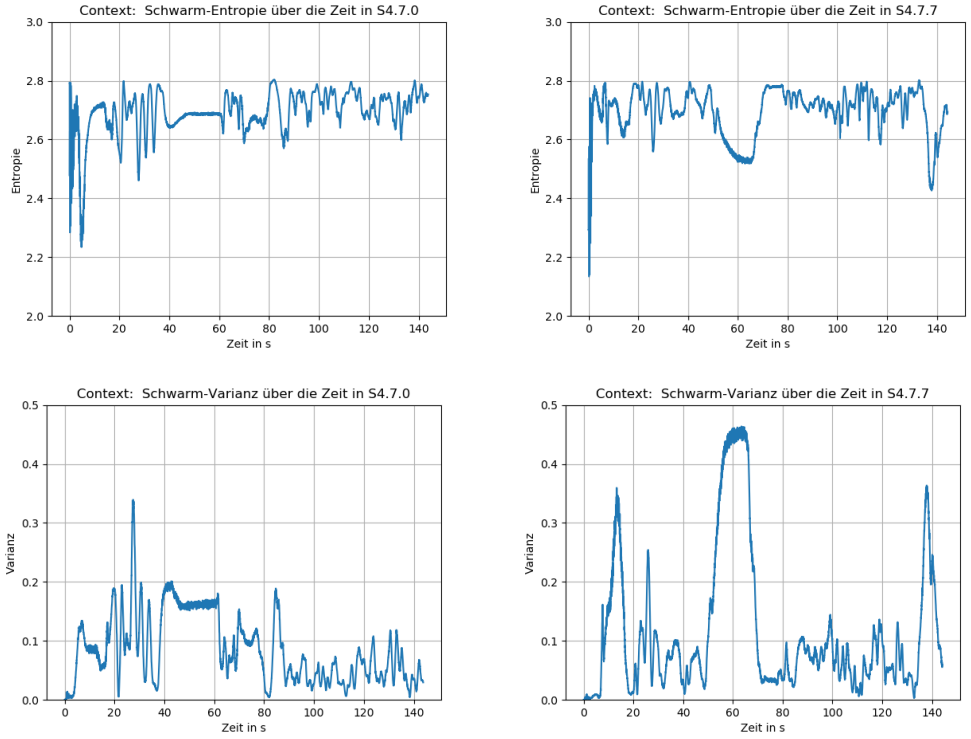


Abbildung 4.14.: Vergleich zwischen der Entropy/Varianz in S4.7.0 und S4.7.7

die ungeeignete Interest-Funktion und eine zu niedrige Walldanger, wenn der Schwarm zu häufig die Karte verlässt.

Wenn in Betracht gezogen wird, dass die Kopter immer I-0 sofort mit abfliegen, wie es einige Experimente geschafft haben, ist die Experimentzeit nicht wesentlich höher als bei den drei Vorgängern, bei denen die Karte nur halb so groß war. Im Gegensatz zu Swarm-Steering ist beim Context-Steering stark auffällig, dass in diesem Szenario die Out-Of-Map-Counter aus den Boxplots 4.15 stark erhöht sind. Anhand der ersten Heatmap 4.16 ist erkennbar, dass die-

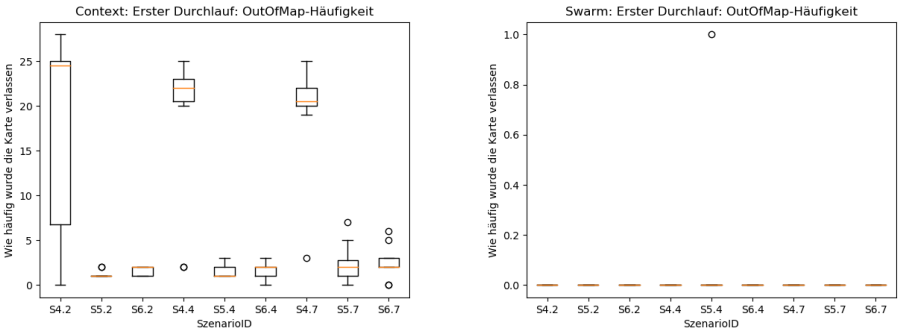


Abbildung 4.15.: Out-Of-Map-Häufigkeit pro Szenario in S4-6

se hohen Zahlen zustande kommen, indem die Kopter beim durchqueren des Tunnels sehr häufig seine Grenzen überfliegt. Bei der zweiten Heatmap 4.16 mit sieben Kopter fällt das weniger auf. Da aber fast doppelt so viele Drohnen fliegen und der Minimalabstand fast identisch ist, müssen mehr Bots außerhalb der Map fliegen. Dadurch, dass der Swarm-Steering Ansatz den Tunnel nie anfliegt, kann er die Karte auch nicht verlassen.

Szenario 5/6

Da in diesen Szenarios der Tunnel vergrößert wurde, funktionieren hier die Ansätze wieder wesentlich besser als in S4. Context-Steering hat in beiden Fällen eine Erfolgsquote von (100 %). Allerdings besteht auch wie in den ersten Szenarien das Problem, dass teilweise I-0 erst am Ende abgeflogen wird. An der ersten Heatmap 4.19 und der Abbildung 4.17 ist sehr gut zu erkennen, dass diese Rückwege sehr häufig außerhalb der Map liegen, was zu einer hohen Out-Of-Map-Quote führt. Zusätzlich ist anhand der Heatmap zu erkennen, dass die Dichte des Schwarms sehr stark zunimmt, wenn er auf die Wand

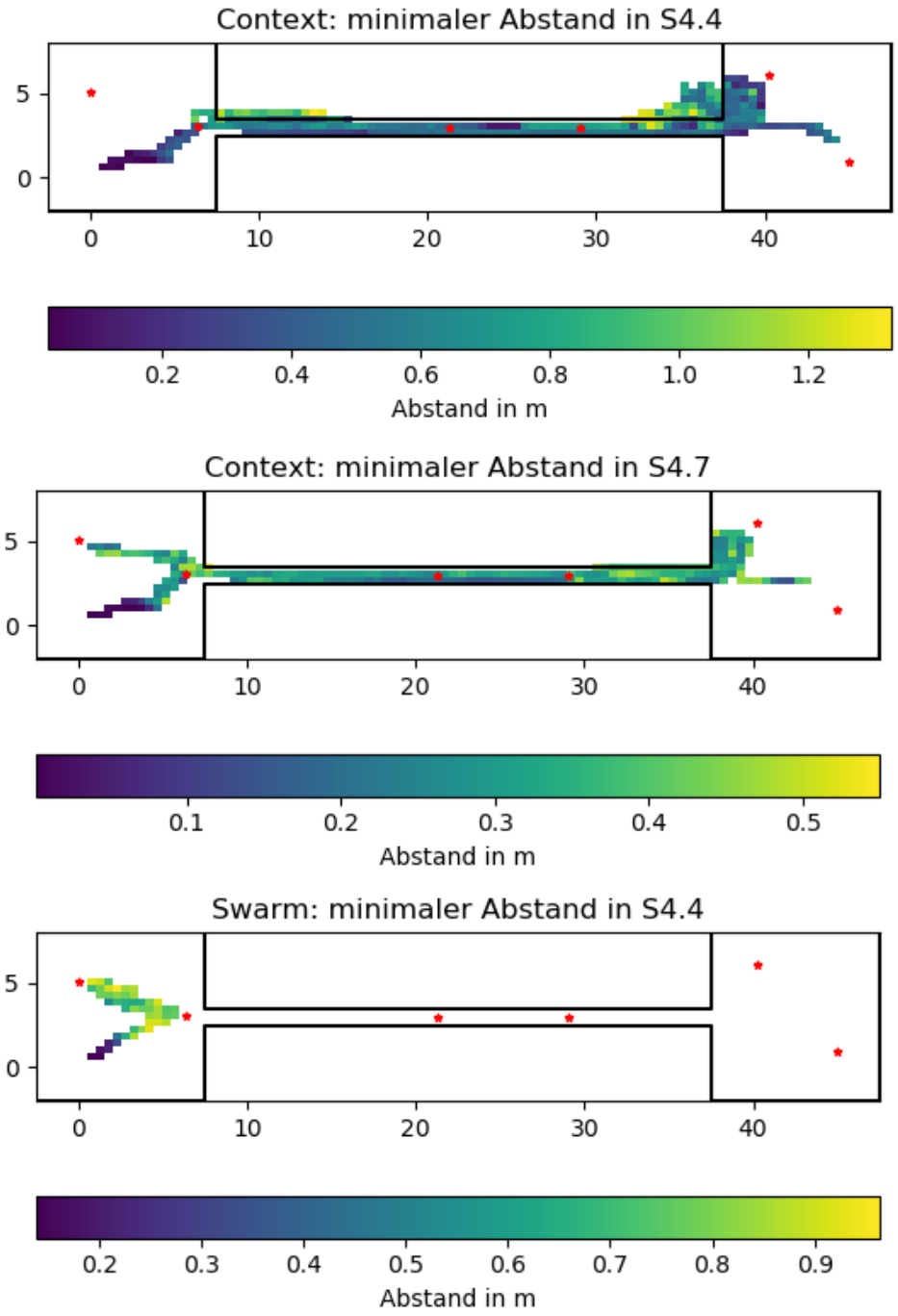


Abbildung 4.16.: Verschiedene Heatmaps für Szenario 4

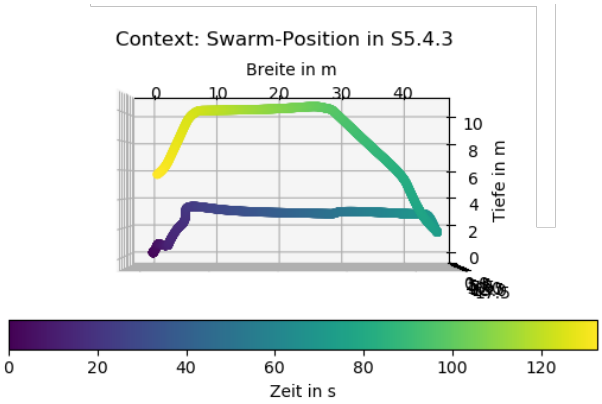


Abbildung 4.17.: Schwarmposition in S5.4.3

zufliegt, wodurch die Abstoßungskräfte der Kopter untereinander zunehmen. Diese werden irgendwann so groß, dass sie die Wand-Danger übersteigen und sich die Kopter durch die Wand durchdrücken. Wenn die Map einmal verlassen wurde und keine Wall-Danger mehr wirkt, kann ein beliebiger Weg in Richtung I-0 gefunden werden, der nur noch von der Velocity-Map und den eigenen Schwarm-Maps abhängig ist. Eine Lösung für das Problem ist, die Stärke der Walldanger weiter zu erhöhen, sodass die Interest-Kraft die Walldanger nicht überschreiten kann. Zusätzlich kann der Abstand, der vom $\epsilon_{DangerWall}$ abhängt, vergrößert werden ab dem die Richtung blockiert.
In Swarm-Steering tritt ein anderes Problem auf, welches einen großen Einfluss auf Context-Steering in S4 hatte. Sobald der Schwarm den Tunnel angeflog, wurde I-0 nie erreicht. Wenn aber I-0 angeflogen wurde, hat der Schwarm den Tunnel nie betreten. Da dies auf der letzten Heatmap 4.19 nicht erkennbar ist, wurden zwei weitere im Vergleich 4.18 angelegt, die dieses Problem aufzeigen. Wie schon vorher besprochen, liegt das einerseits am Grundproblem des Swarm-Steerings und an der gemeinsamen Interest-Funktion, die für beide Ansätze genutzt wird. Die Entropie/Varianz-Graphen 4.20, die für beide Experimente aufgestellt wurden, zeigen bei der Varianz einen sehr großen Unterschied. Bei dem Experiment welches den Tunnel betreten hat, ist ein viel größerer Varianzanstieg zu beobachten, was die viel größere flächenmäßige Aufteilung des Schwarms im Raum beim Betreten des Tunnels zeigt. Im Vergleich dazu ist im unteren Graphen S5.7.7 mit Context-Steering abgebildet. Hier ist ein stärker oszillierendes Verhalten zu beobachten als beim Swarm-Steering. Allerdings ist die Varianz hier kleiner obwohl der Tunnel betreten wird. Da

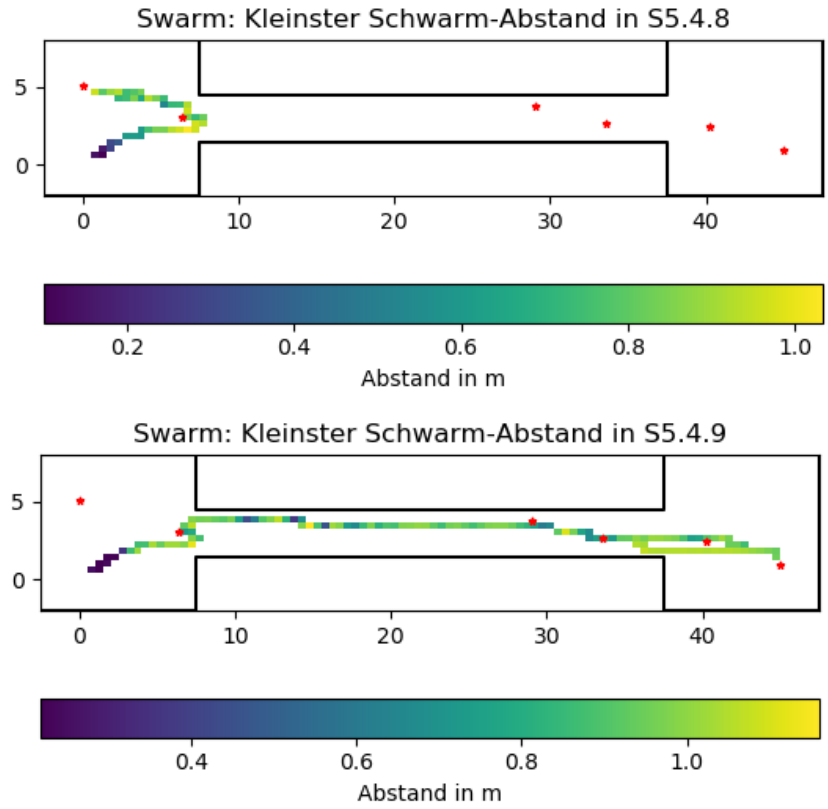


Abbildung 4.18.: Swarm-Heatmap-Vergleich S5.4.8 und S.5.4.9

der Context-Steering-Schwarm sehr häufig im Tunnel die Karte verlässt, muss er sich nicht so stark an ihn anpassen, wie es beim Swarm-Steering nötig ist. Wenn sieben Kopter fliegen ist zusätzlich die Tunnelgröße ein Problem. In S5 hat diese Kombination die gleichen Probleme wie in S4, welche mit einem noch größeren Tunnel in S6 gelöst wurden. Dass S5 bei zwei und vier Bots funktioniert und bei sieben nicht kann nur daran liegen, dass bei kleineren Bot-Mengen auch weniger Bots den richtigen Weg erkennen müssen. Wenn von sieben Bots nur einer so liegt, dass er einen IP durch den Tunneleingang wahrnimmt und deshalb eine kleinere Abstoßungskraft in der Richtung hat, kann er durch die eigenen Schwarm-Anziehungskräfte den Schwarm fast nicht in die richtige Richtung ziehen. Beim Swarm-Steering muss ein IP so positioniert sein, dass in dessen Richtung der Tunneleingang liegt. Dadurch ist die entgegengesetzte Dangerwall-Kraft wesentlich kleiner als bei den anderen IPs und der Kopter neigt dazu, in diese Richtung zu fliegen. Zusätzlich erhöhen

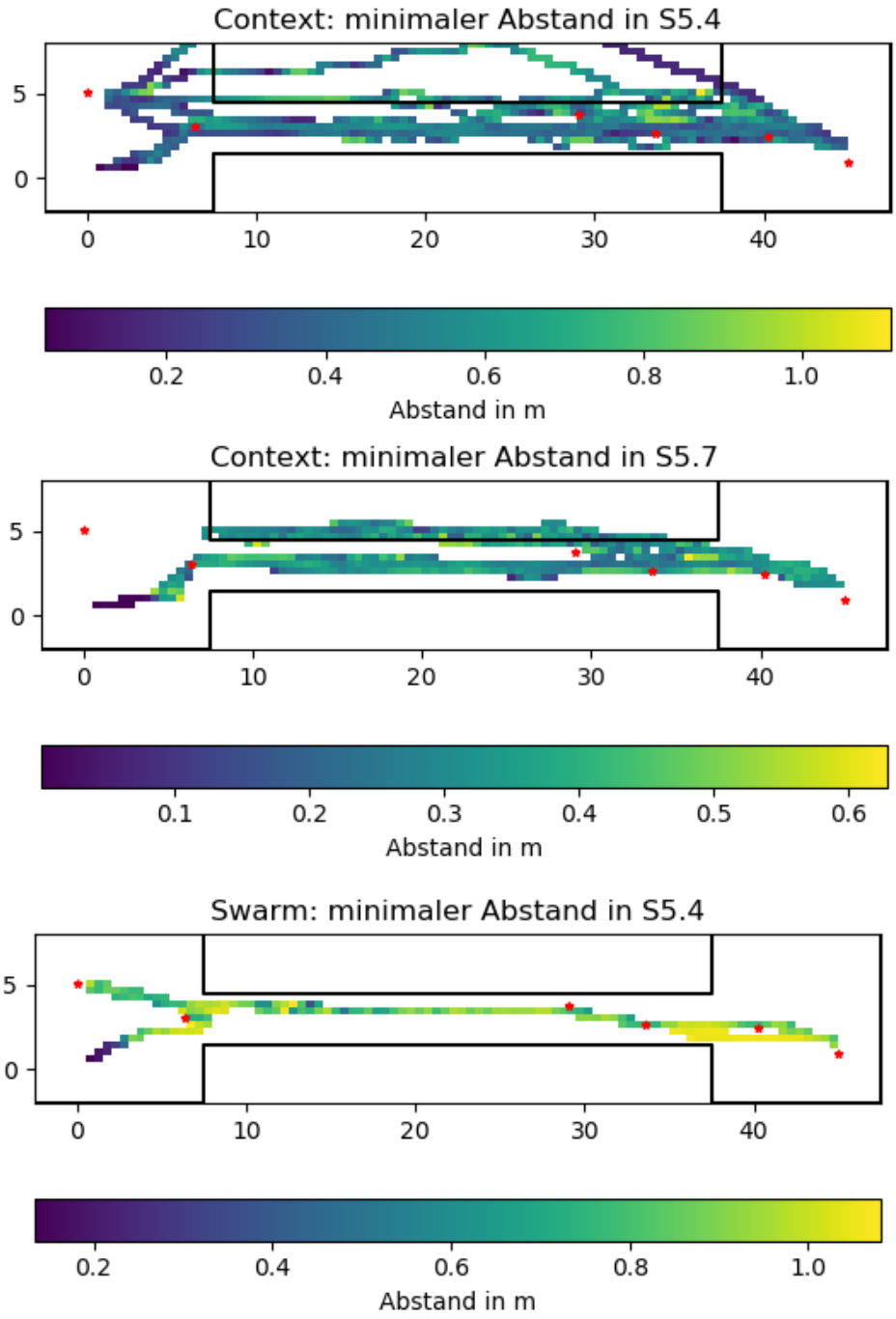


Abbildung 4.19.: Verschiedene Heatmaps für S5

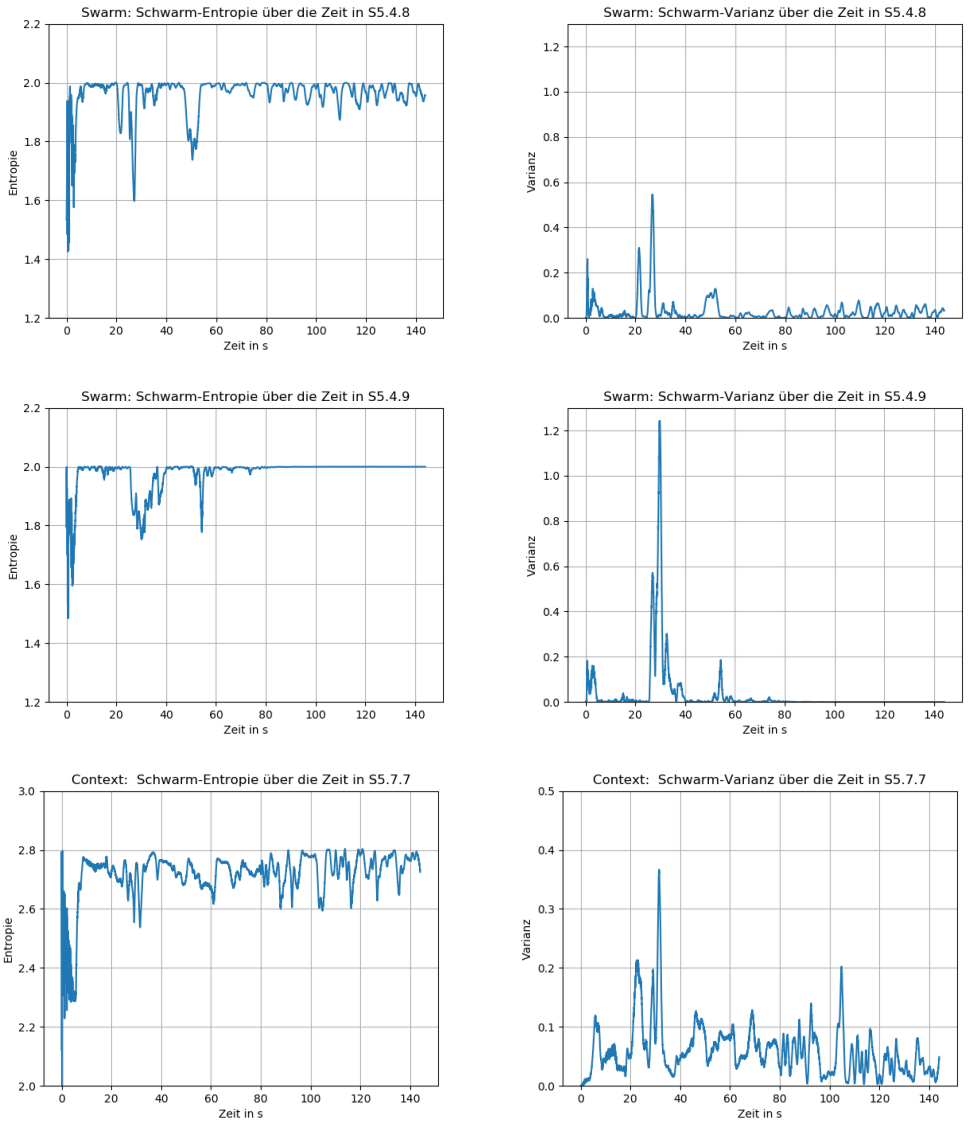


Abbildung 4.20.: Entropy/Varianz in in S5

die größeren Bot-Mengen die wirkenden Schwarmkräfte sehr stark, wodurch die richtige Wegfindung merklich erschwert wird.

Szenario 7

Durch die Auswertung der vorangegangen Szenarien ist davon auszugehen, dass S7 das schwerste Szenario sein sollte. Das aus S1 und S4 beschriebene Verhalten tritt auch in S7 auf. Das sind die Szenarios bei denen der Tunnel einen sehr kleinen Durchmesser hat. Die besprochenen Probleme der Interest-Funktion treten hier bei beiden Ansätzen am stärksten auf. Swarm-Steering erreicht in allen Experimenten nur I-1. Dies ist der IP, der am nächsten am Startpunkt liegt. In der weiteren Zeit wird weder der Tunnel noch I-0 angeflogen, was sehr gut in der zweiten Heatmap 4.21 erkennbar ist. I-0 und der

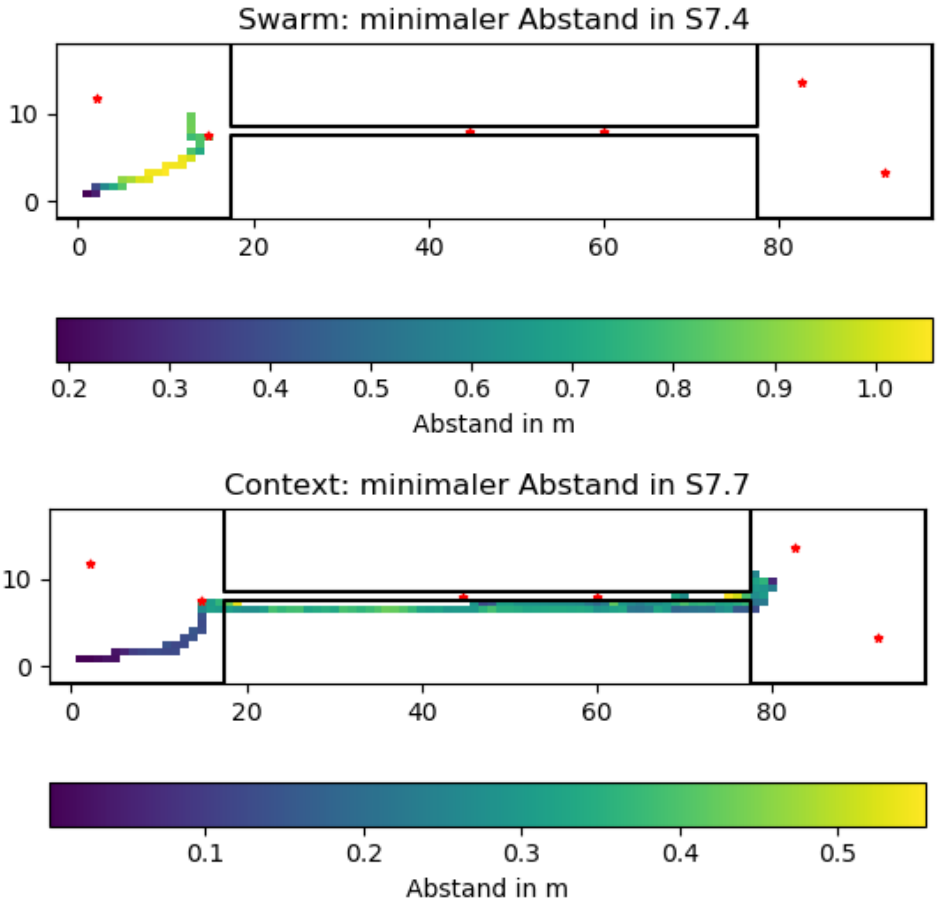


Abbildung 4.21.: Verschiedene Heatmaps für S7

Tunneleingang liegen mehr als 10 m entfernt, was zu einer zu kleinen Kraft führt, um den Schwarm in eine Richtung zu bewegen. Beide Kräfte sind fast entgegengesetzt gerichtet, da I-1 zwischen dem Tunneleingang und I-0 liegt. Durch das Abfliegen von nur einem Punkt sind die Ergebnisse der verschiedenen Box-Whisker-Plots eindeutig. Im Gegensatz zum Swarm-Steering fliegt Context-Steering den Tunnel an und erreicht beide in ihm enthaltenen IPs. Der Tunnel wird, wie auch schon in den Szenarios mit mittlerer Karte bei etwa 30 s erreicht, was in dem in Darstellung 4.22 abgebildeten Varianz-Graphen sehr gut erkennbar ist. Durch die 100 m breite Karte dauert das allerdings so

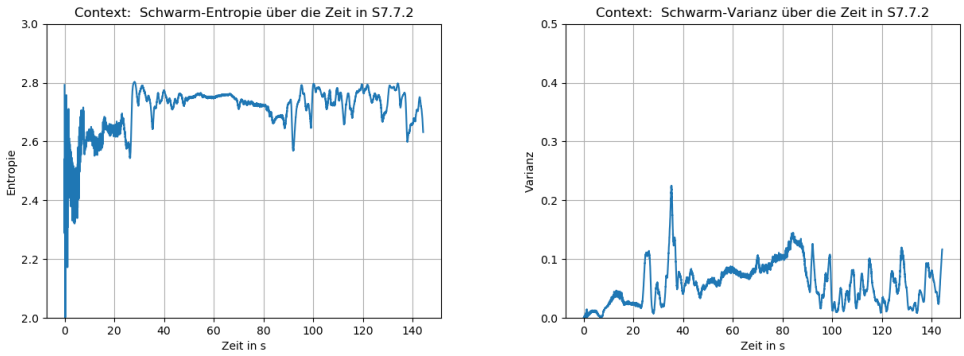


Abbildung 4.22.: Entropy/Varianz in in S7.7

lange, dass das Experiment zu Ende ist, bevor die Kopter alle IPs im rechten Raum erreichen können oder im Fall von sieben Bots, nur der Eingang in den rechten Raum erreicht wird. Zusätzlich wird der letzte Interest ungefähr bei 120 s im Experiment erreicht und bei der großen Karte und der suboptimalen Interest-Funktion sind die verbleibenden 30 s nicht ausreichend. Anhand der zweiten Heatmap aus Abbildung 4.21 ist leicht erkennbar, dass der Schwarm den rechten Raum erreicht aber niemals einen der beiden Interest-Punkte. In Abbildung 4.23 ist zu erkennen, dass der Schwarm zusätzlich sehr häufig die Karte verlässt. Dieses Verhalten tritt besonders in Szenarien auf, bei denen der Tunneldurchmesser nur einen Meter beträgt (S1,S4,S7). In S4 wird die Karte am häufigsten verlassen, wobei sich das sich nicht in der Länge des Verlassens widerspiegelt. Anhand der zweiten Szenario Heatmap 4.21 ist zu erkennen, dass der Schwarm-Mittelpunkt außerhalb des Tunnels liegt. Das bedeutet, dass ein Teil der Kopter nicht im Tunnel fliegt, wodurch die Kopter die Karte weniger verlassen aber sich trotzdem längere Zeit außerhalb befinden. Es ist ziemlich wahrscheinlich, dass Context-Steering mit zusätzlicher Zeit die

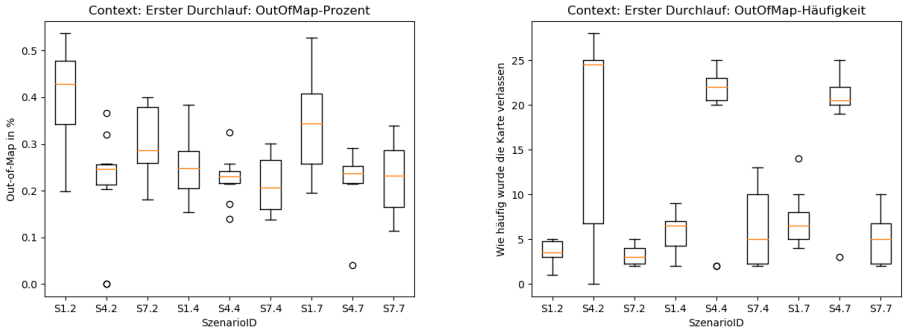


Abbildung 4.23.: OutOfMap-Auswertung in S1,S4 und S7

fehlenden IPs alle abgeflogen hätte. Demnach sollte es neben der geänderten Interest-Funktion mehr Zeit für die einzelnen Versuche geben.

Szenario 8/9

In den letzten beiden Szenarien wiederholt sich das gleiche Verhalten wie im Vergleich zwischen S3 und S4/5. Swarm-Steering betritt den Tunnel bei etwa 25 s, braucht allerdings eine sehr lange Zeit, um ihn zu durchfliegen und erreicht deshalb nicht das Tunnelende. Das ist sowohl an den verschiedenen Swarm-Heatmaps 4.24 erkennbar. Zusätzlich verlässt kein Bot ein einziges Mal die

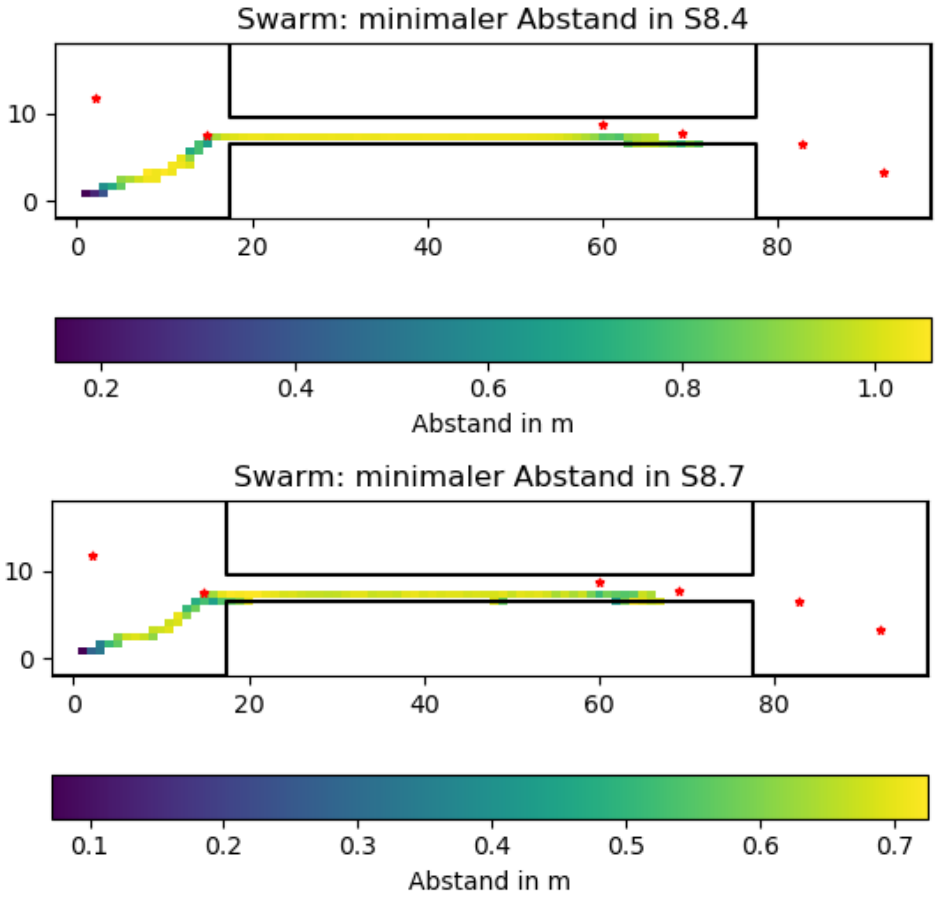


Abbildung 4.24.: Verschiedene Swarm-Steering-Heatmaps für S8/9

Karte. Im Gegensatz dazu verlässt Context-Steering immer wieder die Karte, sammelt aber auch mehr Interest. Wie auch schon in den Vorgängerszenarien wird I-0 nicht ein einziges Mal erreicht, was in den drei Heatmaps 4.25 erkennbar ist. Die Karte ist viel zu groß, um den ausgelassenen IP auf dem Rückweg einsammeln zu können.

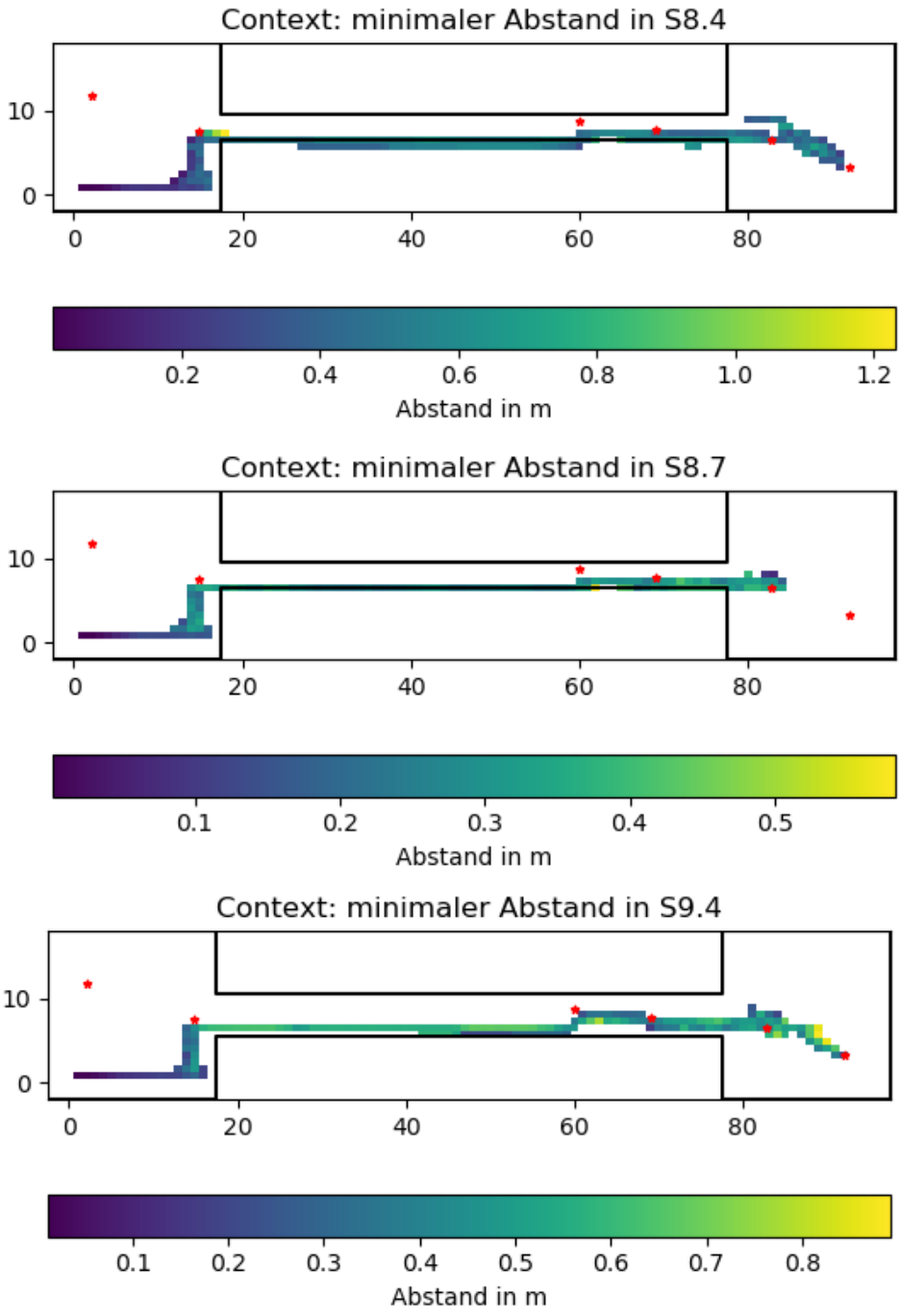


Abbildung 4.25.: Verschiedene Context-Steering-Heatmaps für S8/9

Zusammenfassung

Innerhalb der neun Szenarien wurden beim Context-Steering 136 von 270 (50 %) Experimenten erfolgreich durchlaufen. Swarm-Steering weißt eine Erfolgsquote von 33 % mit 90 von 270 Durchläufen auf. Die bei den Fehlschlägen auftretenden und sich wiederholenden Probleme konnten ausgemacht und analysiert werden. Mithilfe von drei großen Parameteränderungen sollen diese auftretenden Schwachstellen im nächsten Durchlauf behoben oder minimiert werden.

1. Die **Interest-Funktion** muss angepasst werden, damit I-0 nicht als letzter eingesammelt wird. Der Punkt lag häufig näher am Schwarm, wurde aber ignoriert, da die Interest-Summe aller anderen IPs, die im Tunnel und im rechten Raum lagen, dagegen wirkte. So sollte die Funktion nähere Ziele viel stärker hervorheben als eine Gruppe, die weiter weg liegt.
2. Für die Verbesserung der Out-Of-Map-Werte muss einerseits die **Walldanger** und gleichzeitig der ϵ-Constraint für die Wand vergrößert werden. Hierbei ist aber zu bemerken, dass der kleinste Tunnel aufgrund der internen Kommunikations-Simulations-Verzögerungen wahrscheinlich trotzdem nicht erfolgreich beflogen wird.
3. Swarm-Steering hat sich genau so verhalten, wie es erwartet wurde. Die Probleme, die schon in den Grundlagen besprochen wurden, sind genau so innerhalb der Experimente aufgetreten. So kann im Endeffekt neben den oben genannten Funktion nur die **Experimentzeit** erhöht werden, wovon Context-Steering auch profitiert.

Wenn der Context-Steering Ansatz unter der Änderung der oben genannten Punkte weniger die Karte verlässt ohne seine Geschwindigkeit zu verlieren, hat er klare Vorteile gegenüber Swarm-Steering.

4.3. Zweiter Durchlauf

4.3.1. Vorbereitung

Um die neuen Funktionen für die Interest- und Walldanger-Funktionen zu bestimmen, werden mögliche Kandidaten in den Szenarien getestet, in denen das Problem am meisten aufgetreten ist. Diese Werte werden nur für Context-Steering experimentell getestet und dann auf Swarm-Steering übertragen. Zusätzlich werden falls nötig bei beiden Ansätzen noch die Gewichte angepasst. Da die Experimentzeit parameterunabhängig ist, wird sie sowohl für die kommenden Tests als auch für den zweiten Versuch verdoppelt, um beiden Ansätzen genug Zeit zu geben und die Zeit als kritischen Faktor für die Experimente zu eliminieren. Da das Durchfliegen der Wände teilweise auch durch die starke Interest-Funktion hervorgerufen wurde, wird diese Funktion in S8 experimentell bestimmt. Hier wurde der IP I-0 nie eingesammelt, was das Kernproblem vieler Szenarien darstellte, weshalb nicht alle Punkte abgeflogen wurden. Zusätzlich ist das die größte Karte, weshalb der von den Koptern gesammelte Interest immer noch einen Einfluss auf das Gesamtbewegungs-System haben sollte. Die neue Interest-Funktion wird in Formel 4.6 dargestellt.

$$I(q_i|) = 10 \cdot k^{0,25 \cdot ||q_i - p_0||} \tag{4.6}$$

Der Faktor 10 hebt den Maximal-Wert bei einem Abstand von 0 m von eins auf zehn. Dadurch entsteht ein viel größerer Interest-Unterschied bei verschiedenen Entfernungen. Hiermit soll das Problem gelöst werden, dass der addierte Interest-Wert von mehreren weiter entfernten IPs größer ist als der Wert von einem, der näher dran liegt. Die $0,25$ wurde gewählt, um den Anstieg der Kurve langsamer abflachen zu lassen, wodurch das Interest langsamer sinkt. Durch den langsameren Interest-Verlust ist er bei 10 m mit $0,821$ immer noch sehr relevant und fällt erst bei 21 m unter die Minimal-Interest-Grenze. Das sind in den Szenarien sehr realistische Entfernungen für IPs. Im Vergleich zur alten Funktion $\frac{1}{||q_i - p_j||}$ sank der Interest am Anfang sehr schnell und bei einer Entfernung zwischen Kopter und IP ab fünf Meter war der Interest-Unterschied so klein, dass IP-Gruppen immer einen höheren Interest-Wert erhielten als einzelne näher liegende Punkte. Das Szenario wird mit verscheidenen k-Werten durchgeführt, die in Abbildung 4.27 gezeigt werden. Zusätzlich ist dort die alte Funktion in rot abgebildet. Die Idee der einzelnen Faktoren liegt darin, die Funktion vor allem in die Breite zu ziehen, damit auch noch bei Zielen, die nicht nur zwei Meter entfernt sind, ein nicht minimaler Interest-Wert vorliegt. Durch

die drei k-Werte entstehen drei Funktionen, die sich je höher sie sind, immer mehr der Funktionsweise der alten Interest-Funktion annähern. Die Ergebnisse dieser drei Experimente sind in den Heatmaps 4.27 und deren benötigten Experimentzeiten 4.28 dargestellt. An der dritten Heatmap ($k = 5$) ist zu

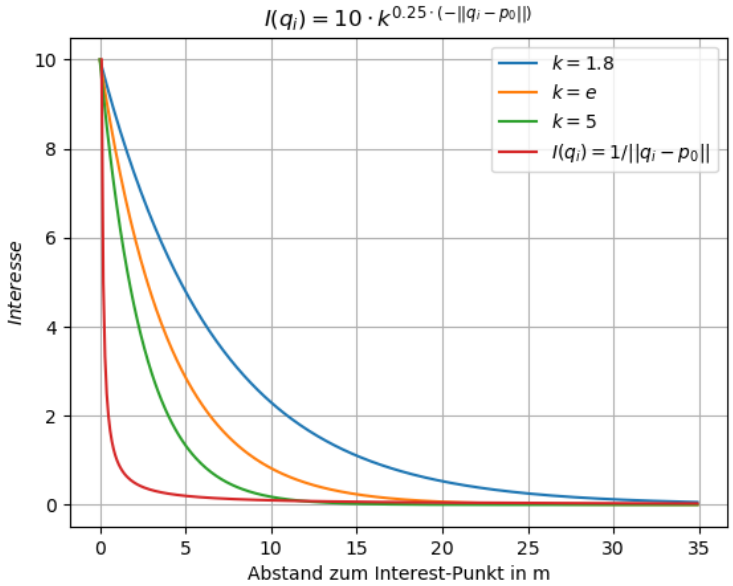


Abbildung 4.26.: Vergleich der verschiedenen neuen Interest-Funktionen

erkennen, dass das bestehende Problem mit dem Interest nicht gelöst wurde. Die beiden ersten Heatmaps konnten das Problem allerdings lösen. Bei den Zeiten ist kein großer Unterschied zwischen den beiden Funktionen zu erkennnen. Eine fünf Sekunden Schwankung ist bei einer Gesamtzeit von $155 - 162\,\mathrm{s}$ vernachlässigbar. Allerdings wurden bei dem Szenario mit $k = 1.8$ bei einem Experiment nicht alle IPs abgeflogen, wie es das Szenario mit $k = e$ getan hat. Deshalb wird für den zweiten vollständigen Durchlauf die Formel 4.7 genutzt.

$$I(q_i) = 10 \cdot e^{0{,}25 \cdot (-||q_i - p_0||)} \tag{4.7}$$

Mit dieser Interest-Funktion wurden die WallDanger-Tests in S5 durchgeführt. Während des ersten Durchlaufs haben hier die Kopter am häufigsten und längsten die Karte im Tunnel verlassen, wenn davon ausgegangen wird, dass das kleine Szenario (S1) zu klein für die Quadcopter ist. Da die Wall-Danger aufgrund der im ersten Durchlauf auftretenden Probleme erhöht werden musste und die neuen Interest-Funktion höhere Werte erreichen kann, wird die

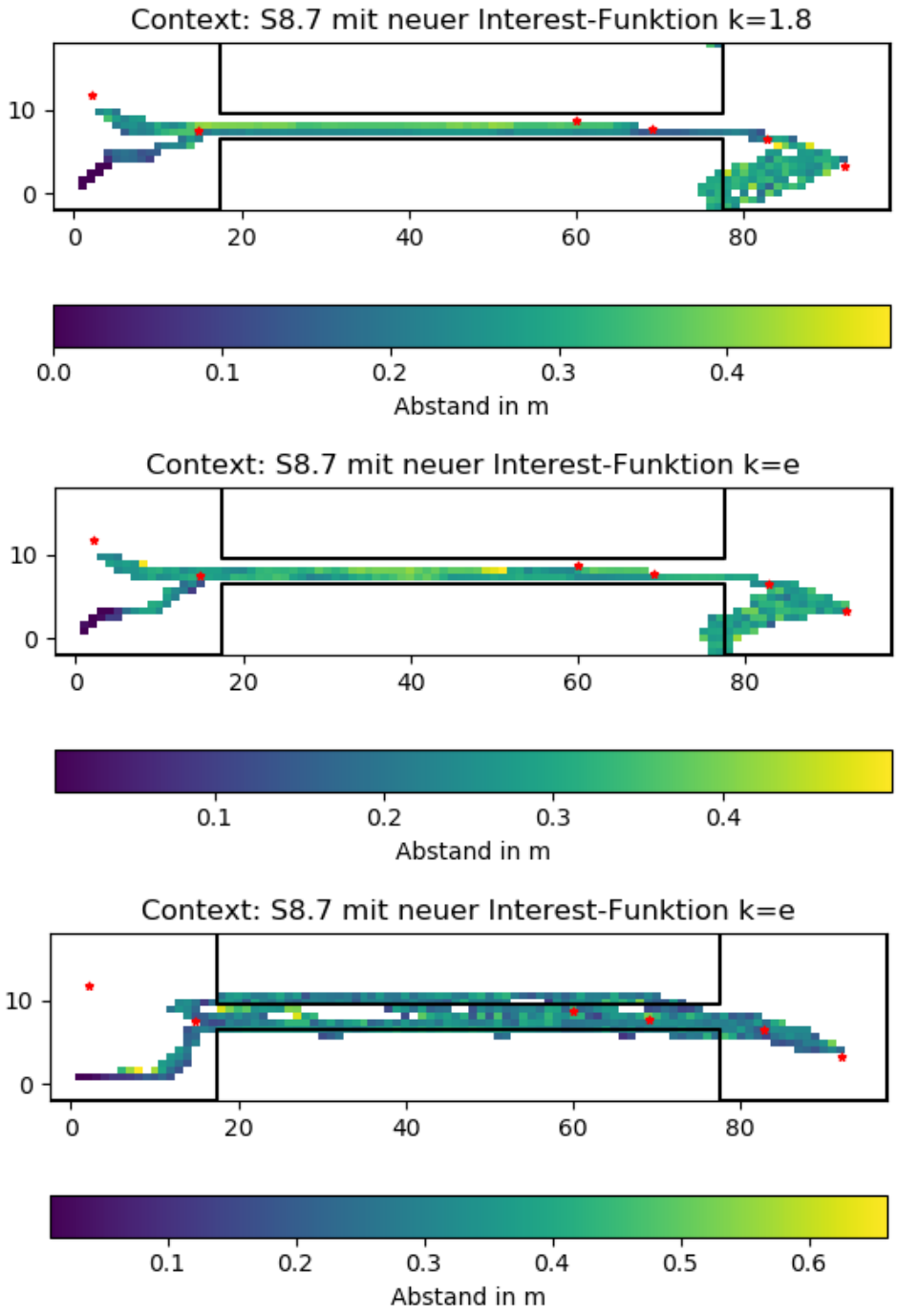


Abbildung 4.27.: HeatMaps der neuen Interest-Funktionen

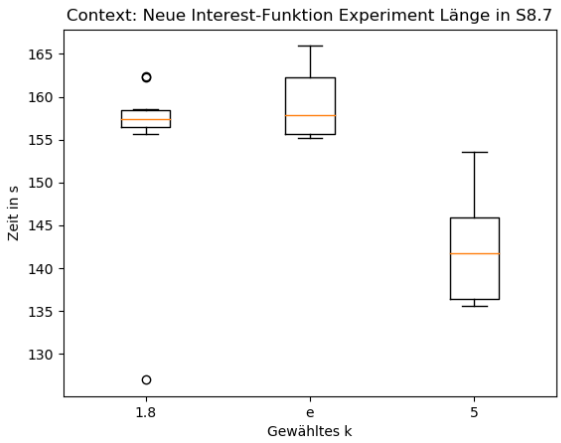


Abbildung 4.28.: Zeitpunkt des letzten IPs mit neuer Interest-Funktion

Original-Funktion $W(x) = \frac{1}{x}$ mit einem Faktor multipliziert. Diese werden in Abbildung 4.29 mit der neuen Interest-Funktion (lila) dargestellt. Hier ist erkennbar, dass die Distanz, ab der der Danger-Wert größer ist als der Interest-Wert, viel größer ist als bei der Original-Funktion. Die ϵ-Grenzwert werden für die jeweilige Funktion immer auf den Wert gelegt, bei dem die Distanz $0,5\,\mathrm{m}$ unterschreitet. Aus den Experimenten mit diesen drei Funktionen ergaben sich die Ergebnisse, die in den HeatMaps 4.30 dargestellt wurden. Abbildung 4.31 stellt die Dauer der Experimenta dar. Wichtig ist, dass das undefinierte Verhalten, was aufgetreten ist sobald alle Punkte abgeflogen wurden, nicht behoben wurde. Deshalb wurden die Ergebnisse so angepasst, dass das Verhalten nicht mit in die Auswertung übernommen wird. Die Ergebnisse aus dem folgenden Durchlauf werden auch in dieser Art bearbeitet werden. Anhand der drei Heatmaps 4.30 ist sehr gut zu erkennen, dass der Schwarm nie die Karte verlassen hat und je stärker die Walldanger war, um so mehr ist der Schwarm in der Mitte des Tunnels geflogen und war wesentlich dichter zusammen. Bei der stärksten Funktion mit einem Faktor von 7.5 hatte der Schwarm einen Minimalabstand von ungefähr $0,4\,\mathrm{m}$ im Tunnel, wobei dieser bei den kleineren Danger-Funktionen auf bis zu $0,8\,\mathrm{m}$ stieg. Dadurch, dass der Schwarm im Tunnel mehr Platz hatte, konnte er auch schneller fliegen. In Abbildung 4.31 ist erkennbar, dass die Experimente durchschnittlich auch fünf Sekunden länger gedauert haben. Aus diesen Ergebnissen wurde die Formel 4.8 mit einem

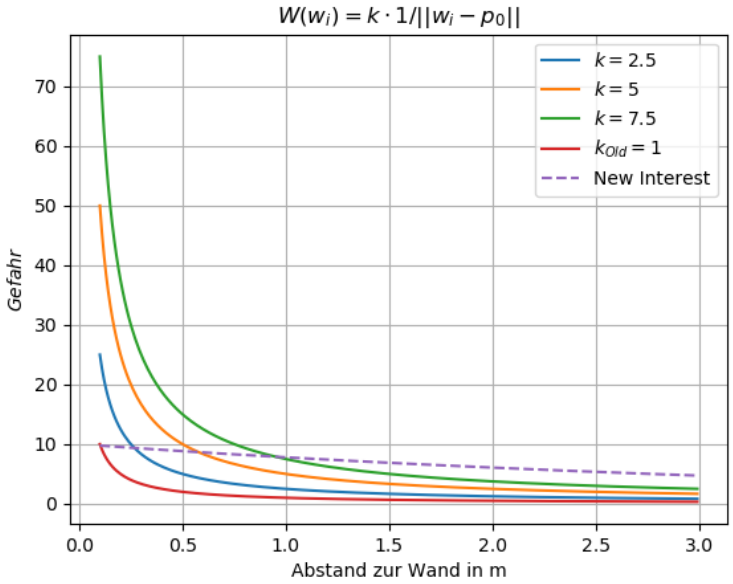


Abbildung 4.29.: Vergleich der verschiedenen neuen Wall-Danger-Funktionen

Parameter	Gesetzter Wert
$w_{Interest}$	$0,5$
$w_{DangerWall}$	$0,3$

Tabelle 4.7.: Konfigurations-Änderung für den 2. Durchlauf in Swarm-Steering

Faktor von fünf als neue Walldanger-Funktion festgelegt.

$$W(w_i) = 5 \cdot \frac{1}{||w_i - p_0||} \tag{4.8}$$

Die neuen Interest -und Dangerfunktionen wurden daraufhin in den Swarm-Steering Ansatz übertragen. Bei einem Durchlauf fiel allerdings auf, dass eine starke Anpassung der Gewichte nötig war. Die neue Interest-Funktion gab viel höhere Werte zurück als die alte. Die Geschwindigkeit der Kopter wurde durch angepassten Interest-Werte zu stark erhöht. Dadurch flogen die Kopter viel zu schnell, um vernünftig innerhalb der Map zu navigieren. So wurden die Gewichte von sowohl Interest als auch Danger stark reduziert basierend auf der geflogenen Geschwindigkeit in S1. Die genauen Gewichtsänderungen sind in Tabelle 4.7 dargestellt. Mit diesen besprochenen Anpassungen kann der zweite Durchlauf gestartet werden.

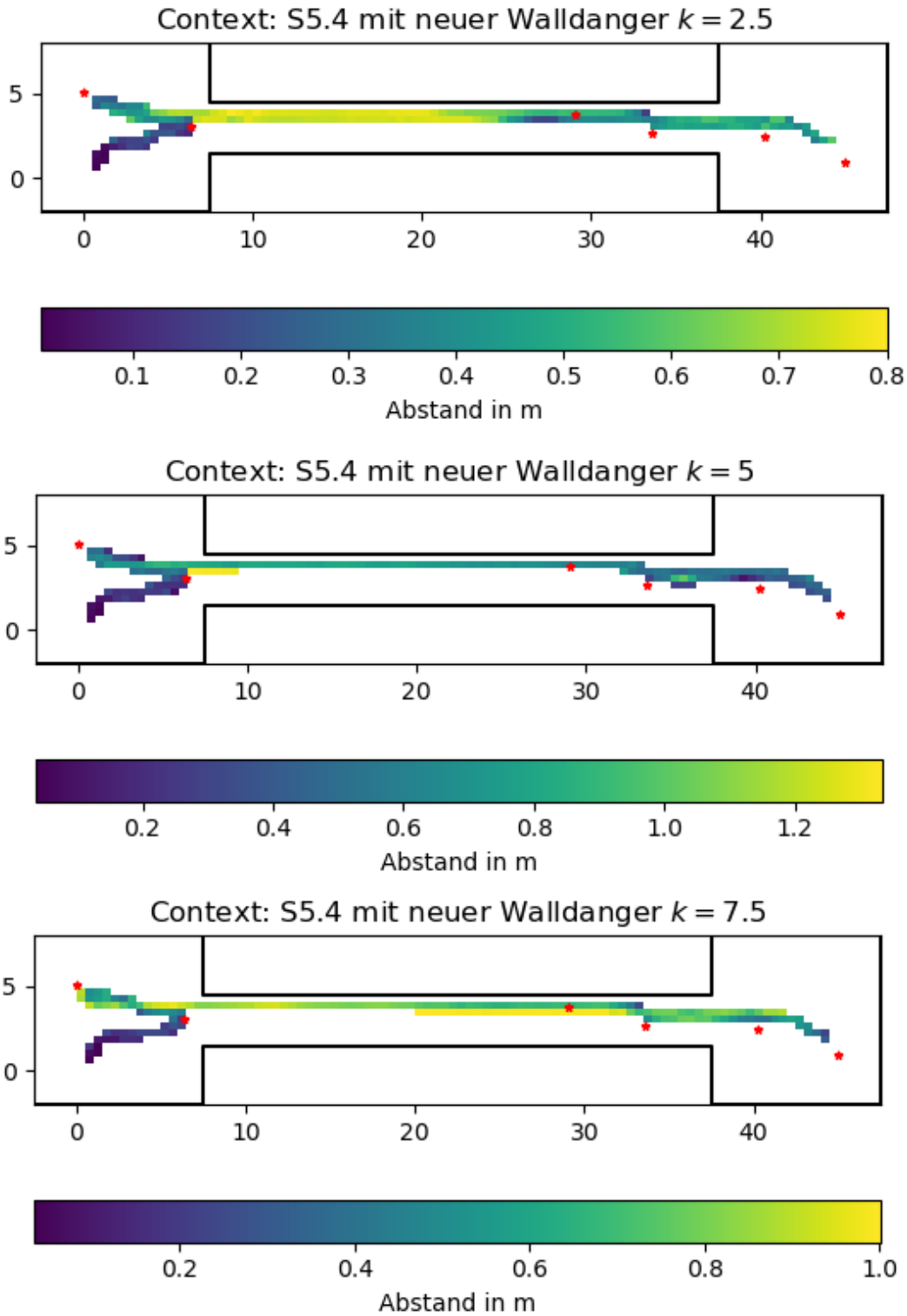


Abbildung 4.30.: HeatMaps der neuen WallDanger-Funktionen

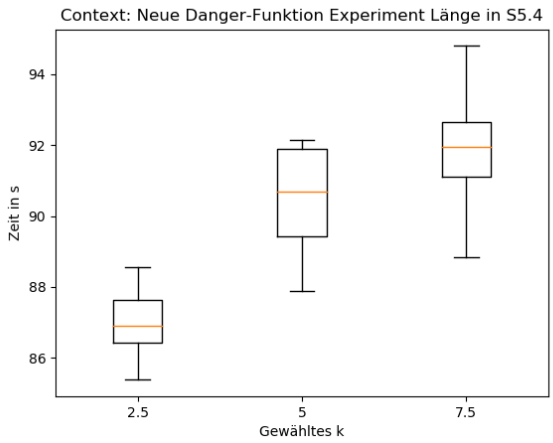


Abbildung 4.31.: Zeitpunkt des letzten IPs mit neuer Walldanger-Funktion

4.3.2. Durchführung

Wie auch im ersten Durchlauf werden die Experimente auf dem gleichen PC durchgeführt. Die Wegpunkt-Seeds werden zur besseren Vergleichbarkeit nicht verändert.

4.3.3. Auswertung

Bei der Auswertung vom zweiten Durchlauf wird sich hauptsächlich auf die Unterschiede zum Ersten bezogen. Die zu Beginn aufgestellten Tabellen 4.8 und 4.9 zeigen die ersten sehr großen Unterschiede. Hinzukommen die verschiedenen Box-Whisker-Plots und Heatmaps an wichtigen Stellen. Wie auch schon in der Vorbereitung für den zweiten Durchlauf wird die Analyse ab den Zeitpunkt beendet, an dem alle IPs in ausreichender Anzahl abgeflogen wurden, wodurch das unerwünschte Verhalten am Ende nicht mit in die verschiedenen Ergebnisdarstellungen aufgenommen wird.

Szenario 1-3

Die neuen Funktionen bewirken, dass Context-Steering in allen drei Szenarios mindestens zehn Sekunden schneller seine Aufgabe abschließt als im ersten Durchlauf. Zusätzlich sinkt der OutOfMap-Prozentsatz in S1 von 20 bis 45 % auf 5 % und in den anderen beiden Szenarien auf fast 0 %. Die Karte wird

Szenario-Index	maximal erreichte Interest-Punkte	Ergebnis
		Zwei Copter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	6	Alle Experimente haben alle Punkte erreicht
5	6	Alle Experimente haben alle Punkte erreicht
6	6	Alle Experimente haben alle Punkte erreicht
7	6	Fünf Experimente haben alle Punkte erreicht
8	6	Alle Experimente haben alle Punkte erreicht
9	6	Alle Experimente haben alle Punkte erreicht
		Vier Copter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	6	Alle Experimente haben alle Punkte erreicht
5	6	Alle Experimente haben alle Punkte erreicht
6	6	Alle Experimente haben alle Punkte erreicht
7	5	I-4 wurde nie erreicht
8	6	Alle Experimente haben alle Punkte erreicht
9	6	Alle Experimente haben alle Punkte erreicht
		Sieben Copter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Alle Experimente haben alle Punkte erreicht
3	6	Alle Experimente haben alle Punkte erreicht
4	6	Alle Experimente haben alle Punkte erreicht
5	6	Alle Experimente haben alle Punkte erreicht
6	6	Alle Experimente haben alle Punkte erreicht
7	6	Ein Experiment hat alle Punkte erreicht
8	6	Sechs Experimente haben alle Punkte erreicht
9	6	Acht Experimente haben alle Punkte erreicht

Tabelle 4.8.: Context-Steering-Ergebnisse für den zweiten Durchlauf

Szenario-Index	maximal erreichte Interest-Punkte	Ergebnis
		Zwei Copter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Sechs Experimente haben alle Punkte erreicht
3	6	Drei Experimente haben alle Punkte erreicht
4	6	Ein Experiment hat alle Punkte erreicht
5	2	I-0 und I-1 sind die einzigen erreichten Punkte
6	6	Alle Experimente haben alle Punkte erreicht
7	2	I-0 und I-1 sind die einzigen erreichten Punkte
8	2	I-0 und I-1 sind die einzigen erreichten Punkte
9	2	I-0 und I-1 sind die einzigen erreichten Punkte
		Vier Copter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Zwei Experimente haben alle Punkte erreicht
3	4	I-4 und I-5 wurden nie erreicht
4	2	I-0 und I-1 sind die einzigen erreichten Punkte
5	2	I-0 und I-1 sind die einzigen erreichten Punkte
6	6	Ein Experiment hat alle Punkte erreicht
7	2	I-0 und I-1 sind die einzigen erreichten Punkte
8	2	I-0 und I-1 sind die einzigen erreichten Punkte
9	2	I-0 und I-1 sind die einzigen erreichten Punkte
		Sieben Copter
1	6	Alle Experimente haben alle Punkte erreicht
2	6	Zwei Experimente haben alle Punkte erreicht
3	6	Vier Experimente haben alle Punkte erreicht
4	2	I-0 und I-1 sind die einzigen erreichten Punkte
5	2	I-0 und I-1 sind die einzigen erreichten Punkte
6	2	I-0 und I-1 sind die einzigen erreichten Punkte
7	2	I-0 und I-1 sind die einzigen erreichten Punkte
8	2	I-0 und I-1 sind die einzigen erreichten Punkte
9	2	I-0 und I-1 sind die einzigen erreichten Punkte

Tabelle 4.9.: Swarm-Steering-Ergebnisse für den zweiten Durchlauf

trotzdem gleich oft verlassen. Innerhalb der drei Szenarios gibt es keine merklichen Unterschiede zwischen verschiedenen Bot-Mengen.
Beim Swarm-Steering haben sich die Ergebnisse anhand des maximalen Interest-Gains stark verschlechtert. Vor den Änderungen haben alle Konfigurationen alle drei Szenarios vollständig durchlaufen, wobei mit den Änderungen nur von 47 von 90 Experimenten einen erfolgreichen Durchlauf hatten. Das ist fast 50 % schlechter als im ersten Durchlauf. Der Grund dafür sind angepassten Interest- und Walldanger-Funktionen im Umweltteil der Kräfteberechnung. Diese sind vergleichsweise zu den anderen wirkenden Kräften viel zu stark, und mussten mit den Gewichten erheblich reduziert werden. Damit verlieren sie aber wieder ihre Wirksamkeit. Zusätzlich wurden die Gewichte nur für S1 angepasst, was bei einer Kartenabweichung in S2 und S3 zu unvorhersehbaren Problemen innerhalb des Ansatzes führen kann, wenn die Parameter nicht speziell darauf angepasst werden. Darüber hinaus verlässt der Schwarm wesentlich länger die Karte als bei den alten Parametern und steigt somit von ungefähr 10 % auf bis zu 55 % bei zwei Koptern. Anhand der Heatmaps 4.32 ist zu erkennen, dass die Danger-Werte mit den neuen Gewichten nicht groß genug waren und der Schwarm bei jedem Experiment die Karte verlassen hat.

Szenario 4-6

Im ersten Durchlauf enstanden hier die ersten sichtbaren großen Probleme, die dazu führten, dass einige Experimente nicht erfolgreich abgeschlossen wurden. Mit den neuen Funktionen steigt die Erfolgsrate von vorher 47 Experimenten (52 %) auf 90 Experimente (100 %). Zusätzlich dazu verkleinert sich bei allen drei Szenarios die dafür benötigte Zeit. Bei Szenario 5 und 6 von etwa 140 s auf 100 s mit einer Out-Of-Map-Prozent Verbesserung von 25 % auf nahezu 0 %. Dies ist auch sehr gut an den S6-Heatmaps 4.33 zu erkennen. Anhand der Boxplots 4.34 für S4 ist zwar Zeiterhöhung zu erkennen, was allerdings auch daran liegt, dass im ersten Durchlauf nicht alle Experimente vollständig waren und damit die Zeit die es gebraucht hätte, um den letzten fehlenden IP (I-0) abzufliegen, nicht in die Zeitberechnung der Boxplots A.3 abgebildet wird. Dadurch steigt die Experimentlänge auf einen Wert von mehr als 150 s, wodurch der zweite Durchlauf mit 100 s wieder kürzer ist. Zusätzlich sinkt der Out-Of-Map-Prozentsatz, dargestellt in Abbildung 4.35 von 25 % auf 10 % und ist damit auch viel kleiner als beim ersten Durchlauf. Die 10 % kommen

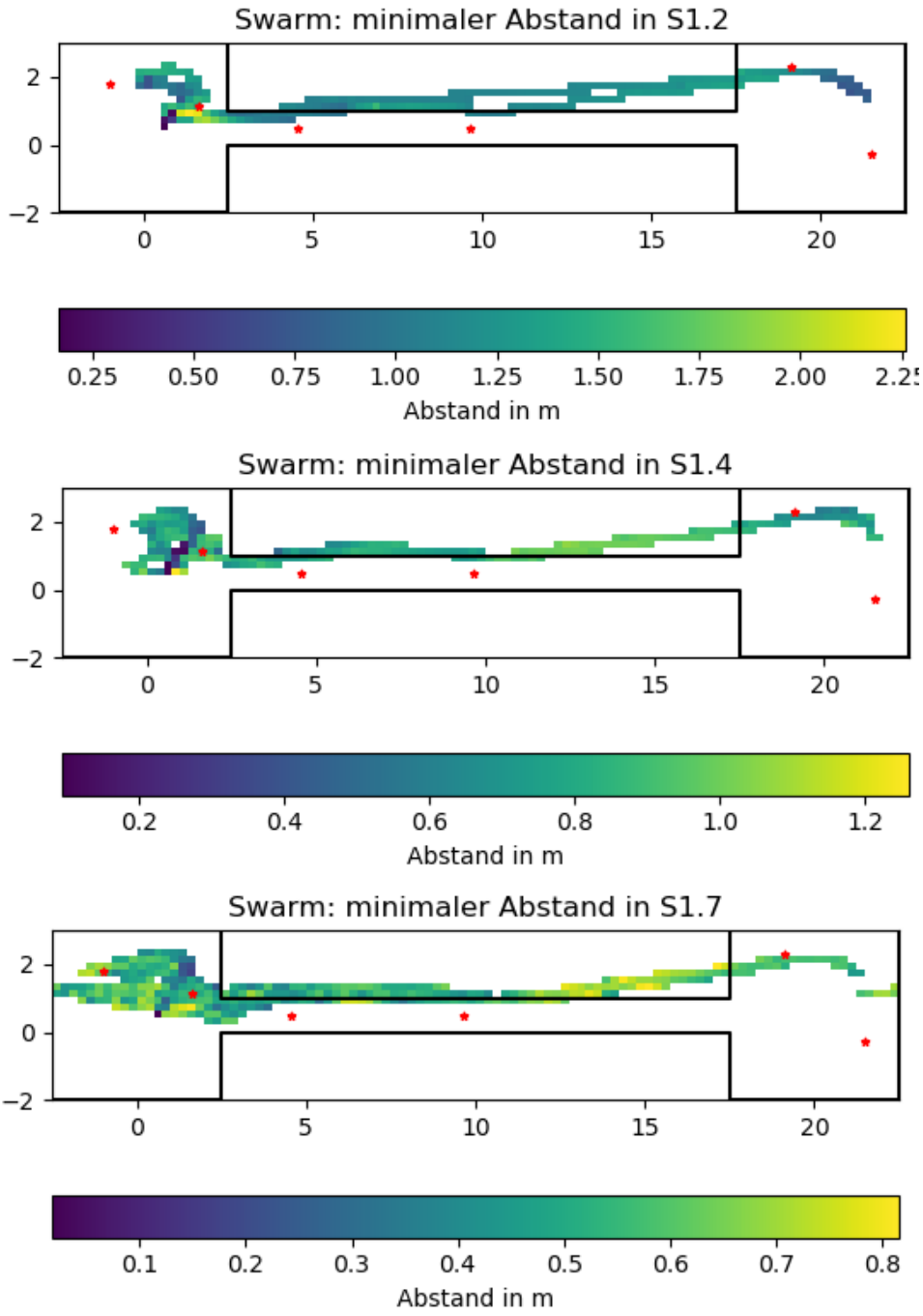


Abbildung 4.32.: Verschiedene Swarm-Steering-HeatMaps für S1

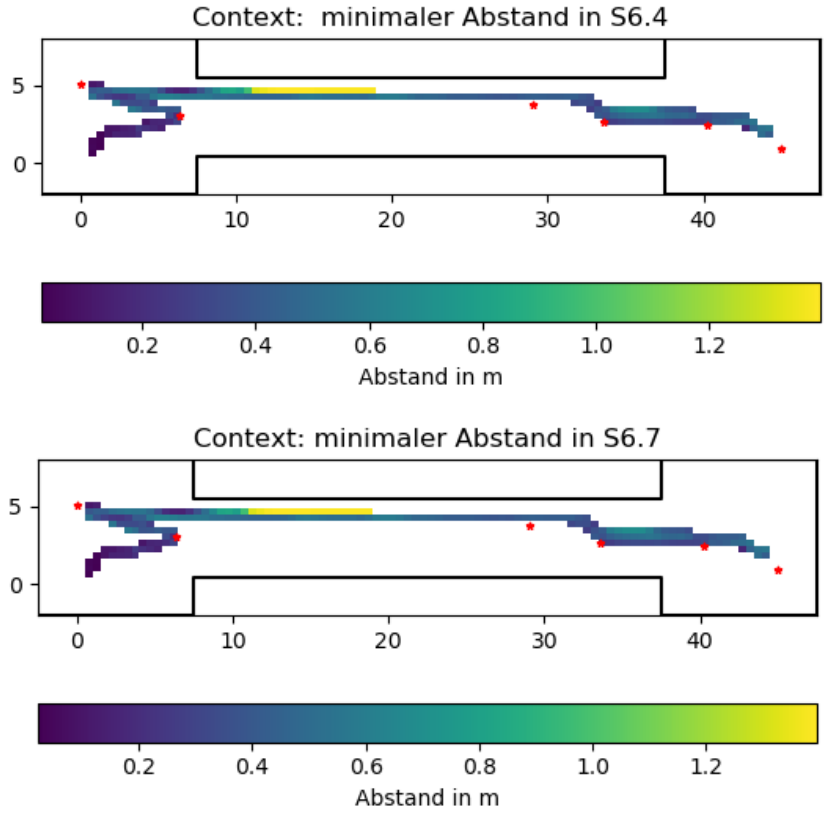


Abbildung 4.33.: Verschiedene Context-Steering HeatMaps für S6

daher, dass der Tunnel sehr schlecht gefunden wird und die Kopter teilweise beim Anflug die Karte verlassen, wie in den Heatmaps 4.37 zu erkennen ist. Innerhalb dieser Szenarios gibt es beim Context-Steering wieder keinen großen Unterschied bei den verwendeten Bot-Mengen. Das ist allerdings beim Swarm-Steering nicht der Fall . Hier schneiden größere Bot-Mengen schlechter ab als kleinere. Dies fällt vor allem bei S6 auf. Hier sinkt die Erfolgsrate von 100 % bei zwei Koptern auf 10 % bei vier Koptern auf 0 % bei sieben Koptern. Wie an den HeatMaps 4.36 zu erkennen ist, erreichten die Experimente mit sieben Bots nicht die Mitte des Tunnels und waren trotz der doppelten zur Verfügung stehenden Zeit wesentlich langsamer als bei den Versuchen im ersten Durchlauf. Ein Grund dafür kann die wesentlich höhere Walldanger sein. Durch die Änderung der Funktionen und ihrer Gewichte verschiebt sich das Kräftegleichgewicht erheblich und offensichtlich in eine nicht gewollte Richtung. Je mehr Bots im Experiment fliegen, um so wichtiger ist dieses Verhältnis. Wenn der Schwarm allerdings einen Weg findet kann er die Experimente sogar in

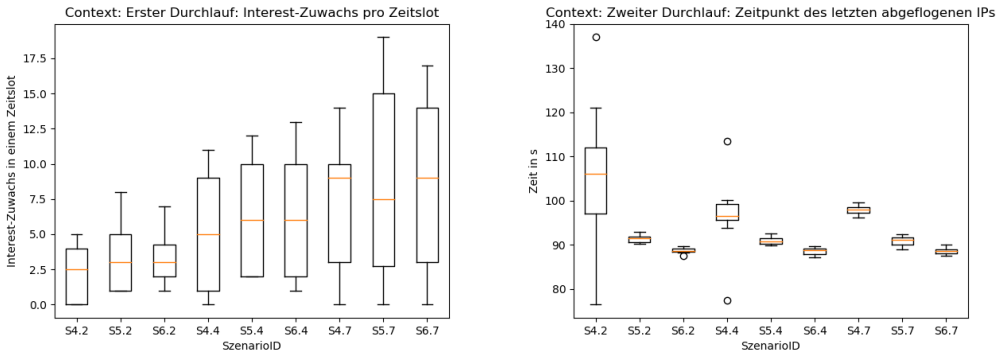


Abbildung 4.34.: Vergleich S4-6 pro Durchlauf mit letzter Interest-Erhöhung

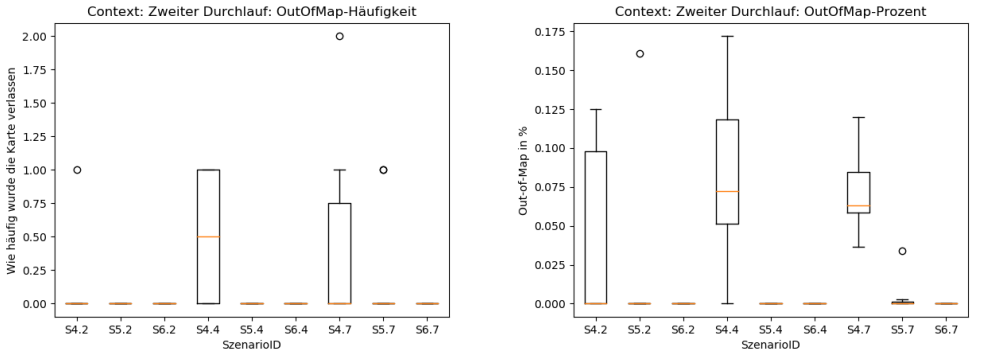


Abbildung 4.35.: Out-Of-Map-Werte in S4-6

40 s abschließen. Das ist 10 s schneller als Context-Steering. Damit performed Swarm-Steering in S4 bei zwei verwendeten Quadcoptern besser als Context-Steering. Zusätzlich steigt die Erfolgsquote von Swarm-Steering in Vergleich zum ersten Durchlauf von 0/90 auf 12/90. Sobald nicht alle Punkte abgeflogen werden, performt Swarm-Steering aber wesentlich schlechter. Außerhalb von S6 und einer Ausnahme in S4 werden nur I-0 und I-1 erreicht, was in einer wesentlich kleineren Gesamt-Interest über alle drei Szenarien im Vergleich zum ersten Durchlauf führt. Der Unterschied zwischen dem erfolgreichen Durchlauf aus S4 und einem nicht erfolgreichen ist in den Heatmaps 4.37 dargestellt.

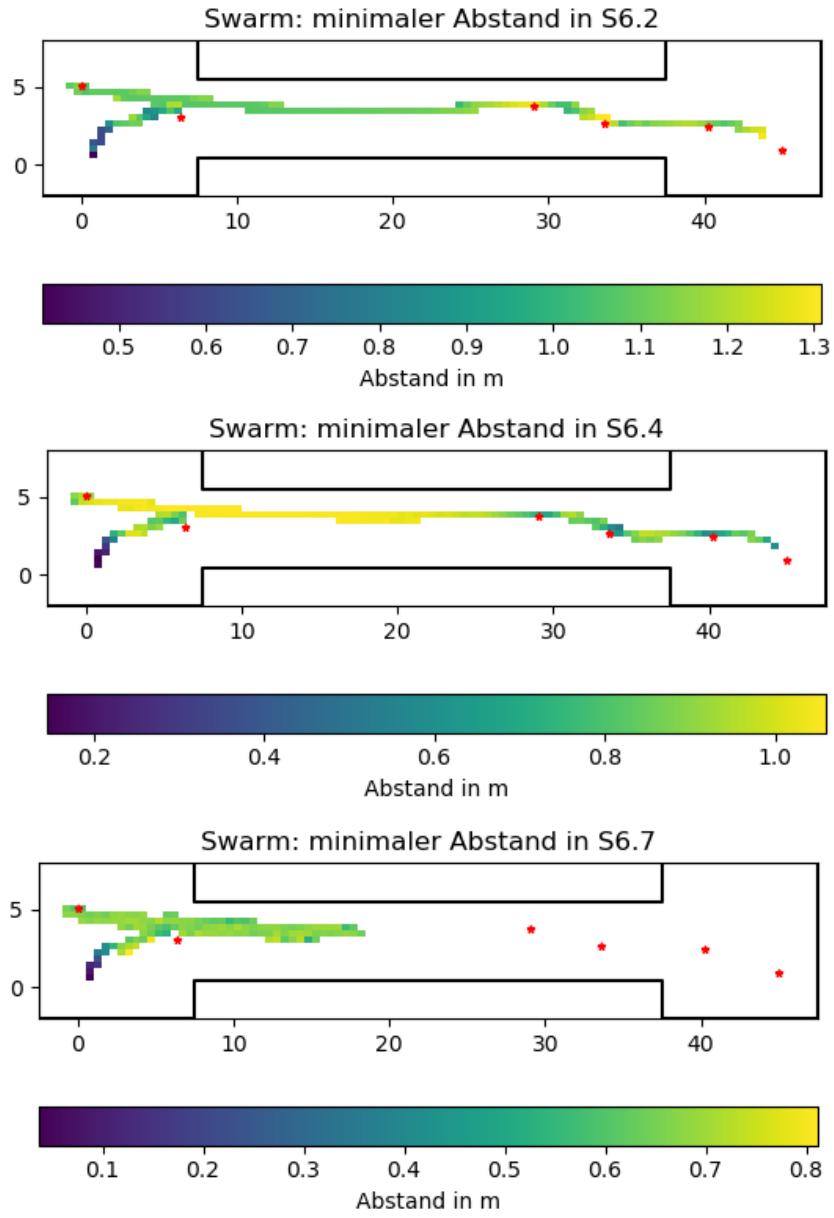


Abbildung 4.36.: Verschiedene Swarm-Steering HeatMaps für S6

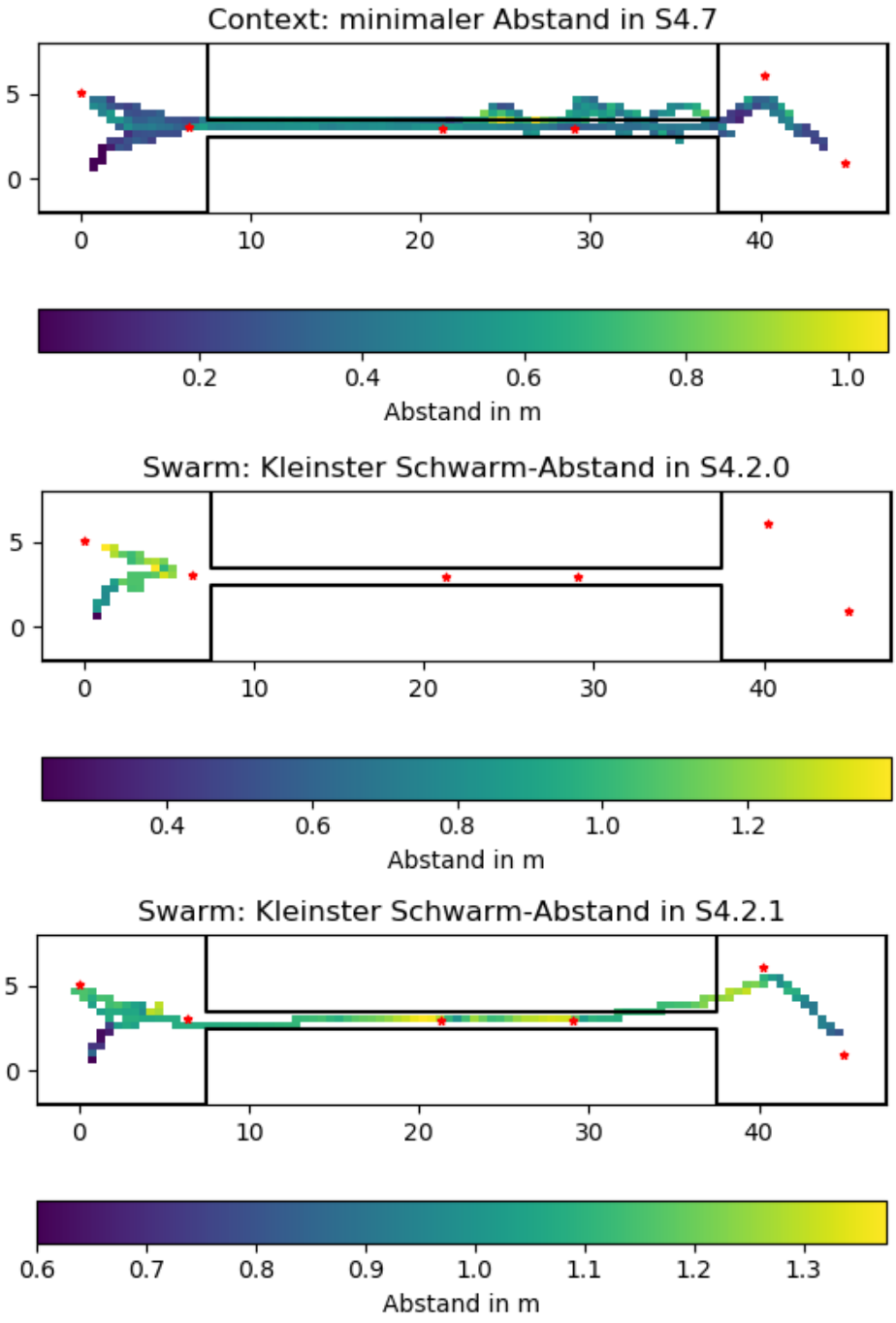


Abbildung 4.37.: Verschiedene HeatMaps für S4

Szenario 7-9

In den letzten drei Szenarios verbessert sich Context-Steering am stärksten. Die Erfolgsquote steigt von 0/90 auf 60/90. Allerdings steigt bei S8 und S9 die Out-Of-Map-Zeit erheblich von fast 0 % auf teilweise über 30 %. Anhand der Heatmaps 4.38 ist klar ersichtlich, dass die Parameter für sieben Agenten pro Experiment getestet wurden. Mit sieben Kopter verlässt der Schwarm die

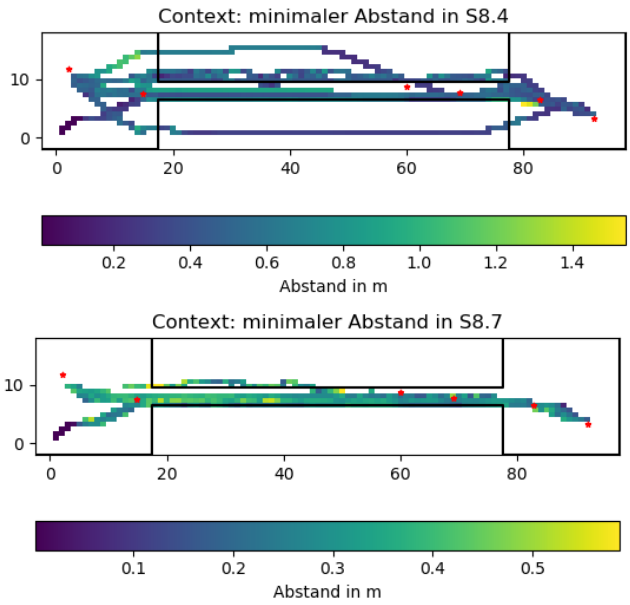


Abbildung 4.38.: Verschiedene Context-Steering-HeatMaps für S8

Karte über alle Experimente nur in sehr wenigen Ausnahmen. Bei vier Koptern tritt das Problem aus S5 aus dem ersten Durchlauf auf, dass der IP I-0 zuletzt eingesammelt wird. Anhand des Vergleichs der beiden Heatmaps 4.39 ist zu sehen, dass die Kopter auf dem Rückweg wieder die Karte verlassen. Das passiert aber nicht bei allen Durchläufen mit vier Koptern, wie auch in dem Heatmap-Vergleich 4.39 zu sehen ist. Dabei ist S8.4.0 das Experiment, dass IP I-0 auf dem Rückweg abfliegt und S8.4.6 direkt zu Beginn. Bei den den Entropie- und Varianzgraphen 4.39 ist darauf zu achten, dass S8.4.6 nur 160 s lang dauert und S8.4.0 im Vergleich fast 260 s. Innerhalb der ersten 160 s unterscheiden sich die beiden Graphen teilweise sehr stark. S8.4.0 hat zwischen 110 s und 140 s ein starkes Tief. Hier betritt der Schwarm zum zweiten mal den Tunnel auf dem Rückweg. Der Flug durch ihn hindurch dauert viel länger, da der generierte Interest von einem Punkt viel schwächer ist und damit

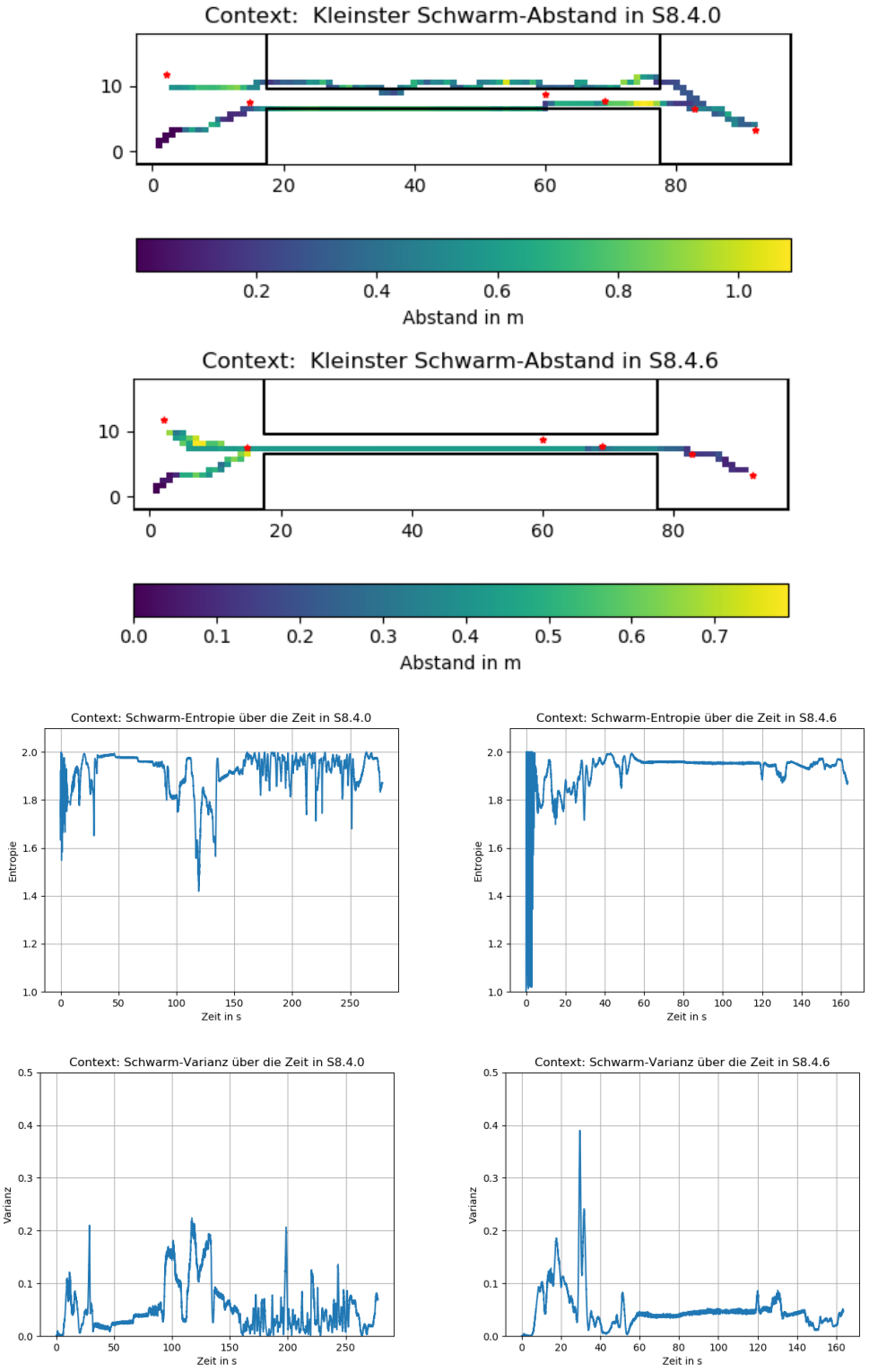


Abbildung 4.39.: Vergleich Context-Steering S8.4

der Schwarm nur schwach in die Richtung gelenkt wird. Da S8.4.6 I-0 sofort abfliegt, muss es den Tunnel nicht nochmal besuchen und hat deshalb nur ein starkes oszillierendes Verhalten am Anfang.

Durch das Verhalten, I-0 nicht zu Beginn mit abzufliegen, sind S8 und S9 die einzigen Szenarien im zweiten Durchlauf, bei dem der Schwarm wesentlich mehr als 15 % außerhalb der Karte war. Eine Besonderheit, die in S7.7 aufgetreten ist, ist das einzelne Kopter sich das erste mal vom Schwarm gelöst haben. Anhand der beiden Heatmaps 4.41 ist zu erkennen, dass das Zentrum des Schwarms im Tunnel verbleibt obwohl bei Experiment 4 alle IPs abgeflogen werden. Das ist auch an den Entropie -und Varianzgraphen 4.41 zu dem Experiment ersichtlich, da hier ab 160 s die Entropie stark abfällt. Dabei steigt die Varianz bei S7.7.3 bei 260 s auf über 160, was außerhalb aller bis dahin gemessenen Größenordnungen liegt. Um dieses Verhalten genauer zu untersuchen, wurde für die beiden Experimente ein neuer Graph 4.40 erstellt, der die mittlere Distanz der Kopter voneinander über die Zeit darstellt. Ab ungefähr

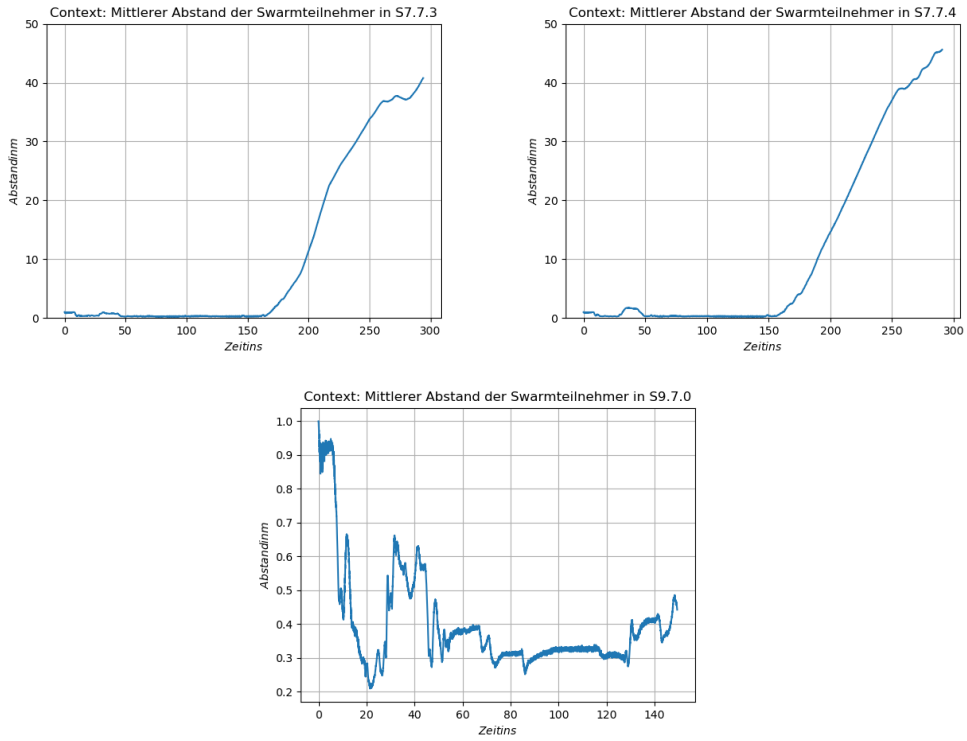


Abbildung 4.40.: Mittlerer Drohnenabstand über die Zeit

160 s steigt der durchschnittliche Abstand des Schwarms enorm an. Besonders

ist dabei, dass Kopter, die den Schwarm verlassen nicht nur für sehr kurze Zeit die Karte verlassen. In dem Vergleichs-Experiment 4.41, bei dem die letzten beiden IPs I-4 und I-5 nicht erreicht werden, ist ein ähnliches Verhalten zu beobachten. Da der Schwarm mit Swarm-Steering und den neuen Funktionen innerhalb von S4-6 viel zu langsam war, ist das Problem hier auch vertreten. Alle drei Kopter-Mengen erreichen nur den Anfang aller Tunnel, was in den Heatmaps 4.42 ersichtlich ist. Genau wie davor bewegen sich die Kopter viel zu langsam, um das Experiment erfolgreich abschließen zu können. Der Grund dafür ist der gleiche, der schon bei den letzten drei Szenarien besprochen wurde. Auf den noch größeren Karten fällt das jedoch noch mehr ins Gewicht.

4.4. Gesamtauswertung

Mit dem Abschluss des zweiten Durchlaufs können einige Aussagen über die beiden Steering-Ansätze getroffen werden. Context-Steering hat im zweiten Durchlauf sehr viel besser funktioniert und hat seine Erfolgsquote von 136/270 auf 240/270 und damit von 50 % auf 89 % erhöht. Zusätzlich sind die Out-Of-Map Zeiten in fast allen Fällen deutlich kleiner geworden. Obwohl diese kleiner geworden sind, konnte die Geschwindigkeit in der alle IPs abgeflogen sind beibehalten und teilweise verbessert werden. Trotzdem ist zu sagen, dass der kleinste Tunneldurchmesser von einem Meter in Verbindung mit einer größeren Karte die größte Herausforderung für beide Ansätze darstellt. Durch die gewählte Parametrisierung war der Schwarm häufig viel zu langsam, um effizient durch den Tunnel zu fliegen. Zusätzlich dazu konnten nicht alle Probleme des ersten Durchlaufs gelöst werden. Beim Context-Steering muss die Interest-Funktion weiter optimiert werden, damit das bei beiden Durchgängen auftretende Problem gelöst wird.

Im Vergleich zum Context-Steering hat sich die Swarm-Steering-Erfolgsquote von 90/270 im ersten Durchlauf auf 59/270 verschlechtert und damit von 33 % auf 22 %. Dafür verantwortlich sind die neuen Funktionen, die viel stärkere Werte zurück gegeben haben und damit einen viel größeren Einfluss auf die Gesamtkraft hatten. Die Verkleinerung der Gewichte konnte dem nur bedingt entgegenwirken. Swarm-Steering ist sehr parameterabhängig ist und die Parameter wurden nur für ein Szenario in einer Bot-Kombination getestet. Deshalb hat die getestete Kombination außerhalb des Test-S1 sehr schlecht funktioniert.

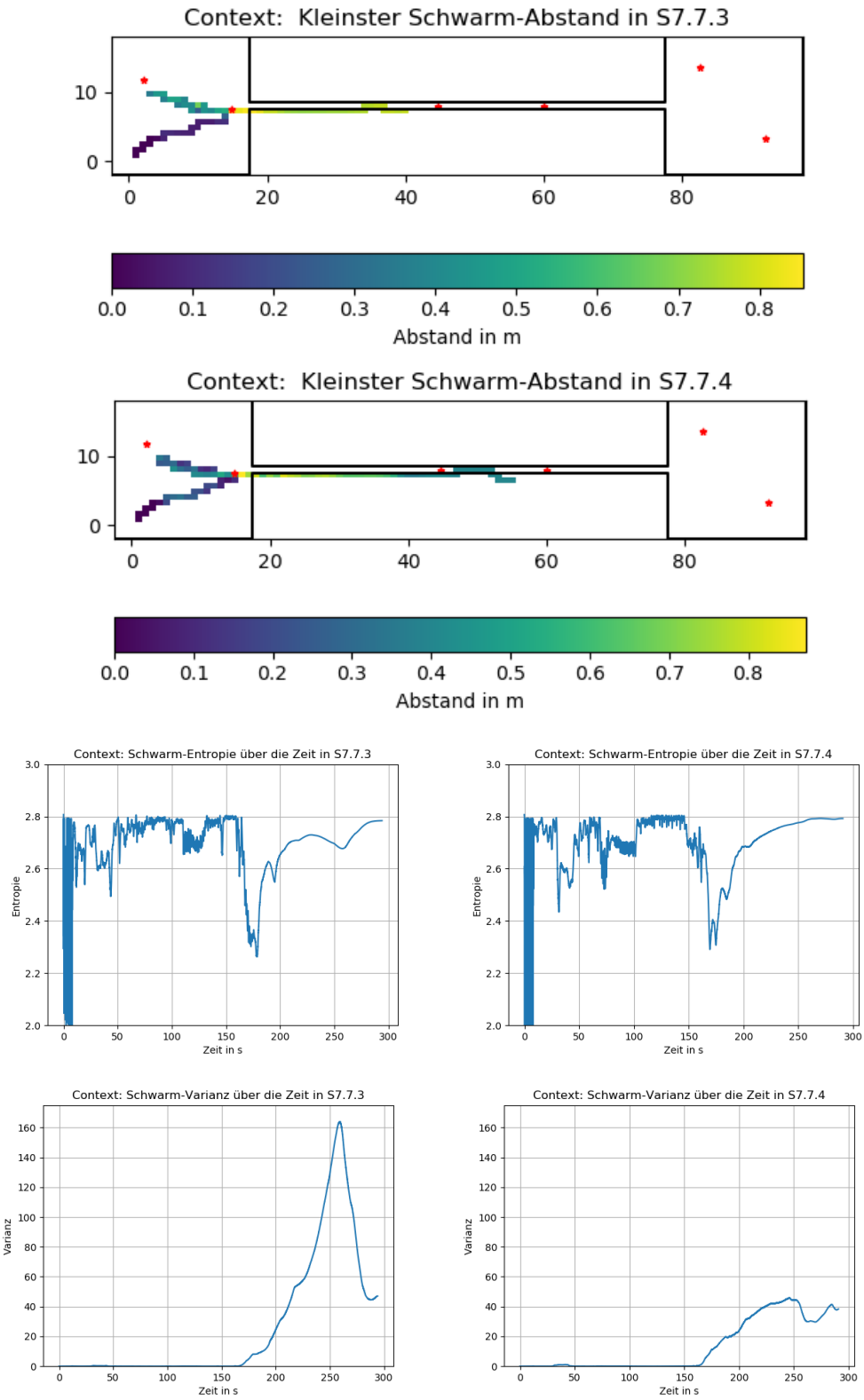


Abbildung 4.41.: Vergleich zwischen S7.7.3 und S7.7.4

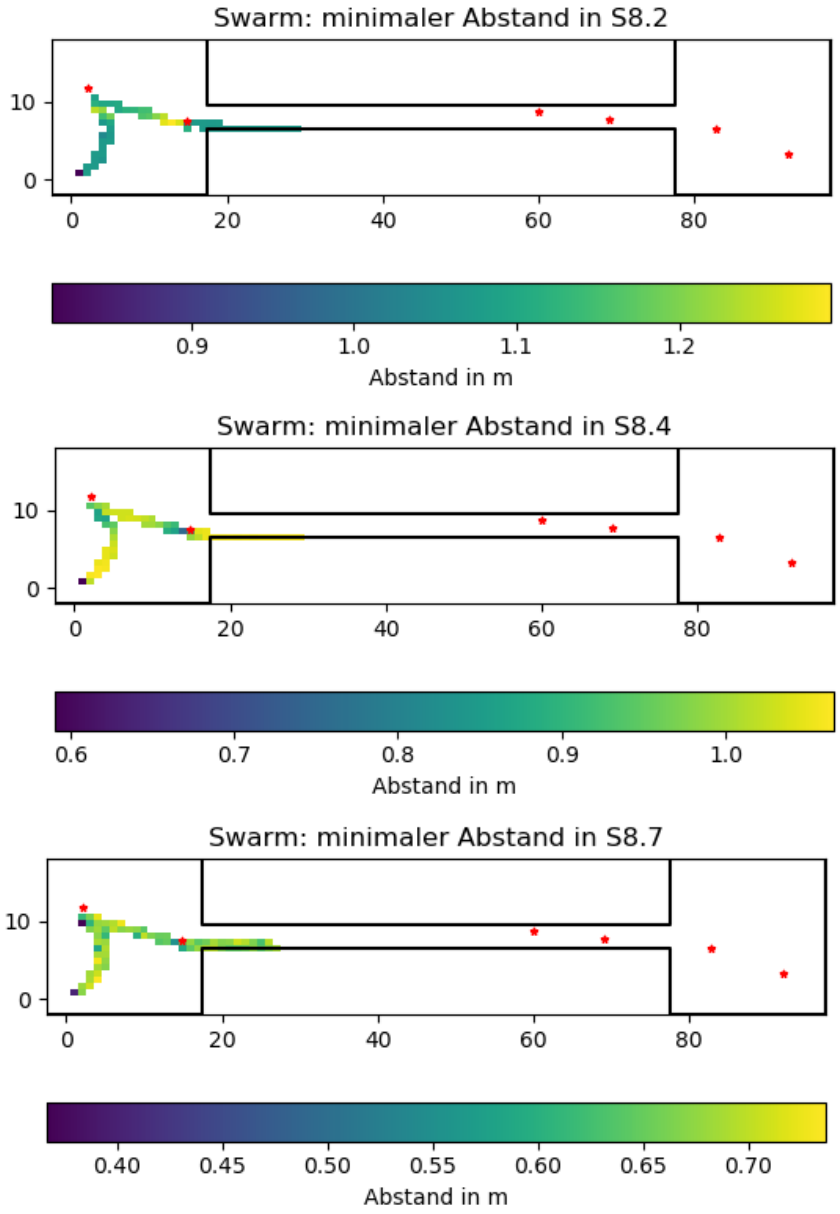


Abbildung 4.42.: Verschiedene Swarm-Steering-HeatMaps für S8

Durch das Abschließen von jeweils zwei Durchläufen pro Swarmbewegungs-Ansatz ist es möglich, Antworten auf die anfangs gestellten Hypothesen zu finden.

Die erste Hypothese, dass Context-Steering eine höhere Erfolgsquote erreicht als Swarm-Steering, konnte innerhalb beider Durchläufe bewiesen werden. Wenn für Swarm-Steering der bessere erste Durchlauf gezählt wird, übertrifft Context-Steering die 90 erfolgreichen Experimente von Attraction/Repulsion Ansatz um 150 weitere Versuche. Aufgrund dessen, dass die gewählten Parametersätze in Context-Steering getestet und auf nur Swarm-Steering übertragen wurden ist das Ergebnis einseitig. Im Idealfall sollte der Swarm-Steering-Ansatz gesondert Parametrisiert werden. Da eine hohe Vergleichbarkeit der beiden Ansätze das Ziel war, wurde dies hier jedoch nicht durchgeführt.

Die zweite Hypothese, dass sieben Bots die Aufgabe schneller abschließen, können kann nicht bestätigt werden. Dadurch, dass die Kopter sehr stark als Schwarm zusammen geflogen sind, waren mehr Bots eher ein Hindernis. Je mehr geflogen sind, desto langsamer hat sich der Schwarm fortbewegt, was dazu geführt hat, dass bei größeren Karten im ersten Durchlauf häufig weniger Interest eingesammelt wurde als bei kleineren Bot-Mengen. Dieses Problem konnte teilweise im zweiten Durchlauf gelöst werden, wodurch sich dich Experimentzeiten der verschiedenen Botmengen eher angeglichen haben. Beim Swarm-Steering konnte die Hypothese auch wiederlegt werden. Anhand von S1-3 und S6 im zweiten Durchlauf war ersichtlich, dass mehr Bots die Experiment-Ergebnisse stark verschlechtern. Vor allem in S6 ist das Ergebnis von zehn erfolgreichen Experimenten auf null gesunken. Der Grund dafür ist die Parameteranpassung im zweiten Durchlauf, die nur auf Experimenten mit zwei Koptern basierte.

Die letzte Hypothese, dass das System stabiler ist sobald weniger Bots existieren kann auch nicht bestätigt werden, da es in beiden Durchläufen keine Kollisionen der Kopter untereinander gab und alle Entropie -und Varianz-Graphen klare erklärbare Verläufe hatten. Die großen Unterschiede in den Graphen konnten auf die verschiedenen Verhaltensweisen, wie I-0 auf dem Rückweg einsammeln aus S5 4.19 oder das Aufteilen des Schwarms 4.41 in S7, zurück geführt werden. Swarm-Steering hatte bis auf einzelne Ausnahmen in S1-3 im ersten Durchlauf gleichmäßige Entropie -und Varianzwerte. Grund dafür war auch die teilweise sehr schlechte Performance beim einsammeln der

IPs. Wenn der Schwarm nie den Tunnel betreten hat, kann er auch keine großen Stabilitätsprobleme bekommen.

5. Zusammenfassung und Ausblick

Ziel der Arbeit war es herauszufinden, wie Context-Steering im Vergleich zu Attraction-Repulsion in der Schwarmrobotik abschneidet. Diese Arbeit zeigt die Vorteile der Verwendung von Context-Steering gegenüber eines Attraction-Repulsion-Systems. Schon im ersten Durchlauf in Kapitel 4.2 konnte in Szenario 2 bestätigt werden, dass Context-Steering sich erfolgreich auf 3D übertragen lässt.

Innerhalb der zwei Experiment-Durchläufe gelang es, geeignete Parameter für beide Ansätze zu finden. Dabei wurde festgestellt, dass Context-Steering weitaus einfacher zu parametrisieren ist. In der Vorbereitung des zweiten Durchlaufs wurden die Interest -und Walldanger-Funktionen geändert. Mit diesen Änderungen hat Context-Steering eine Steigerung der Erfolgsquote von 50 % auf 89 % erzielt.

Im Gegensatz dazu hat die Anpassung der Interest -und Dangerfunktionen beim Swarm-Steering dazu geführt, dass der Schwarm nur noch undefiniertes Verhalten gezeigt hat. Eine Veränderung der Gewichte der beiden Funktionen löste das Problem. Dennoch verschlechterte sich die Erfolgsquote beim Swarm-Steering im zweiten Durchlauf von 33 % auf 22 %. Der Grund dafür ist, dass Swarm-Steering ein viel geschlosseneres System ist, bei dem alle Kräfte ineinander wirken. Wenn ein Teil der Formel geändert wird, sind die Auswirkungen auf das Kräftegleichgewicht enorm und der Ansatz hat wesentlich schlechter funktioniert.

Context-Steering hat dieses Problem mit seiner stark entkoppelten Funktionsweise nicht. Es kann viel leichter mit neuen Context-Maps, Funktionen und anderen Features erweitert werden. Ein gutes Beispiel dafür ist, dass die Implementierung eines Minimalinterests beim Context-Steering sehr einfach war und im Gegensatz dazu, beim Swarm-Steering sehr schwer umzusetzen wäre, da ein Eingriff in das Kräftesystem schnell zu unvorhersehbaren Verhalten

führen kann. Damit ist die zweite Teilfrage beantwortet, eine funktionierende Parametrisierung existiert für beide Ansätze, wobei sie in Context-Steering leichter zu finden ist und eine höhere Erfolgsquote aufweist.

Ein stabiles Schwarmverhalten in dreidimensionalen Szenarien konnte für beide Steering-Ansätze nachgewiesen werden. Während Swarm-Steering mit Ausnahme der kleinsten Karte (S1-3) durchgängig innerhalb der Map sehr gleichmäßige Entropie -und Varianzwerte aufwies, verbesserte sich Context-Steering im zweiten Durchlauf. Die Probleme des ersten Durchlaufs konnten größtenteils mit der neuen Walldanger-Funktion gelöst werden. Dadurch sank die Out-Of-Map-Zeit des größten Schwarms in den meisten Experimenten auf 0 % und in S1.7 von 35 % auf unter 5 %. Zusätzlich wurde das oszillierende Verhalten der Entropie-Varianzwerte über alle Szenarien reduziert.

Die Beantwortung der Teilfragen lässt eine Aussage der Forschungsfrage zu. **Bei ähnlicher Parametrisierung und vergleichbarem Aufbau schneidet Context-Steering um 56% besser ab als Swarm-Steering.** Dieser Prozentsatz ist die Differenz der Erfolgsquoten der beiden besten Durchläufte. Die einfache Handhabung, Integration und Erweiterbarkeit von Context-Steering bietet klare Vorteile gegenüber dem sehr geschlossenem herkömmlichen Attraction-Repulsion-Kräfte-System.

Jedoch wurden in dieser Arbeit verschiedene Parameter nicht ausführlich betrachtet. Zukünftige Tests könnten stärker auf die Geschwindigkeit der Schwärme achten beziehungsweise diese als weiteren Parameter untersuchen. Damit lassen sich Rückschlüssen auf die Effizienz des Abfliegens der Interest-Punkte und auf die Stabilität des Schwarms schließen. Die Generierung der IPs kann so angepasst werden, dass zum Beispiel Punkte nicht mehr im Tunnel erscheinen können, wodurch das Finden des Tunneleingangs stark erschwert werden würde. Es kann auch getestet werden, wie sich der Schwarm verhält, falls IPs außerhalb der Karte generiert werden würden. Hierbei müsste beim Swarm-Steering-Schwärme enorme Probleme aufweisen und sich höchstwahrscheinlich sich in einen Deadlock begeben.

Anhand der beiden Durchläufe war die Effektivität von optimalen Interest-/Dangerfunktionen sehr gut sichtbar. Diese können in Zukunft weiter angepasst werden, wobei die Danger-Funktion beim Swarm-Steering definitiv auf Grundlage der Funktion aus dem ersten Durchlauf angepasst werden sollte. Da Context-Steering die Grundlage für die Tests der geeigneten Parameter war, kann in Zukunft getestet werden, inwiefern der Ansatz performed, wenn die

Parameter auf Swarm-Steering ausgelegt und dann in Context-Steering übertragen werden. Falls die Parametrisierung sich als zu schwierig herausstellt, ist auch zudem möglich, eine geeignete Belegung mithilfe von evolutionären Algorithmen zu finden. Diese Verfahren wurde bereits in den Context-Steering-Grundlagen 2.3 angesprochen und in Paper [9] genauer erläutert.

Ein weiterer wichtiger Punkt ist, dass anhand der Hardware-Performance der Kopter auch die Auflösung der Context-Map verändert werden kann, um bessere Ergebnisse zu erzielen. Falls sie ein zu stark begrenzender Faktor ist, sollte in zukünftigen Experimenten das im Context-Steering-Kapitel 2.3 besprochene SubSlot-Movement eingebaut werden. Dies sind nur einige der Änderungen oder Anregungen die für zukünftige Experimente umgesetzt werden könnten.

A. Anhang

Wegpunkt	Position(ENU) in Szenario		
	1	2	3
I0	-1/1.81/5.2	-1/1.81/5.2	-1/1.81/5.2
I1	1.65/1.12/4.58	1.65/1.12/4.58	1.65/1.12/4.58
I2	4.56/0.5/7.14	4.56/1.09/7.14	4.56/1.09/7.14
I3	9.64/0.5/2.42	4.17/0.33/8.24	4.17/0.33/8.24
I4	21.5/-0.29/6.96	19.18/0.48/7.02	19.18/0.48/7.02
I5	19.13/2.29/8.08	21.5/-0.29/6.96	21.5/-0.29/6.96
	4	5	6
I0	0.01/5.12/5.2	0.01/5.12/5.2	0.01/5.12/5.2
I1	6.4/3.03/7.36	6.4/3.03/7.36	6.4/3.03/7.36
I2	29.03/3/0.83	29.03/3.77/0.83	29.03/3.03/0.83
I3	21.28/3/2.42	33.54/2.66/0.63	33.54/2.66/0.63
I4	45/0.93/6.96	40.37/2.47/7.02	40.37/2.47/7.02
I5	40.26/6.09/8.08	45/0.93/6.96	45/0.93/6.96
	7	8	9
I0	2.23/11.75/5.2	2.23/11.75/5.2	2.23/11.75/5.2
I1	14.8/7.57/7.36	14.8/7.57/7.36	14.8/7.57/7.36
I2	60.06/8/0.83	60.06/8.77/0.83	60.06/9.54/0.83
I3	44.59/8/2.42	69.06/7.83/0.63	69.09/7.66/0.63
I4	92.01/3.37/6.96	82.75/6.45/7.02	82.75/6.45/7.02
I5	82.53/13.68/8.08	92.01/3.37/6.96	92.01/3.37/6.96

Tabelle A.1.: Wegpunktpositionen für jedes Szenario

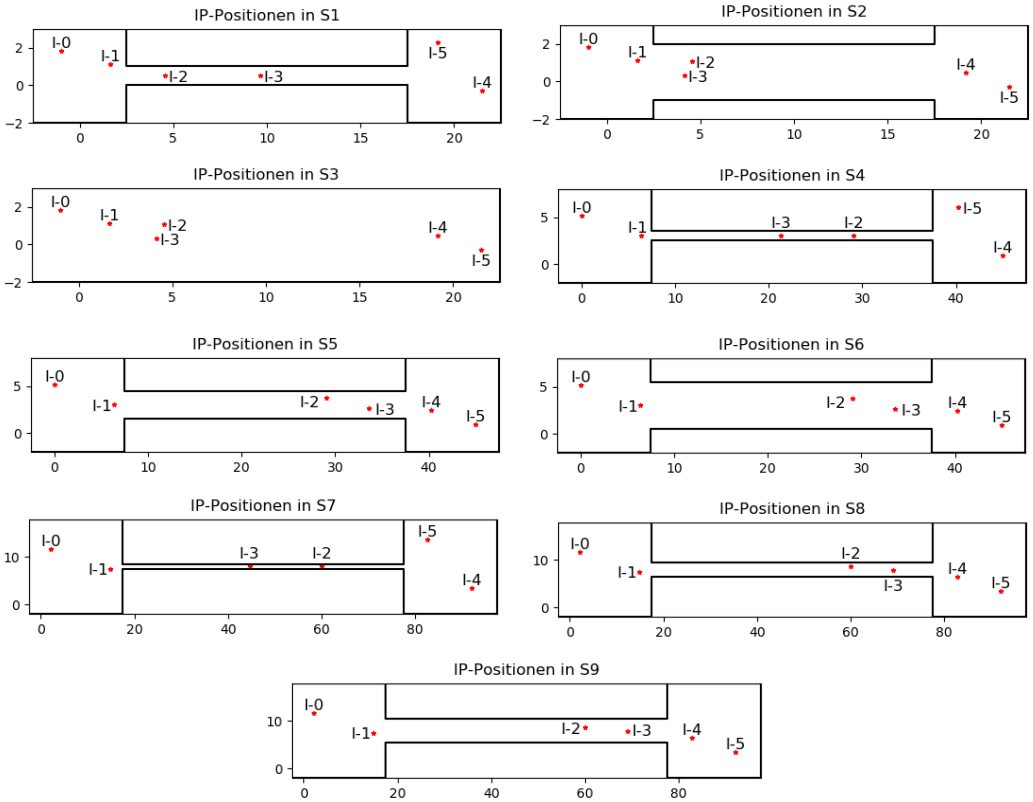


Abbildung A.1.: Kartenaufbau mit generierten Interest-Punkten

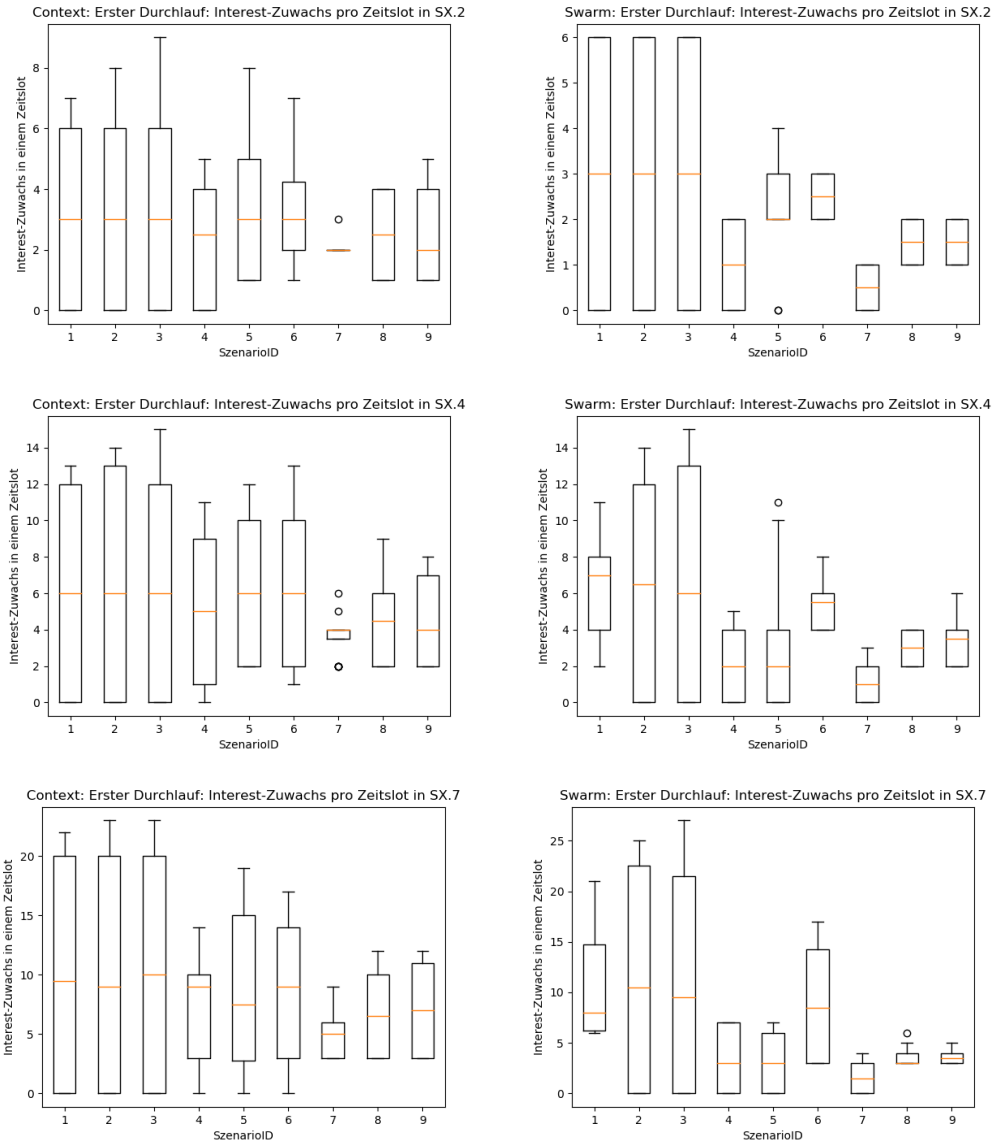


Abbildung A.2.: Eingesammelte IPs pro Zeitslot im ersten Durchlauf

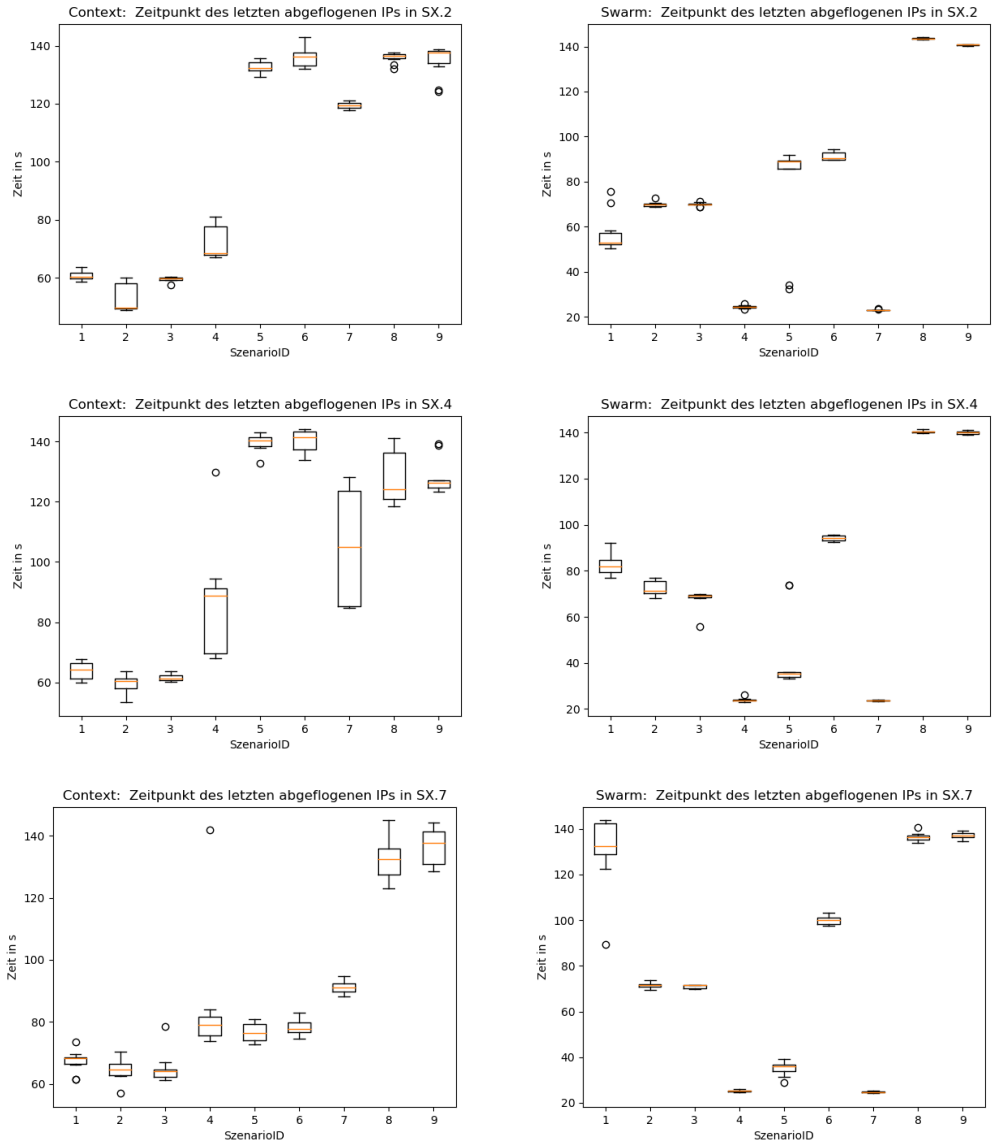


Abbildung A.3.: Letzter Zeitpunkt der Interest-Erhöhung im ersten Durchlauf

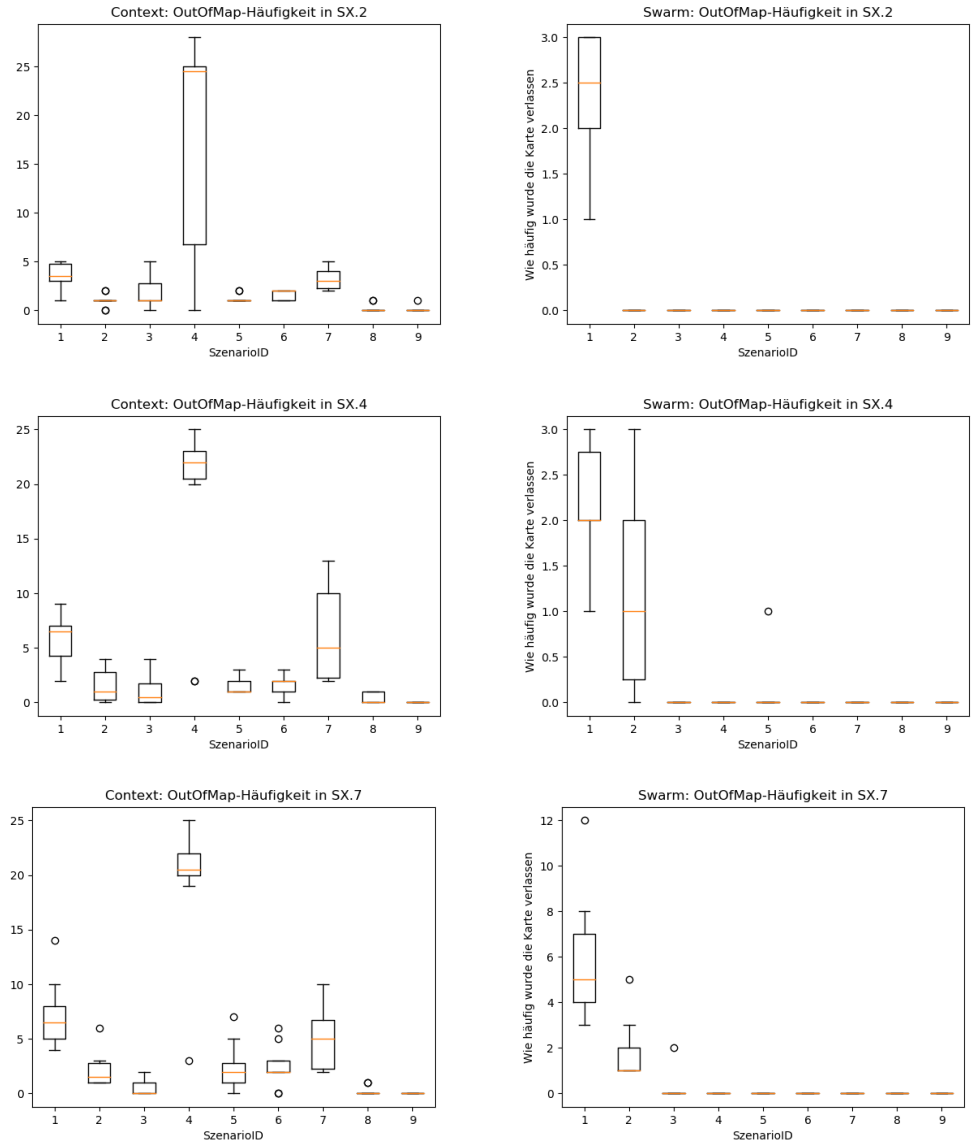


Abbildung A.4.: Wie oft verlässt der Kopter die Map im ersten Durchlauf

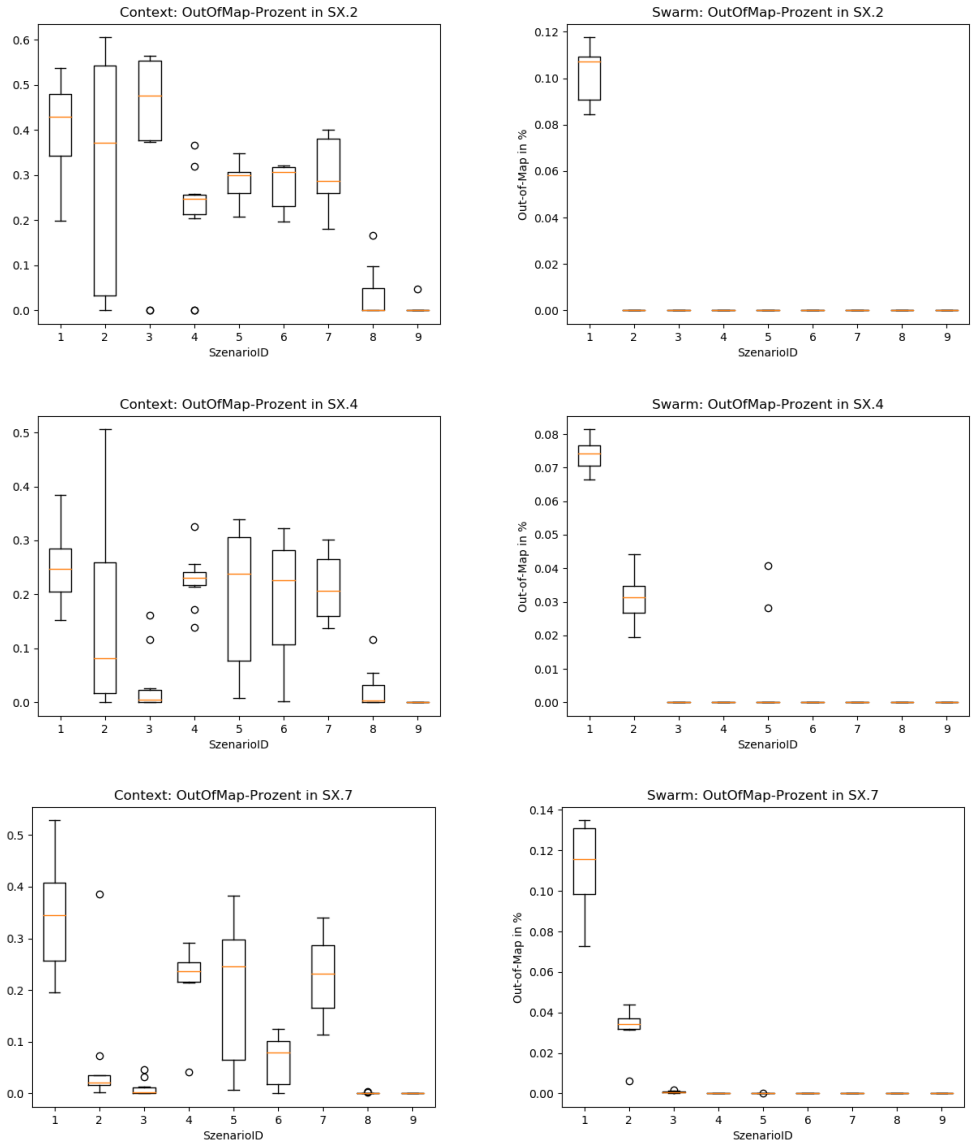


Abbildung A.5.: Out-Of-Map Zeit (in %) pro Szenario im ersten Durchlauf

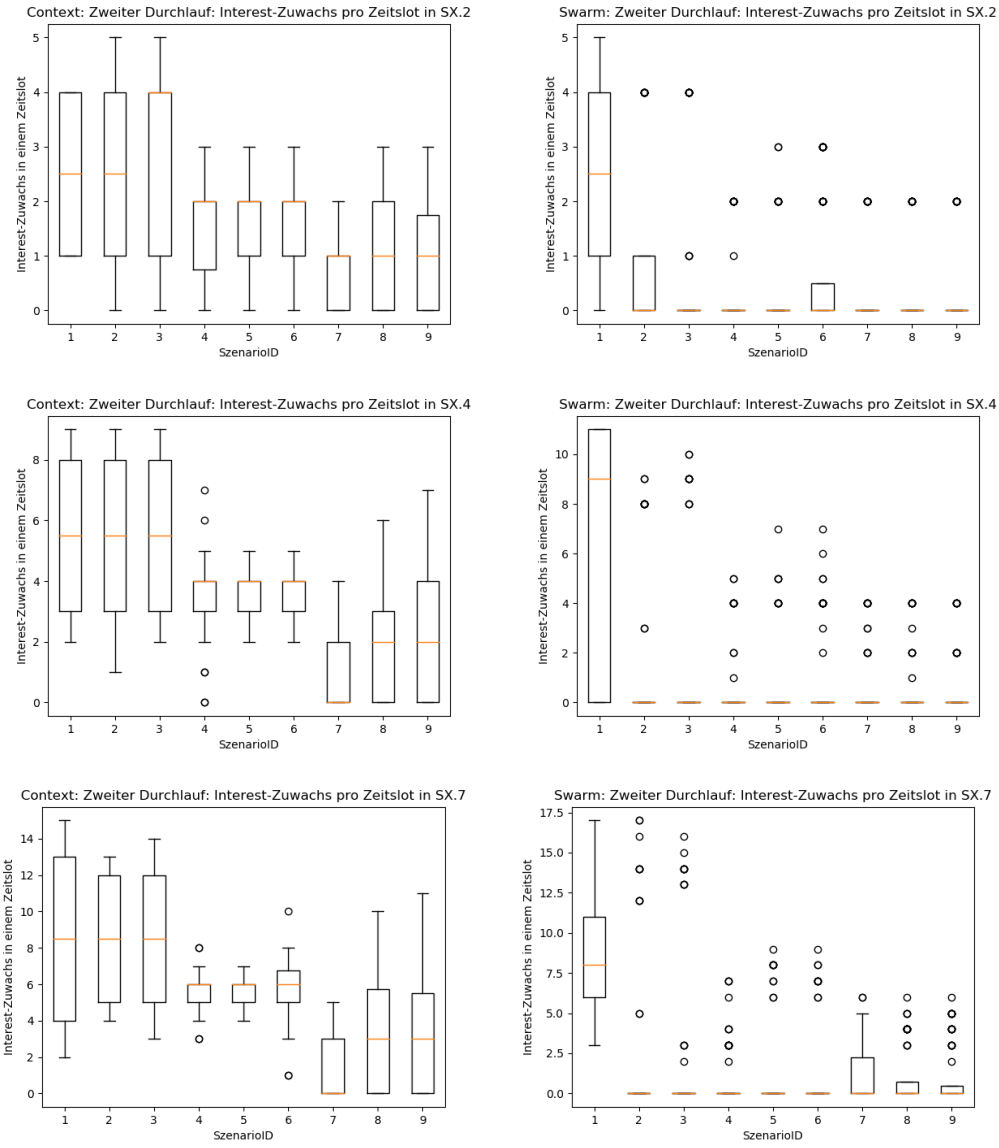


Abbildung A.6.: Eingesammelte IPs pro Zeitslot im zweiten Durchlauf

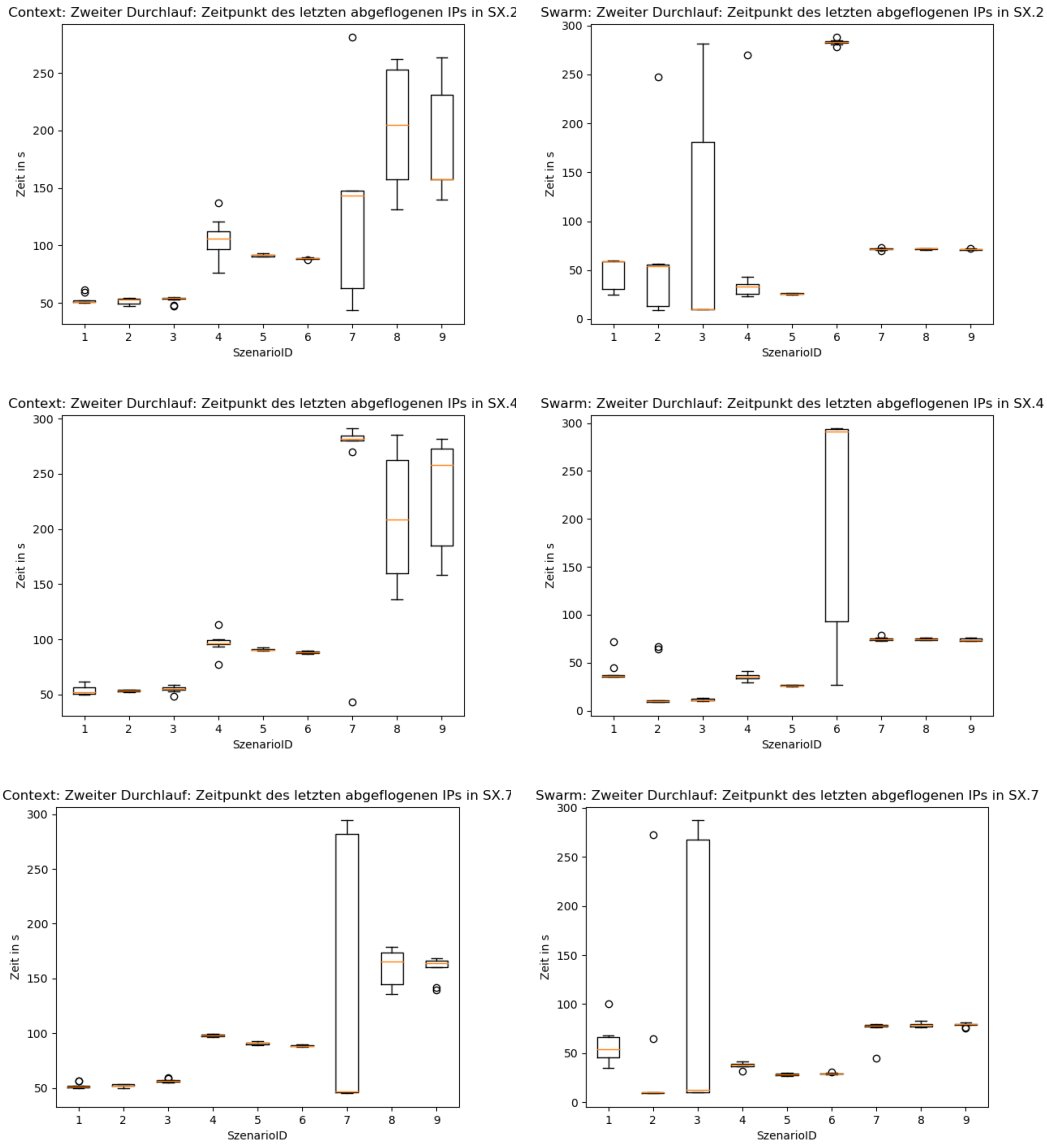


Abbildung A.7.: Letzter Zeitpunkt der Interest-Erhöhung im zweiten Durchlauf

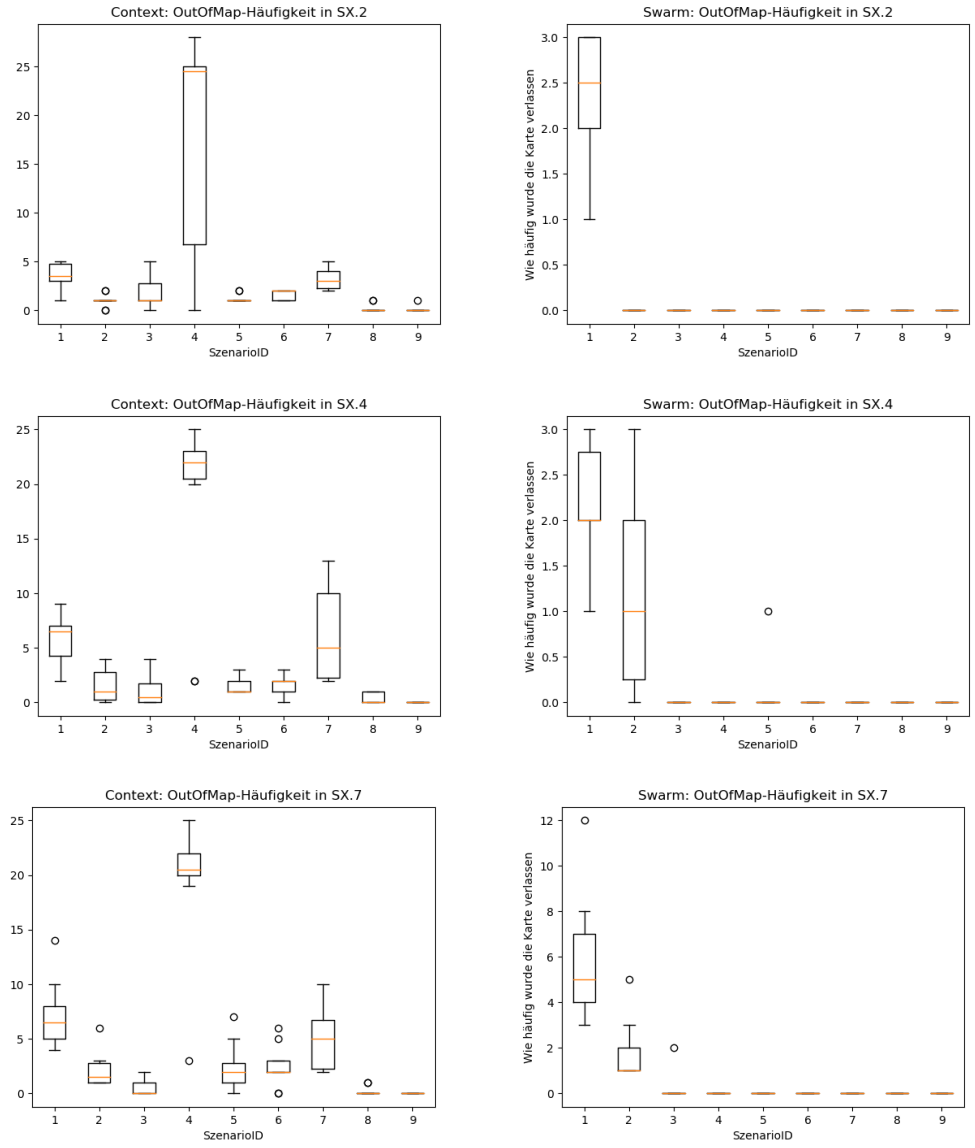


Abbildung A.8.: Wie oft verlässt der Kopter die Map im zweiten Durchlauf

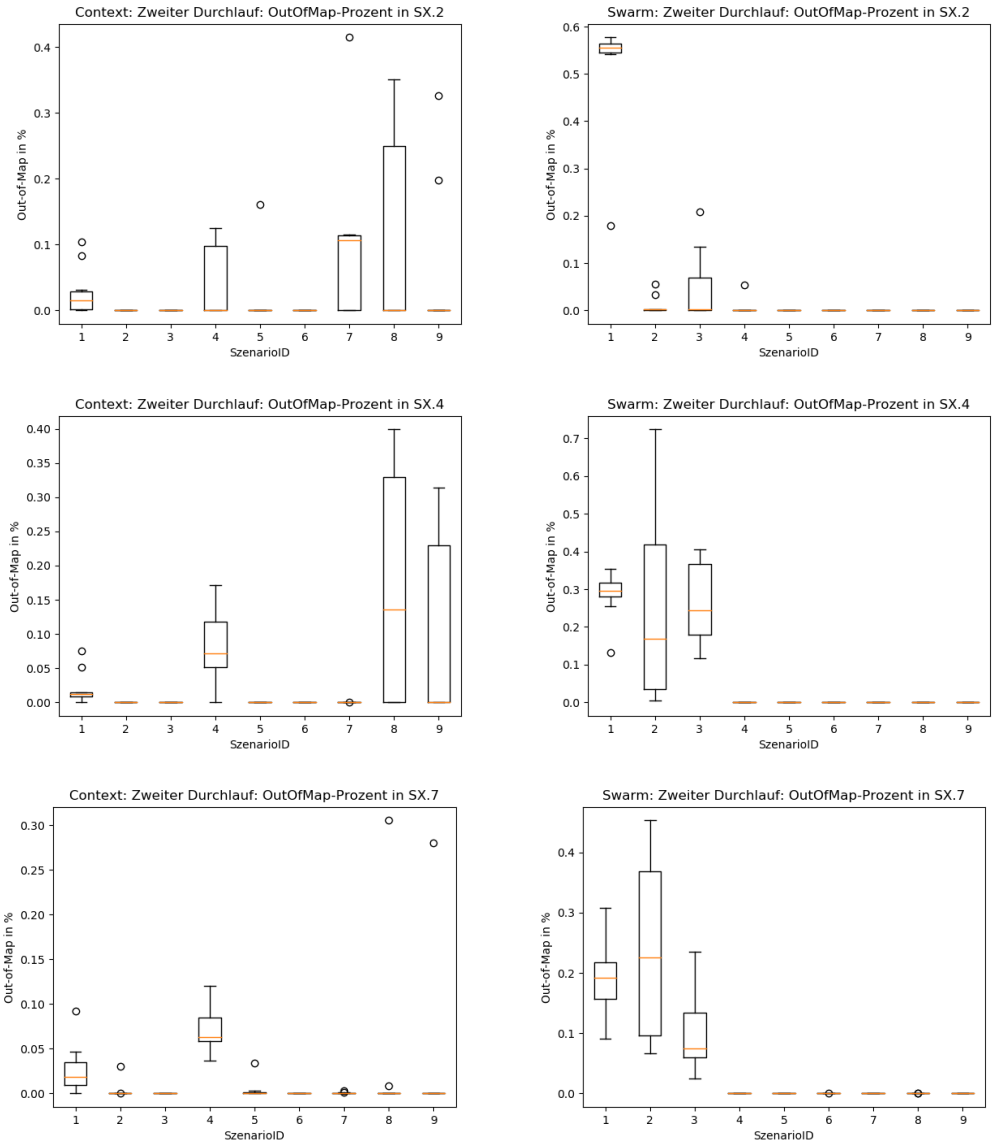


Abbildung A.9.: Out-Of-Map Zeit (in %) pro Szenario im zweiten Durchlauf

Literaturverzeichnis

[1] Paparazzi-overview. `https://wiki.paparazziuav.org/wiki/Overview#System_Architecture`, Abrufdatum 17.07.2023.

[2] The paparazzi project. `https://wiki.paparazziuav.org/wiki/General`, Abrufdatum 17.07.2023.

[3] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In Paolo Dario, Giulio Sandini, and Patrick Aebischer, editors, *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[4] Jon Berndt. Jsbsim: An open source flight dynamics model in c++. In *AIAA Modeling and Simulation Technologies Conference and Exhibit*, page 4923, 2004.

[5] Marc-Andrė Blais and Moulay A. Akhloufi. Reinforcement learning for swarm robotics: An overview of applications, algorithms and simulators. *Cognitive Robotics*, 3:226–256, 2023.

[6] Sanaz Mostaghim Christoph Steup, Sebastian Mai. Evaluation platform for micro aerial - indoor swarm robotics. Magdeburg, 2016. Otto von Guericke Universität: Fakultät für Informatik.

[7] Xiao Cui and Hao Shi. An overview of pathfinding in navigation mesh. *International Journal of Computer Science and Network Security*, 12(12):48–51, December 2012.

[8] Kalyanmoy Deb. *Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction*, pages 3–34. Springer London, London, 2011.

[9] Alexander Dockhorn, Martin Kirst, Sanaz Mostaghim, Martin Wieczorek, and Heiner Zille. Evolutionary algorithm for parameter optimization of context-steering agents. *IEEE Transactions on Games*, 15(1):26–35, 2023.

[10] Alexander Dockhorn, Sanaz Mostaghim, Martin Kirst, and Martin Zettwitz. Multi-objective optimization and decision-making in context steering. In *2021 IEEE Conference on Games (CoG)*, pages 1–8, 2021.

[11] Andrew Fray. Context steering: behavior driven steering at the macro scale. *Game AI Pro 2: Collected Wisdom of Game AI Professionals*, page 183–193., 2015.

[12] Martin Kirst. Multi-criteria optimized context steering for autonomous movement in games, 2015.

[13] M. Anwar Ma'sum, Grafika Jati, M. Kholid Arrofi, Adi Wibowo, Petrus Mursanto, and Wisnu Jatmiko. Autonomous quadcopter swarm robots for object localization and tracking. In *MHS2013*, pages 1–6, 2013.

[14] R.V. Meshcheryakov, P.M. Trefilov, A.V. Chekhov, S.A.K Diane, K.D. Rusakov, E.A. Lesiv, M.A. Kolodochka, K.O. Shchukin, A.K. Novoselskiy, and E. Goncharova. An application of swarm of quadcopters for searching operations. *IFAC-PapersOnLine*, 52(25):14–18, 2019. 19th IFAC Conference on Technology, Culture and International Stability TECIS 2019.

[15] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Springer New York, NY, 30 September 1998.

[16] Prof. Dr. Sanaz Mostaghim. Swarm intelligence: Swarm aggregatiion, folien, 2022.

[17] James A. Preiss, Wolfgang Honig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304, 2017.

[18] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, page 25–34, New York, NY, USA, 1987. Association for Computing Machinery.

[19] Carlos A. Toro-Arcila, Héctor M. Becerra, and Gustavo Arechavaleta. Visual path following with obstacle avoidance for quadcopters in indoor environments. *Control Engineering Practice*, 135:105493, 2023.

[20] Vivek Shankar Varadharajan and Giovanni Beltrame. *Get Together! Multi-robot Systems: Bio-Inspired Concepts and Deployment Challenges*, pages 299–332. Springer Nature Singapore, Singapore, 2022.

[21] Lars Wagner, Christopher Olson, and Alexander Dockhorn. Generalizations of steering - a modular design. In *2022 IEEE Conference on Games (CoG)*, pages 580–583, 2022.

[22] Christian Wustrau. Building a scalable swarm application using sphero robots, 2020.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Philipp Thoms

Magdeburg, 26.09.2023