OTTO-VON-GUERICKE UNIVERSITÄT MAGDEBURG

FAKULTÄT FÜR INFORMATIK
INSTITUT FÜR INTELLIGENTE KOOPERIERENDE SYSTEME

MASTER THESIS

Multikriteriell optimiertes Context Steering für autonome Bewegung im Gebiet der Schwarmrobotik

Autor:
Andrei STEIN

Betreuer:
Prof. Dr. Sanaz MOSTAGHIM
Dipl.-Inform. Christoph STEUP

25. Januar 2018

Selbstständigkeitserklärung

Ich erkläre hiermit, die vorliegende Arbeit selbständig und nur mit erlaubten Hilfsmitteln angefertigt zu haben. Die verwendeten Quellen und Hilfsmittel sind im Text kenntlich gemacht und im Quellenverzeichnis vollständig aufgeführt. Das gesamte Material, welches von Internet-Quellen entnommen wurde, basiert auf dem Stand vom 25. Januar 2018.

Magdeburg, 25. Januar 2018 *Andrei Stein*

Abstrakt

Die vorliegende Arbeit stellt eine Bewegungssteuerung für einen autonomen Quadrokopter vor. Als Grundlage wird dazu, das aus der Spieleindustrie stammende Konzept des Context Steering genutzt. Dabei wird auf die Einbeziehung der Sensoren des Quadrokopters eingegangen. Das Konzept des Context Steering wird erweitert, um zusätzlich ein Schwarmverhalten von mehreren Quadrokopter zu ermöglichen. Die resultierende Bewegungssteuerung wird in einer Simulationsumgebung evaluiert. Das Ergebnis ist eine Bewegungssteuerung, die dem Quadrokopter ein autonomes Fliegen in einer ihm unbekannten Umgebung ermöglicht. Am Ende der Arbeit werden die Probleme und mögliche Lösungsansätze diskutiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Idee für diese Arbeit entstand am SwarmLab der Otto-von-Guericke Universität Magdeburg. Das SwarmLab hat sich das Ziel gesetzt, die theoretischen Konzepte aus dem Bereich der Schwarmintelligenz in praktischen Anwendungen zu nutzen. Dazu wurde der Finken Quadrokopter von der Arbeitsgruppe entwickelt. Der Finken stellt eine Plattform dar, mit der sich verschiedene Konzepte ausprobieren und verwirklichen lassen. Ein wichtiges Konzept stellt dabei das autonome Fliegen dar. Ein stabiles Flugverhalten ist eine der Voraussetzungen für viele weitere Konzepte der Schwarmintelligenz. Aus diesem Grund beschäftigt sich diese Arbeit mit einer Bewegungssteuerung für den Finken. Dabei baut diese Arbeit auf einem Ansatz aus der Spieleindustrie auf. Das sogenannte Context Steering wurde bereits von Kirst in seiner Master Thesis untersucht [1]. Das Ziel dieser Arbeit ist es, den Ansatz des Context Steering auf den Finken anzuwenden. Dazu muss untersucht werden, wie die Sensoren des Finken für das Context Steering genutzt werden können. Zusätzlich soll ein Schwarmverhalten ermöglicht werden. Für diesen Zweck muss das Grundkonzept angepasst und erweitert werden. Zum Abschluss der Arbeit soll die Bewegungssteuerung evaluiert und auf ihre Eignung analysiert werden.

1.2 Stand der Forschung

Es gibt viele Arbeiten, die sich mit autonom fliegenden Quadrokoptern beschäftigen. Dabei lassen sich die Arbeiten in zwei Gruppen unterteilen. Zu der ersten Gruppe gehören Arbeiten, die sich auf eine möglichst akkurate Bewegungssteuerung konzentrieren. Diese benutzen jedoch häufig ein externes Trackingsystem zur Positionsbestimmung [2, 3]. Somit sind nur Szenarien unter Laborbedingungen möglich. Für den Außenbereich gibt es bereits sehr fortschrittliche Lösungen im kommerziellen Bereich, wie zum Beispiel den Phantom 4 Quadrokopter der Firma DJI [4]. Dieser und ähnliche Quadrokopter nutzen eine Kombination aus GPS-Daten und Kamera-Bildinformationen zur Positionsbestimmung [5, 6].

Zur Zweiten Gruppe gehören Arbeiten, die sich mit einer Bewegungssteuerun beschäftigen, die ohne jegliche Art von externer Positionsbestimmung auskommt und das Fliegen in einer unbekannten Umgebung als Ziel haben. Zu diesem Zweck wird die Umgebung durch Sensoren abgetastet. Die in dieser Arbeit vorgestellte Bewegungssteuerung gehört ebenso zu dieser Gruppe. Es gibt verschiedene Ansätze in diesem Bereich. Die Abtastung der Umgebung kann zum Beispiel mit Kameras erfolgen [6–8]. Andere Arbeiten nutzen einen Laser zum Abtasten der Umgebung [9, 10]. Da der Finken jedoch keinen Lasersensor oder Kamera besitzt, sind diese Methoden nicht für den Finken geeignet. Der Finken besitzt Ultraschallsensoren zum Abtasten der Umgebung. Hierzu gibt es ebenfalls Arbeiten, die Ultraschallsensoren benutzen [11].

Das Abtasten der Umgebung ist nur ein Aspekt einer Bewegungssteuerung. Ein weiterer Aspekt ist die Steuerung der Bewegung an sich. Viele Arbeiten nutzen eine Art von Trajektorienplanung zu diesem Zweck [12, 13]. Dabei wird eine optimale Flugkurve berechnet und der Quadrokopter wird so gesteuert, dass er sich entlang der Flugkurve bewegt. In dieser Arbeit wird ein Ansatz verfolgt, der aus der Spielindustrie stammt. In Computerspielen wird ebenfalls eine Bewegungssteuerung benötigt, um die vom Computer gesteuerten Akteure in der Spielwelt zu bewegen. Ein weitverbreiteter Ansatz ist hierfür das Konzept der sogenannten Steering Behavior [14, 15]. Die Steering Behavior bewirken ein bestimmtes Bewegungsverhalten und führen bei der Kombination mit einander, zu einem komplexen Verhalten des Akteurs. Außerdem sind Steering Behavior einfach zu implementieren und benötigen nur wenig Rechenleistung. Ein zu den Steering Behavior erweitertes Konzept, stellt das Context Steering dar [16]. Im Unterschied zu den Steering Behavior, werden beim Context Steering die Umgebungsinformationen

zur Bewegungssteuerung berücksichtigt. Das Context Steering wurde bereits in der Arbeit von Kirst untersucht [1]. Dessen Erkenntnisse bilden die Grundlage für diese Arbeit.

1.3 Gliederung der Arbeit

Die Arbeit ist in sechs Kapitel unterteilt. In diesem Kapitel wurden bereits die Motivation und Ziele dieser Arbeit genannt. Zudem wurde der aktuelle Stand der Forschung zu diesem Thema vorgestellt.

In Kapitel 2 werden die theoretischen Hintergründe zu diesem Thema erläutert. Dabei werden sowohl das Konzept des Context Steering, als auch die Methoden zur Auswahl einer Bewegungsrichtung besprochen.

Die entwickelte Bewegungssteuerung wird anschließend in Kapitel 3 vorgestellt. Dabei werden zuerst die Anforderungen an solch eine Bewegungssteuerung und die Spezifikationen des verwendeten Quadrokopters besprochen. Des Weiteren, werden die verschieden Methoden zur Auswahl einer Bewegungsrichtung diskutiert und am Ende des Kapitel wird der Ansatz im Detail erläutert.

Kapitel 4 handelt von der Implementierung der Bewegungssteuerung in einer Simulationsumgebung. Dabei wird zuerst die Simulationsumgebung V-REP und anschließend das virtuelle Modell des Finken vorgestellt.

In Kapitel 5 wird die Bewegungssteuerung evaluiert. Die Testumgebung und die einzelnen Szenarien werden dabei vorgestellt und die Ergebnisse der Testläufe präsentiert. Die Ergebnisse werden anschließend diskutiert.

Im letzten Kapitel werden die erreichten Ziele zusammengefasst und es werden Möglichkeiten für zukünftige Arbeiten diskutiert.

Kapitel 2

Theoretischer Hintergrund

Dieses Kapitel beschäftigt sich mit den theoretischen Hintergründen dieser Arbeit. Der vorgestellte Lösungsansatz für das Context basierte Schwarmverhalten hat die beiden Konzepte des Context Steerings und der multikriteriellen Entscheidungsfindung als Grundlage. Dabei stellt das Context Steering das Grundkonzept für eine Bewegungssteuerung autonomer Agenten dar. Die Konzepte der multikriteriellen Entscheidungsfindung finden innerhalb des Context Steering ihre Anwendung.

Die Kombination beider Ansätze bildet die Grundlage für eine Vielzahl von Erweiterungen. Das im nächsten Kapitel vorgestellte Context basierte Schwarmverhalten ist solch eine Erweiterung.

2.1 Context Steering

Context Steering ist ein alternatives Konzept zur Bewegungssteuerung, welches einige Probleme der Steering Behaviors umgeht. Ebenso wie die Steering Behaviors, kommt es aus dem Bereich der Spieleentwicklung und wurde im Jahr 2015 von Andrew Fray in einem Buch über Künstliche Intelligenz in Computerspielen vorgestellt [16].

Die Grundidee des Context Steerings ist es, zuerst Informationen über die Umgebung zu sammeln, um anschließend die nächste Bewegungsrichtung unter Berück-

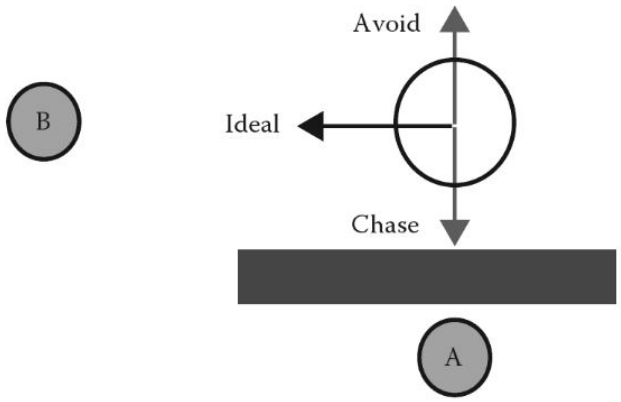

Abbildung 2.1: Die von den Steering Behaviors resultierenden Richtungen und die eigentlich ideale Richtung. [16]

sichtigung der Umgebungsinformationen zu wählen. Damit unterscheidet sich dieser Ansatz zu den Steering Behaviors, bei den für jedes Behavior eine separate Bewegungsrichtung bestimmt und mit den anderen Richtungen zu einer resultierenden Bewegungsrichtung vereint wird.

Abbildung 2.1 zeigt ein Beispiel an dem der Vorteil des Context Steerings gegenüber den Steering Behaviors deutlich wird. In dem Beispiel ist ein Agent zu sehen, der die beiden Steering Behaviors Avoid und Chase besitzt. Beide Behaviors liefern eine Bewegungsrichtung zurück. Das Chase-Behavior liefert eine Richtung, die zum Ziel mit der kürzesten Entfernung zeigt. In diesem Fall ist es Ziel A. Das Avoid-Behavior liefert eine Richtung zurück, die von Hindernissen weg zeigt. In dem Beispiel stellt das Rechteck ein Hindernis dar, welches dem Agenten den Weg zu Ziel A versperrt.

Die Kombination beider Richtungsvektoren führt zu einem nicht idealen Ergebnis. Der Agent kann sich entweder in Richtung des Ziels A bewegen und damit auch in Richtung der Wand, oder er kann sich von der Wand weg und somit auch weg vom Ziel A bewegen. Eine Bewegungssteuerung, die ausschließlich mit den beiden Steering Behaviors arbeitet, ist nicht in der Lage, die ideale Lösung in Richtung Ziel B zu liefern. In diesem Fall wäre es besser, wenn das Seek Behavior merken würde, dass der Weg zum kürzesten Ziel durch ein Hindernis blockiert

ist und stattdessen ein Vektor in Richtung des weiter entfernten Ziel B als Ergebnis zurückgeben würde. Jedes Behavior erfüllt eine bestimme Aufgabe und berücksichtigt nicht die Informationen anderer Behavior.

Genau an diesem Umstand setzt das Konzept des Context Steerings an. Zuerst werden Informationen über die aktuelle Umgebung geholt und in Context Maps gespeichert, anschließend wird unter Berücksichtigung der Umgebungsinformationen eine Bewegungsrichtung ausgewählt. Die beiden folgenden Kapitel erläutern die beiden Schritte ausführlich.

2.1.1 Context Maps

Context Maps speichern Informationen über die umlegende Umgebung. Für das Beispiel in Abbildung 2.1 erstellt Fray insgesamt zwei Context Maps [16]. Die Umgebung in dem Beispiel wird rund um den Agenten in zwölf unterschiedliche Richtungen abgetastet. Die Anzahl der Abtastrichtungen kann je nach Anwendung variiert werden. Abbildung 2.2 zeigt die Interest-Map. Diese speichert einen Wert für jede Abtastrichtung, der das Interesse repräsentiert.

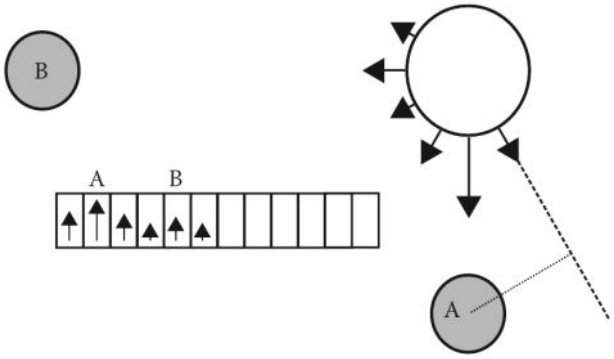

Abbildung 2.2: Die Interest-Map mit den Werten $I = f_i(x)$, die das Interesse in jede der zwölf Abtastrichtungen x beschreibt $|I| = 12$. [16]

Dabei hat eine Richtung ein hohes Interesse, wenn sie zu einem der Ziele *A* oder *B* zeigt und ein niedriges Interesse, wenn sie von einem Ziel weg zeigt. Der Werte-

bereich von $f_i(x)$ und die Funktion selbst kann individuell je nach Anwendungsfall bestimmt werden. Abbildung 2.3 zeigt die zweite Context-Map, die Fray bei seiner vorgestellten Methode nutzt. Es handelt sich hierbei um die sogenannte Danger-Map. Die Danger-Map speichert die Intensität der Gefahr $f_d(x)$ für jede Abtastrichtung. Zur Berechnung dieser Gefahrintensität eignet sich der Abstand zu den als gefährlich eingestuften Objekten.

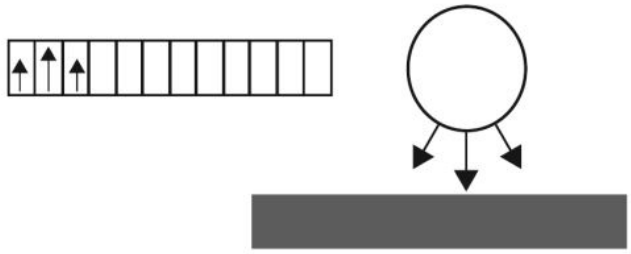

Abbildung 2.3: Die Danger-Map mit den Werten $D = f_d(x)$ beschreibt die Gefahr in jede der zwölf Abtastrichtungen $|D| = 12$. [16]

Nachdem die Context-Maps erzeugt und mit Werten gefüllt sind, kann eine Bewegungsrichtung ausgewählt werden. Für die Auswahl der Bewegungsrichtung gibt es keine vorgegebene Methode, daher bleibt viel Spielraum für Ansätze und zukünftige Forschung in diesem Bereich. Eine sehr einfache Methode, die auch von Fray in seinem Beispiel benutzt wird, soll hier zunächst erklärt werden um einen Einstieg in das Thema zu geben.

Fray schlägt folgendes, als einen einfachen Ansatz vor (siehe Abbildung 2.4). Zuerst wird ein Filter auf der Danger-Map erstellt *(i)*. Es wird der kleineste Gefahrenwert ausgewählt und alle Richtungen mit einem höheren Gefahrenwert werden verworfen. In diesem Fall ist der kleinste Gefahrenwert $D_{min} = 0$. Aus diesem Grund werden alle Abtastrichtungen die eine Gefahr erkannt haben aus dem Filter ausgeschlossen.

Im zweiten Schritt *(ii)* wird der Filter auf die Interest-Map angewandt. Damit bleiben in diesem Fall nur Richtungen in der Interest-Map übrig, bei denen keine Gefahr in der Danger-Map erkannt wurde. Aus den verbleibenden Richtungen aus der Interest-Map wird die Richtung mit dem höchsten Interessenwert ausgewählt *(iii)*. Im letzten Schritt *(iv)* wird der ausgewählte Richtungsvektor als Geschwin-

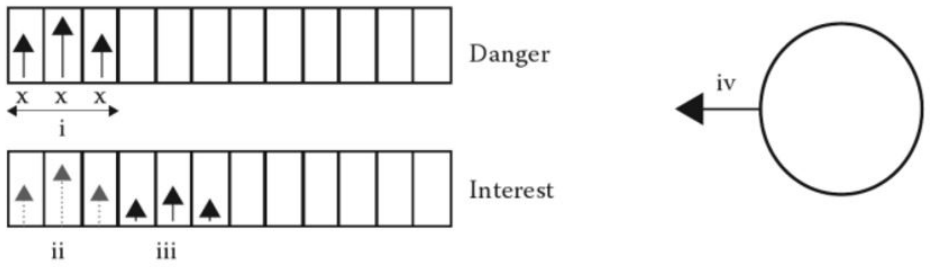

Abbildung 2.4: Schema eines einfachen Ansatzes zur Auswahl eine Bewegungsrichtung nach Fray. [16]

digkeitsvektor für die nächste Bewegung des Agenten benutzt. Dabei hängt die Höhe der Geschwindigkeit von dem Interessenwert aus der Interest-Map ab. Die ausgewählte Bewegungsrichtung stimmt mit der idealen Richtung aus Abbildung 2.1 überein. Damit wird einer der Vorteile des Context Steerings gegenüber der Steering Behavior gezeigt.

2.1.2 360° Bewegung

Das im letzten Kapitel vorgestellte Beispiel (siehe Abbildung 2.1) lässt die Vermutung zu, die resultierende Bewegung des Agenten könnte sehr eingeschränkt sein, da nur eine begrenzte Anzahl an Bewegungsrichtungen zur Auswahl stehen. Ein Agent der sich nur in wenige Richtungen bewegen kann, erscheint für den Betrachter eher künstlich oder roboterhaft. Diese eingeschränkte Bewegung eignet sich deshalb nicht für Agenten, die ein lebendiges Lebewesen simulieren sollen.

Eine mögliche Lösung besteht darin, die Anzahl der Abtastrichtungen und somit die Anzahl an möglichen Bewegungsrichtungen zu erhöhen. Dies führt jedoch zu einer höheren Berechnungsintensität. Fray stellt in seiner Arbeit eine alternative Lösung vor, die eine Bewegung in alle Richtungen erlaubt, ohne dabei die Anzahl der Abtastrichtungen zu erhöhen.

Das Prinzip wird in Abbildung 2.5 veranschaulicht. Nachdem eine Bewegungsrichtung mit dem höchsten Interessenwert aus der Interest-Map ausgewählt wird,

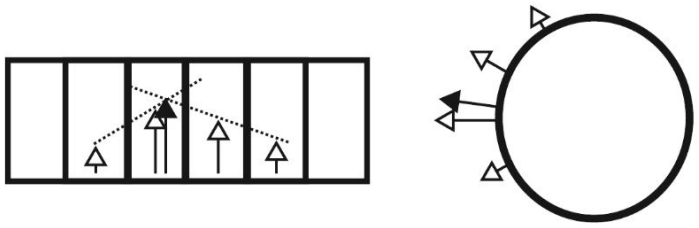

Abbildung 2.5: 360° Bewegung zwischen den Abtastrichtungen durch Gradienten Bildung und Interpolation nach Fray. [16]

werden die benachbarten Werte für eine Gradienten Bildung genutzt. Anschließend wird der Schnittpunkt der beiden Gradienten bestimmt und die dazugehörige Bewegungsrichtung berechnet. Die neue Bewegungsrichtung besitzt mindestens einen genauso hohen Interessenwert wie die zuvor ausgewählte Richtung und befindet sich zwischen den Abtastrichtungen. Somit ist eine Bewegung in 360° um den Agenten möglich und der Agent wirkt für den Betrachter lebendiger.

2.1.3 Filterung

Eine Filterung der Werte in den Context Maps kann in einigen Fällen erwünscht sein, um Ausreißer zu verhindern und die Werte insgesamt zu glätten. Ein solcher Fall wird in Abbildung 2.6 dargestellt. In der Abbildung ist ein Agent zu sehen, der sich in der Nähe zweier Wände befindet. Es wird in dem Beispiel angenommen, dass die Gefahrenobjekte mit Hilfe von Strahlen detektiert und als Gefahrenwerte in die Danger-Map geschrieben werden. Die enge Lücke zwischen den Wänden wird dabei als Gefahrenfrei eingestuft, da eine Detektion der zu engen Lücke mit den Abtaststrahlen nicht erfolgt ist *(i)*. Dies kann dazu führen, dass der Agent in Richtung der Lücke fliegt und sich so in unnötige Gefahr begibt.

In diesem Fall kann eine nachträgliche Filterung der Werte in der Danger-Map das Problem umgehen. Dazu wird eine diskrete Faltung [17] mit einem geeigneten Filterkern durchgeführt:

$$(f_d * g)(x) = \sum_{k=-\infty}^{\infty} f_d(k) g(x-k) \tag{2.1}$$

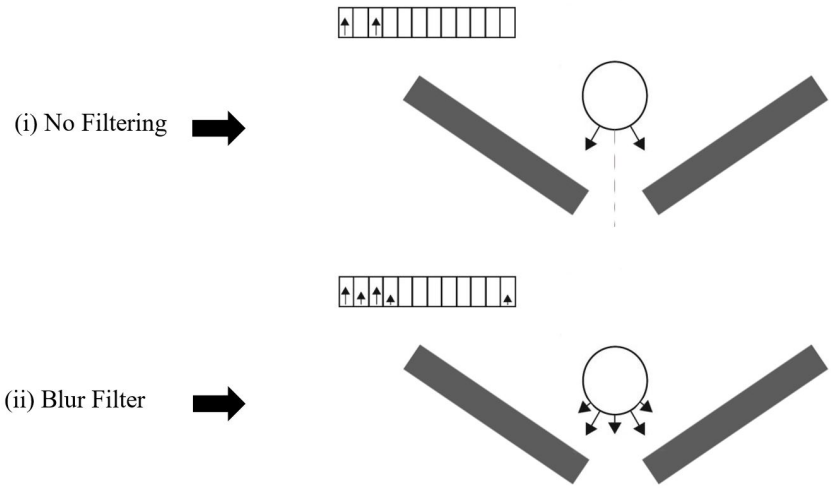

Abbildung 2.6: *(i)* Danger-Map ohne Filterung. *(ii)* Danger-Map mit einem Gaußschen Unschärfe Filter.

Als Filterkern eignet sich ein Unschärfefilter. Dieser verwischt die Werte in der Danger-Map, sodass eingetragene Gefahrenwerte teilweise auch auf die benachbarten Felder übertragen werden:

$$g(x) = \begin{bmatrix} 0.125 \\ 0.750 \\ 0.125 \end{bmatrix} \tag{2.2}$$

Die resultierende Danger-Map ist in Abbildung 2.6 *(ii)* zu sehen. Mit der gefilterten Danger-Map erkennt der Agent die gefährliche Lücke und versucht erst gar nicht sich dort hin zu bewegen. Im gezeigten Beispiel wird nur die Danger-Map gefiltert. Eine Filterung ist aber auch für die Interest-Map und jede andere Context-Map möglich, um bestimmte Ergebnisse zu erzielen.

2.1.4 History Blending

In realen Szenarien mit zum Beispiel einen fliegenden Quadrokopter, werden Hindernisse häufig mit Abstandssensoren erfasst. Die Sensoren liefern in der Regel nicht immer korrekte Werte, sodass es gelegentlich zu Ausreißern kommen kann.

Diese Ausreißer spiegeln sich dann ebenfalls in der Danger-Map wieder. Dies kann zu falschen Entscheidungen des Agenten führen.

Fray stellt in seinen Ausführungen auch für diese Problematik einen Lösungsansatz vor. Bei diesem Ansatz werden die Werte $f_{d/i}^{t-1}(x)$ der zuletzt verwendeten Context-Maps gespeichert und mit den aktuellen Werten $f_{d/i}^{t}(x)$ gemischt. Das Vermischen kann durch die folgende Operation durchgeführt werden:

$$(f_{d/i}^{t-1}(x) + f_{d/i}^{t}(x))\alpha(x) = \alpha f_{d/i}^{t-1}(x) + (1-\alpha)\alpha f_{d/i}^{t}(x) \tag{2.3}$$

Der Parameter $\alpha \in [0, 1]$ bestimmt dabei, welcher der beiden Werte dominanter gewichtet wird. Ein hoher Wert für α bewirkt eine langsamere Veränderung der Werte über die Zeit gesehen, als bei einem niedrigen α Wert. Dadurch wirken sich einzeln auftretende Ausreißer nicht so stark auf die Auswahl der nächsten Bewegungsrichtung aus.

2.2 Multikriterielle Entscheidungsfindung für pareto-optimale Bewegung

Im letzten Kapitel 2.1 wurde die Erzeugung der Kontext Maps erläutert. Die Context Maps enthalten Informationen über die lokale Umgebung des Agenten. Diese Informationen sind jeweils mit einem Richtungsvektor verknüpft. Für die Bewegung des Agenten, muss in jedem Iterationsschritt einer der Richtungsvektoren ausgewählt werden. Dazu werden in diesem Kapitel einige Ansätze vorgestellt [18]. Die Ausführungen basieren auf der Master Thesis von Martin Kirst [1]. Kirst nutzt in seiner Arbeit Methoden aus dem Bereich der Multikriteriellen-Optimierung (MCO) für die Auswahl der Bewegungsrichtung.

Die Interest- und Danger-Map aus dem Beispiel in Abbildung 2.4 können zusammen in einem Graphen abgebildet werden. Abbildung 2.7 zeigt so einen Graphen. Jede mögliche Bewegungsrichtung besitzt einen Gefahren- und Interessenwert. Der Agent soll im optimalen Fall in die Richtung fliegen, in der die Gefahr möglichst gering und das Interesse möglichst hoch ist. Daraus folgt ein Minimie-

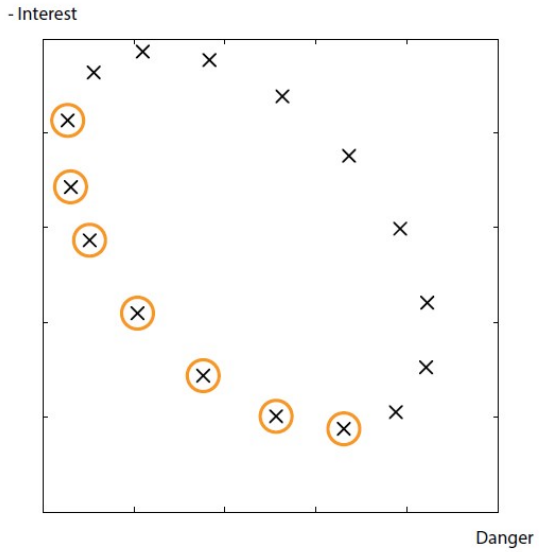

Abbildung 2.7: Beispiel für ein diskretes MCO Problem. Die Kreuzte stellen die Abtastrichtungen mit den dazugehörigen Werten $-f_i(x)$ und $f_d(x)$ aus den beiden Context Maps dar. Die mit einem Kreis markierten Richtungen bilden die Pareto-Front. [1]

rungsproblem, bei dem eine Bewegungsrichtung x bestimmt wird:

$$\begin{aligned} &minimiere \;\; \{f_d(x), -f_i(x)\} \\ &in \;\; Abhängigkeit \;\; von \;\; x \in 1, 2, ..., r \end{aligned} \tag{2.4}$$

Die Funktionswerte $f_i(x)$ für das Interesse müssen dazu negiert werden. In den folgenden Kapiteln werden drei Methoden vorgestellt, die Kirst in seiner Arbeit benutzt um eine Bewegungsrichtung zu bestimmen.

2.2.1 Weighting Method

Bei dieser Methode werden die Funktionen $f_i(x)$ und $f_d(x)$ gewichtet und miteinander zu Z_{d+i} summiert [19,20]. Dadurch entsteht das folgende Minimierungsproblem:

$$\begin{aligned} &minimiere \;\; w_d f_d(x) - w_i f_i(x) \\ &in \;\; Abhängigkeit \;\; von \;\; x \in 1, 2, ..., r \end{aligned} \tag{2.5}$$

Für die Gewichte gilt $w_d, w_i \in [0, 1]$ und $w_d + w_i = 1$. Durch die Summenbildung zu einem Wert, findet die Optimierung nur noch in einer Dimension statt. Die Bewegungscharakteristik des Agenten wird durch die Gewichte beeinflusst. Ein Agent mit einer hohen Gewichtung w_d wählt eher Richtungen aus, in dem die Gefahr gering ist. Dadurch bevorzugt der Agent offene Bereiche in der Umgebung und meidet enge Gassen und andere gefährliche Situationen. Im Gegensatz dazu bewegt sich ein Agent mutiger und bevorzugt Richtungen mit hohem Interesse, wenn die Gewichtung w_i hoch ist. Algorithmus 2.1 beschreibt die Weighting Method mit einem Pseudocode, wobei D und I die beiden Context Maps darstellen:

Algorithm 2.1 Weighting method in pseudocode [1]

```
1: procedure WeightingDecision(D, I, w_d, w_i)
2:     Z_{d+1} ← array of size |D|          ▷ Combined objective function
3:     m ← ∞                                ▷ Index for best found minimum
4:     for k = 1, ..., |D| do               ▷ |D| is context map resolution
5:         Z_{d+1}[k] ← w_d D[k] − w_i I[k]
6:         if m = ∞ or Z_{d+1}[k] < Z_{d+1}[m] then
7:             m ← ∞
8:     return m
```

2.2.2 Constraint Method

Das Konzept der Constraint Method oder auch ε-Constraint Method genannt, wurde 1971 von Haimes et al. vorgestellt [21]. Bei diesem Konzept wird eine Funktion zum Optimieren ausgewählt. Für die restlichen Funktionen wird jeweils eine Obergrenze als ε-Bedingung festgelegt. In dem Beispiel aus Abbildung 2.4 wird wie zuvor $f_i(x)$ optimiert. Für $D = f_d(x)$ wird eine Bedingung $\varepsilon_d \in [0, 1]$

festgelegt. Dies hat zur Folge, dass nur Richtungen für die Auswahl berücksichtigt werden, dessen Gefahrenwert unterhalb ε_d liegt. Aus den übrig gebliebenen Richtungen wird dann die Richtung mit dem höchsten Interesse $f_i(x)$ ausgewählt. Sollte es keine Richtung geben, die einen Gefahrenwert unterhalb ε_d besitzt, wird die Richtung ausgewählt, die die ε-Bedingung am wenigsten verletzt (siehe Algorithmus 2.2). Da es sich um ein Minimierungsproblem handelt, wird die negierte Interessen-Funktion $-I = -f_i(x)$ verwendet. Das Minimierungsproblem wird folgenderweise formuliert:

$$\begin{aligned} & minimiere \;\; -f_i(x) \\ in \;\; Abhängigkeit \;\; von \;\; & f_d(x) \leq \varepsilon_d, \\ & x \in 1, 2, ..., r \end{aligned} \tag{2.6}$$

Die ε-Bedingung ε_d beeinflusst die Bewegungscharakteristik des Agenten ähnlich zu den Gewichten der Weighting Mehtod aus dem vorherigen Kapitel 2.2.1. Mit der ε-Bedingung lässt sich das Bewegungsverhalten jedoch direkter steuern, da sich im Gegensatz zur Weighting Mehtod gezielt Richtungen mit zu hohen Gefahrenwert ausschließen lassen. Ein niedriger Grenzwert ε_d lässt nur Richtungen mit geringer Gefahr zur Auswahl zu und führt somit zu einem vorsichtigen Verhalten des Agenten. Bei einem hohen Grenzwert ε_d können dagegen auch gefährliche Richtungen vom Agenten ausgewählt werden.

Algorithm 2.2 Constraint method in pseudocode [1]

```
1:  procedure CONSTRAINTDECISION(D, I, ε_d)
2:      m ← ∞                                   ▷ Index for best found minimum
3:      for k = 1, ..., |D| do                  ▷ |D| is context map resolution
4:          if D[k] ≤ ε_d then
5:              if m = ∞ or −I[k] < −I[m] then
6:                  m ← k
7:      if m = ∞ then                           ▷ If no sample satifies constraint
8:          m ← 1
9:          for k = 2, ..., |D| do
10:             if D[k] < D[m] or (D[k] = D[m] and −I[k] < −I[m]) then
11:                 m ← k
12:     return m
```

2.2.3 Hybrid Method

Die Hybrid Method [22, 23] kombiniert die beiden Ansätze aus den vorherigen Kapitel 2.2.1 und Kapitel 2.2.2. Für die Gefahrenfunktion $D = f_d(x)$ wird eine Obergrenze ε_d als ε-Bedingung festgelegt. Wie schon bei der Constraint Method aus Kapitel 2.2.2, werden zuerst alle Richtungen ausgeschlossen, dessen Gefahrenwert $f_d(x)$ höher ist als die Bedingung ε_d. Auf den übrig gebliebenen Richtungen wird die Weighting Method aus Kapitel 2.2.1 angewendet. Es wird eine gewichtete Summe gebildet $Z = w_d f_d(x) - w_i f_i(x)$, wobei gilt $w_i + w_d = 1$. Anschließend wird die Richtung mit der kleinsten Summe zur Bewegung ausgewählt. Sollte es keine Richtung geben, die die ε-Bedingung erfüllt, so wird die Richtung mit dem kleinsten Gefahrenwert $f_d(x)$ ausgewählt. Der genaue Algorithmus wird in Form eines Pseudocodes in Algorithmus 2.3 beschrieben. Das dazugehörige Minimierungsproblem wird beschrieben durch:

$$\begin{aligned} minimiere \;\; & w_d f_d(x) - w_i f_i(x) \\ in \;\; Abhängigkeit \;\; von \;\; & f_d(x) \leq \varepsilon_d, \\ & x \in 1, 2, ..., r \end{aligned} \tag{2.7}$$

Die Hybrid Method ist sehr flexibel anwendbar. Wird die ε-Bedingung auf $\varepsilon_d = 0$ gesetzt, so verhält sich die Methode wie die Weighting Method, da alle Richtungen die ε-Bedingung erfüllen. Wird stattdessen $w_i = 1$ gesetzt, so verhält sich die Methode wie die Constraint Method, da die Richtung nach $f_i(x)$ augewählt wird. Die ε-Bedingung stellt eine harte Bedingung dar, mit der sich Richtungen ausschließen lassen, wohingegen die Gewichte w_d, w_i eine Präferenz für bestimmte Richtungen definieren.

Algorithm 2.3 Hybrid method in pseudocode [1]

```
1:  procedure HybridDecision(D, I, w_d, w_i, ε_d)
2:      Z_{d+1} ← array of size |D|                ▷ Combined objective function
3:      m ← ∞                                      ▷ Index for best found minimum
4:      for k = 1, ..., |D| do                     ▷ |D| is context map resolution
5:          Z_{d+1}[k] ← w_d D[k] − w_i I[k]
6:          if D[k] ≤ ε_d then
7:              if m = ∞ or Z_{d+i}[k] < Z_{d+i}[m]) then
8:                  m ← k
9:      if m = ∞ then                              ▷ If no sample satifies constraint
10:         m ← 1
11:         for k = 2, ..., |D| do
12:             if D[k] < D[m] or (D[k] = D[m] and Z_{d+i}[k] < Z_{d+i}[m]) then
13:                 m ← k
14:     return m
```

Kapitel 3

Context Steering basiertes Schwarmverhalten

In diesem Kapitel wird die neu entwickelte Methode zur Bewegungssteuerung eines Quadrokopterschwarms mit Hilfe von Context Maps vorgestellt. Dazu werden zuerst in Kapitel 3.1 die Anforderungen an die Bewegungssteuerung genannt. Anschließend wird in Kapitel 3.2 eine geeignete Methode zur Steuerung der Quadrokopter bestimmt. Diese wird in Kapitel 3.3 ausführlich erläutert. Die verwendeten Context Maps werden in Kapitel 3.4 vorgestellt.

3.1 Anforderungen

Vor der Entwicklung einer Bewegungssteuerung, müssen zuerst alle Anforderungen definiert werden. In diesem Abschnitt werden die Anforderungen beschrieben, die als Grundlage für die vorgestellte Methode zur Bewegungssteuerung dienen. Die Anforderungen ergeben sich aus zwei Bereichen. Zum einen ergeben sich Anforderungen durch die ausgewählte Umgebung und dem Szenario in der sich der Agent bewegen soll, zum anderen werden Anforderungen durch den Agenten selbst und seinen Fähigkeiten, an eine Bewegungssteuerung gestellt. Die Bewegungssteuerung eines Fahrzeuges hat zum Beispiel andere Anforderungen als die eines Flugzeuges. Die Bewegung des Fahrzeuges findet in zwei Dimensionen (Geschwindigkeit und Richtung) statt, wohingegen beim Flugzeug vier Dimension (Geschwindigkeit, Pitch, Roll und Yaw) gesteuert werden. Ebenso verhält es sich

Abbildung 3.1: 3. Generation des Finken Quadrokopters. [25]

mit der Umgebung, in der sich der Agent bewegt. Draußen im Freien kann GPS[1] zur Positionsbestimmung genutzt werden. Im Innenraumbereich wie zum Beispiel in Gebäuden, ist das GPS-Signal jedoch zu schwach [24] und somit müssen andere Methoden zur Positionsbestimmung genutzt werden oder sogar komplett drauf verzichtet werden.

In dieser Arbeit ist der Agent ein fliegender Quadrokopter namens Finken (siehe Abbildung 3.1). Dieser wird im Swarm Lab am Lehnstuhl für intelligente Systeme der Otto-von-Guericke Universität Magdeburg entwickelt. Zum Zeitpunkt dieser Arbeit wurde der Finken-III bereits zweimal weiterentwickelt und liegt in der dritten Generation vor. Die folgende Liste zeigt Spezifikationen des Finken, die für die Bewegungssteuerung wichtig sind [25]:

- **4 Rotoren:**
  Ermöglicht die Steuerung in den vier Dimensionen Höhe, Yaw und x/y Geschwindigkeit.
- **Optical flow Sensor mit Sonar-Höhenmessung:**
  Misst die aktuelle Höhe und die Geschwindigkeit in der x/y-Ebene.
- **Turm mit 4 Sonar Entfernungssensoren:**
  Misst Entfernung zu in der Nähe liegenden Objekten.
- **802.15.4 basiertes Phasendifferenz-Entfernungsmesser:**
  Misst Entfernungen zu anderen Finken und Knotenpunkten.

[1] GPS - Global Positioning System (deutsch: Globales Positionsbestimmungssystem)

- **802.15.4 Kommunikationsmodul:**
  Ermöglicht die Kommunikation mit anderen Finken und einer Basisstation.

Im folgenden werden die Anforderungen an die Bewegungssteuerung des Finken erläutert. Der Finken soll sich sowohl im Freien, als auch im Innenraum bewegen können. Das Bewegen im Innenraum führt dazu, dass auf eine globale Positionsbestimmung, wie etwa durch GPS, verzichtet werden muss. Die Umgebung ist dem Finken unbekannt. Aus diesem Grund muss der Finken in der Lage sein, Gefahren wie etwa Wände zu erkennen und diesen auszuweichen. Außerdem soll der Finken in der Lage sein Ziele in der Umgebung aufzusuchen. Dabei soll sich ein Finken mit anderen Finken in der nahen Umgebung zu einem Schwarm formieren können.

In den folgenden Kapiteln wird eine Methode zur Bewegungsteuerung beschrieben, die die zuvor genannten Anforderungen mit Hilfe der Sensoren des Finken erfüllt.

3.2 Auswahl einer Methode zur Bewegungssteuerung

Nachdem im letzten Kapitel 3.1 die Anforderungen und die Spezifikationen des Finken Quadrokopters vorgestellt wurden, soll in diesem Kapitel ein geeigneter Ansatz diskutiert werden. Als Grundlage für die Bewegungssteuerung dient in dieser Arbeit das Konzept des Kontext Steerings, welches in Kapitel 2.1 beschrieben ist. Das Context Steering benutzt eine Danger- und eine Interest-Map, um Informationen über Gefahren und interessanten Objekten in der lokalen Umgebung zu sammeln. Für die vorgestellte Bewegungssteuerung wird das Konzept um weitere Context Maps erweitert, um allen Anforderungen aus dem letzten Kapitel gerecht zu werden. Die einzelnen Context Maps und deren Funktion werden im nächsten Kapitel (Kapitel 3.4) ausführlich beschrieben. An dieser Stelle soll zunächst eine geeignete Methode zur Auswahl der Bewegungsrichtung aus Kapitel 2.2 diskutiert werden.

Die Weighting Method aus Kapitel 2.2.1 benutzt eine gewichtete Summe als Kriterium zur Auswahl einer Bewegungsrichtung. Durch die einzelnen Gewichte lassen sich Präferenzen für bestimmte Richtungen steuern. Dabei kann ein und

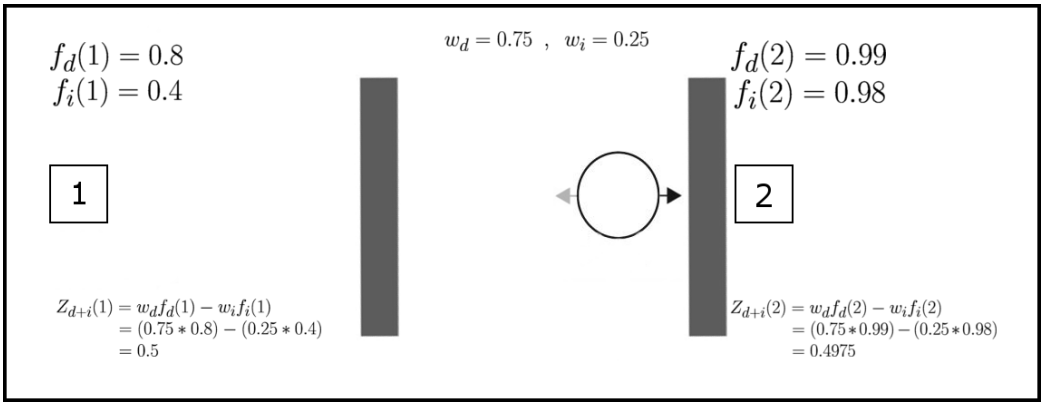

Abbildung 3.2: Ein Szenario bei dem die Weighting Method eine ungewollte Richtung bevorzugt. Ziel 2 wird angesteuert, obwohl die Gefahr in diese Richtung sehr hoch ist.

die selbe Summe durch verschiedene Kombinationen entstehen. Dies führt in bestimmten Situation zu einem Problem bei der Steuerung des Finken.

Ein Extrembeispiel für solch ein Problem wird in Abbildung 3.2 gezeigt. Die Abbildung zeigt schematisch einen Finken, der sich zwischen zwei Zielen befindet. Die Ziele sind als Quadrate mit einer Zahl im Inneren dargestellt. Ziel 1 ist weiter vom Finken entfernt, als Ziel 2. Aus diesem Grund hat die Richtung, die zu Ziel 2 zeigt, einen höheren Interessenwert $f_i(2) = 0.98$. Aber auch der Gefahrenwert in diese Richtung ist durch die nahe stehende Wand sehr hoch $f_d(2) = 0.99$. In dieser Situation sollte der Finken Ziel 1 bevorzugen. Wird jedoch die gewichtete Summe der beiden Richtungen gebildet und miteinander verglichen, so sind beide Summen fast identisch. Die Richtung Ziel 2 wird sogar bevorzugt, weil die Summe in diese Richtung $Z_{d+i}(2) = 0.4975$ kleiner ist als die Summe in Richtung Ziel 1 $Z_{d+i}(2) = 0.5$.

Dieses Beispiel zeigt, dass trotz der Präferenz durch die Gewichte $w_d = 0.75, w_i = 0.25$ zu einem eher vorsichtigen Finken, die Weighting Method in bestimmten Situationen eher gefährliche Richtungen bevorzugen kann. Damit ist die Weighting Method nicht zur Bewegungssteuerung des Finken geeignet. Die Gefahr einer Kollision mit anderen Objekten ist zu hoch, was auch bei der Evaluierung in der Arbeit von Kirst deutlich wird [1]. Dort hat die Weighting Method im Vergleich mit den anderen Methoden zu den meißten Kollisionen geführt.

Als nächste Methode wird Constraint Method aus Kapitel 2.2.2 analysiert. Die Constraint Method hat gegenüber der Weighting Method den Vorteil, dass Richtungen mit einem hohen Gefahrenwert ausgeschlossen werden können. Dies wird über die ε-Bedingung ε_d für den Gefahrenwert $f_d(x)$ gesteuert. In der Arbeit von Kirst [1] hat die Constraint Method außerdem die besten Ergebnisse in den Testszenarien erzielt. Im Unterschied zu der Arbeit von Kirst, werden in dieser Arbeit vier anstatt zwei Context Maps für die Bewegungssteuerung verwendet. Bei dem Konzept der Constaint Method werden abgesehen von der Interest-Map, für alle Context Maps ε-Bedingungen definiert. Nur die Richtungen, die alle ε-Bedingungen erfüllen, werden zur Auswahl einer Bewegungsrichtung zugelassen.

Die hohe Anzahl an Context Maps führt dabei zu folgenden Problem. Je mehr ε-Bedingungen erfüllt werden müssen, desto weniger ist die Wahrscheinlichkeit für Richtungen, die den Anforderungen gerecht werden. Dadurch stehen dem Finken nur sehr wenige Bewegungsrichtungen zur Auswahl. Dies wurde beim Anfertigen dieser Arbeit sehr schnell deutlich. Der Finken bewegte sich sehr eingeschenkt in der Testumgebung. Die Constraint Mehtod ist, wie die Arbeit von Kirst zeigt, für Bewegungssteuerungen mit nur wenigen Context Maps geeignet. Für die in dieser Arbeit entwickelte Bewegungssteuerung mit vier Context Maps ist die Methode jedoch ungeeignet.

Als letzte der drei Methoden, wird die Hybrid Method aus Kapitel 2.2.3 auf ihre Eignung überprüft. Die Hybrid Method kombiniert die beiden Ansätze der Weighting und Constraint Method. Dadurch werden die Nachteile der beiden Methoden kompensiert. In der Arbeit von Kirst hat die Hybrid Method bei der Verwendung von nur zwei Context Maps keinen Vorteil zu der Constraint Method gezeigt. Mit einer höheren Anzahl an Context Maps, wie bei der Bewegungssteuerung in dieser Arbeit, eignet sich die Hybrid Method jedoch gut. Nur mit der Hybrid Method lassen sich gefährliche Bewegungsrichtungen ausschließen, ohne dabei die Richtungsauswahl zu sehr einzuschränken. Im nächsten Kapitel wird die Bewegungssteuerung im Detail beschrieben.

3.3 Genereller Ansatz

Dieses Kapitel beschreibt die Grundidee der entwickelten Bewegungssteuerung. Wie schon zuvor erwähnt, basiert die Bewegungssteuerung auf dem Ansatz des Context Steering aus Kapitel 2.1. Bei diesem Ansatz wird eine Danger- und Interest-Map verwendet, um Gefahren auszuweichen und Ziele in der Umgebung anzufliegen. Für die Erstellung der Danger- und Interest-Map werden die Fähigkeiten des Finken, die in Kapitel 3.1 beschrieben werden, genutzt. Die Gefahrenwerte für die Danger-Map werden mit Hilfe der Sonar Entfernungsmesser berechnet. Die Sonar Entfernungsmesser tasten dazu die Umgebung rundherum um den Finken ab.

Wie in den Anforderungen beschrieben, soll der Finken in der Lage sein bestimmte Ziele in der Umgebung anzufliegen. Für diesen Zweck dient die Interest-Map. Da der Finken keine Informationen über seine globale Position via GPS hat, reicht es nicht aus, ihm die globalen Koordinaten der Ziele zu verfügung zu stellen. Die Positionen der Ziele müssen dem Finken in seinem lokalen Koordinatensystem bekannt sein. Das im Finken verbaute Modul zur Phasendifferenz-Entfernungsmessung kann zu diesem Zweck genutzt werden. Das Modul ist in der Lage die Entfernung zu anderen Modulen in der Umgebung zu messen. Außerdem können Informationen zwischen den Modulen ausgetauscht werden. Jedes Ziel erhält ein Modul. Somit kennt der Finken die Entfernung zu Zielen. Die Richtungsinformationen zu den Zielen fehlen dabei. Eine Möglichkeit die Richtungsinformationen zu bestimmen, ist den Finken anfangs kreisförmig bewegen zu lassen und dabei auf die Änderung der Distanzen zu achten. Damit lässt sich die Richtung zu einem Ziel zumindest erahnen. Im Rahmen dieser Arbeit, wird bei der Simulation der Bewegungssteuerung ein vereinfachtes Modell des Phasendifferenzsensors verwendet. In dem vereinfachten Modell sind die Richtung zu Zielen bekannt.

Als nächstes wird der Aspekt des Schwarmverhaltens erläutert. Damit ein Schwarm von Finken entstehen kann, ist eine Anziehung unter den Finken notwendig. Aus diesem Grund wird eine Attraction-Map verwendet. Sobald ein anderer Finken in der Nähe ist, wird dies mit einem Wert für die Anziehung in der Attraction-Map festgehalten. Somit kann die Bewegungssteuerung bei der Auswahl der nächsten Bewegungsrichtung ein Richtung bevorzugen, mit der ein Schwarm zusammengehalten wird.

Die Bewegungsteuerung entscheidet ein paar mal pro Sekunde über eine neue Be-

wegungsrichtung. In bestimmten Situationen kann es passieren, dass zwei entgegen gesetzte Richtungen sehr ähnliche Charakteristiken bei der Bewertung besitzen. Die Bewegungssteuerung kann in solch einer Situation zuerst eine Richtung wählen und im nächsten Iterationsschritt die andere Richtung. Falls dieses Verhalten oft nach einander auftritt, fängt der Finken an, an Höhe zu verlieren und das Flugverhalten wird instabil. Ein Quadrokopter ist physikalisch nicht dazu veranlagt, mehrmals in der Sekunde seine Bewegungsrichtung stark zu ändern. Aus diesem Grund wird die Same-Direction-Map angelegt. Die Bewegungssteuerung vermerkt die zuletzt gewählte Bewegungsrichtung in der Same-Direction-Map. Dadurch vermeidet die Bewegungssteuerung eine drastische Richtungsänderung im nachfolgenden Iterationsschritt. Somit wird ein stabiles Flugverhalten erzeugt.

Mit Hilfe der vier Context Maps hat die Bewegungsteuerung genügend Informationen, um den Finken in der Umgebung zu steuern. Wie im letzten Kapitel beschrieben, wird die Hybrid Method zur Auswahl einer Bewegungsrichtung genutzt. Dafür muss bestimmt werden, für welche der vier Context-Maps eine ε-Bedingung definiert wird und welche Funktionswerte für die Priorisierung gewichtet werden. In dieser Arbeit wird die Auswahl einer Bewegungsrichtung folgendermaßen durchgeführt. Es werden $\varepsilon_{D_{max}}$ und $\varepsilon_{S_{min}}$ als ε-Bedingung für die Danger- und Same-Direction-Map definiert. Somit werden nur die Bewegungsrichtungen berücksichtigt, für die gilt $f_d(x) \leq \varepsilon_{D_{max}} \wedge f_s(x) \geq \varepsilon_{S_{min}}$. Mit $\varepsilon_{D_{max}}$ wird bestimmt, wie gefährlich eine Bewegungsrichtung höchstens sein darf. $\varepsilon_{S_{min}}$ regelt, wie ähnlich eine Bewegungsrichtung zu der zuletzt ausgewählten Richtung mindestens sein muss. Im nächsten Schritt werden die Funktionswerte aus der Interest- und Attraction-Map zu einer Gewichteten Summe gebildet:

$$F_{i+a}(x) = \sum -w_i f_i(x) - w_a f_a(x) \tag{3.1}$$

Mit den Gewichten lässt sich steuern, ob sich der Finken in Richtung eines Ziels oder des Schwarms bewegt. Im letzten Schritt wird die Richtung mit dem kleinsten Wert $F_{i+a}(x)$ gewählt. Dem Finken wird das Kommando gegeben, in die neue Bewegungsrichtung zu fliegen. Die Geschwindigkeit des Finken wird dabei von dem Gefahrenwert $f_d(x)$ der gewählten Richtung beeinflusst:

$$\underset{v}{\rightarrow} = v_{max} * (1 - f_d(x)) \tag{3.2}$$

Mit der vorgestellten Bewegungssteuerung ist der Finken in der Lage, in unbekannten Umgebungen nach Zielen zu suchen und dabei möglichen Gefahren auszuweichen.

3.4 Context Steering Behavior

Die Bewegungssteuerung soll zu bestimmten Flugverhalten des Finken führen. Der Finken soll Ziele in der Umgebung anfliegen und dabei Wänden und anderen Objekten ausweichen. Zusätzlich soll der Finken von benachbarten Finken angezogen werden um ein Schwarm zu erzeugen. Das Flugverhalten soll dabei in allen Situationen stabil bleiben.

Um die die verschiedenen Verhalten zu realisieren, werden Context Maps genutzt. Jede Context-Map dient einem Zweck und verursacht ein bestimmtes Verhalten des Finken. In diesem Kapitel werden die einzelnen Context Maps vorgestellt. Abbildung 3.3 zeigt die Einbindung der Context Maps in der Bewegungssteuerung. Die Context Maps werden mit Hilfe der Sensordaten des Finken erzeugt. Mit der Hybrid Method werden die Informationen aus den Context Maps kombiniert und es wird eine Bewegungsrichtung bestimmt. Die Bewegungsrichtung wird anschließend wieder an den Finken geleitet. Der Finken führt dann die gewünschte Bewegung mit Hilfe der vier Rotoren aus. Als Ergebnis wird ein Bestimmtes Flugverhalten des Finken erzeugt.

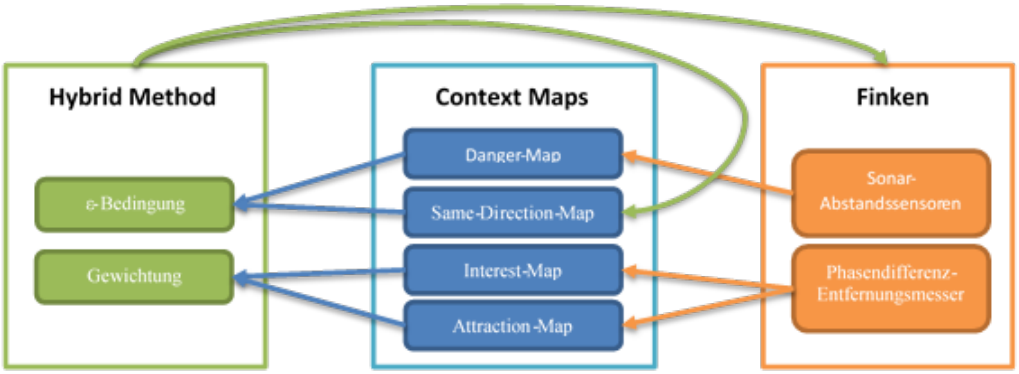

Abbildung 3.3: Schematische Darstellung der Bewegungssteuerung.

3.4.1 Danger Map

Die Danger-Map beinhaltet die Gefahrenwerte $f_d(x)$ für jede Abtastrichtung. Die Gefahrenwerte werden auf Grundlage der Sonar Sensordaten berechnet. Die Sonar Sensoren messen Entfernungen zu anderen Objekten, wie zum Beispiel Wänden. Ein Objekt wird vom Sensor erkannt, sobald es sich in dem Detektionsbereich des Sensors befindet. Dabei liefert der Sensor nur eine Entfernung. Die Information über die genaue Position des detektieren Objektes innerhalb des Detektionsbereiches fehlt. Es wird vereinfachter Weise angenommen, dass sich das Objekt in einem Winkel von $0°$ direkt frontal vor dem Sensor befinden. Ein geeigneter Öffnungswinkel für die Sensoren muss gewählt werden, damit der Messfehler durch die vereinfachte Annahme nicht zu groß wird. In dieser Arbeit wird ein öffnungswinkel von $30°$ genutzt. Dies ist ein akzeptabler Kompromiss zwischen dem erreichten Schwankungsbereich und der Höhe des Messfehlers.

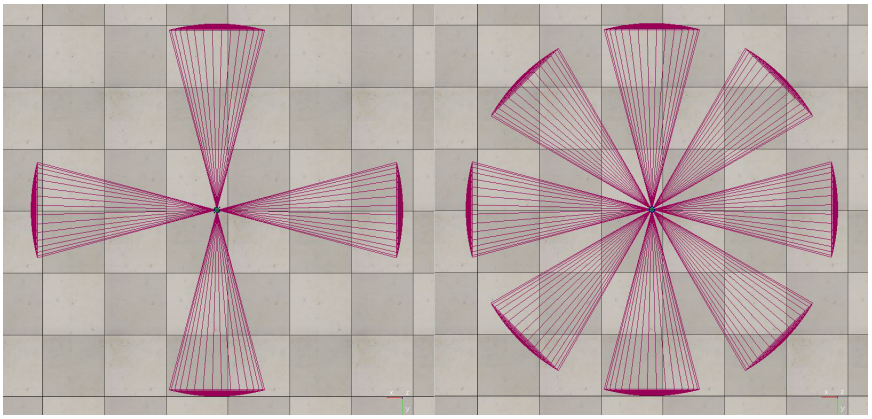

Abbildung 3.4: Darstellung des Finken und dessen Sonar Sensoren von Oben. Die Kegel stellen den Detektionsbereich der Sensoren dar. Links ist ein Finken mit vier Sensoren und rechts mit acht Sensoren zu sehen. Die Sensoren haben einen Öffnungswinkel von $30°$.

Die Abdeckung der Umgebung beträgt bei vier Sensoren mit einem Öffnungswinkel von $30°$ einen Anteil von $\frac{4*30}{360} \approx 33\%$. Die vier Sonar Sensoren des Finken reichen nicht aus, um Objekte in der Umgebung zuverlässig zu detektieren. Es gibt zu viele tote Winkel in denen Objekte unerkannt bleiben. Aus diesem Grund werden in dieser Arbeit acht Sensoren für das Abtasten der Umgebung genutzt. Die Auflösung der Danger-Map ist somit $|D| = 8$. Abbildung 3.4 zeigt den unter-

schied zwischen der Verwendung von vier und acht Sensoren mit einem Öffnungswinkel von jeweils $30°$. Die verwendeten Sonar Sensoren haben eine bestimmte Reichweite $r_{max} = 7.5m$. Die Sensoren liefern eine Entfernung $sensor_d$(x). Diese Entfernung wird auf den Bereich $[0, 1]$ abgebildet und von 1 abgezogen:

$$f_d(x) = 1 - \frac{sensor_d(x)}{r_{max}} \tag{3.3}$$

Das Ergebnis ist der Gefahrenwert $f_d(x)$. Dieser beschreibt, wie hoch die Gefahr in eine bestimmte Richtung ist. Dabei stellt $f_d(x) = 1$ eine sehr hohe Gefahr, und $f_d(x) = 0$ keine Gefahr dar. Der berechnete Gefahrenwert $f_d(x)$ wird in die Danger-Map eingetragen $D[x] = f_d(x)$. Anschließend wird der Gefahrenwert auch anteilsmäßig an die benachbarten Richtungen der Danger-Map verteilt. Somit wird das Problem aus Kapitel 2.1.3 umgangen.

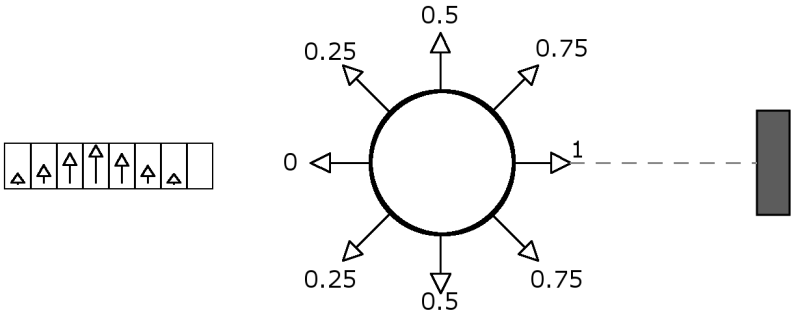

Abbildung 3.5: Verteilung des Gefahrenwertes $f_d(x)$ auf die Nachbarrichtungen. Die Zahlen geben die Gewichtungen für die einzelnen Richtungen an.

In Abbildung 3.5 wird die Filterung veranschaulicht dargestellt. Die Zahlen in der Abbildung geben den Anteil von $f_d(x)$ an, mit dem der Gefahrenwert auf die Nachbarrichtungen verteilt wird. Die Verteilung kann nach belieben angepasst werden. Mit Hilfe der Danger-Map ist der Finken in der Lage, sich sicher in der Umgebung zu bewegen.

3.4.2 Interest Map

Die Interest-Map wird dazu verwendet, den Finken zu bestimmten Zielen in der Umgebung zu bewegen. Dazu wird $f_i(x)$ für jede der acht Abtastrichtungen berechnet. $f_i(x)$ berechnet sich aus dem Kosinus des Winkels zwischen der aktuellen Abtastrichtung $\vec{x}$ und der tatsächlichen Zielrichtung $\vec{v_t}$ (siehe Abbildung 3.6).

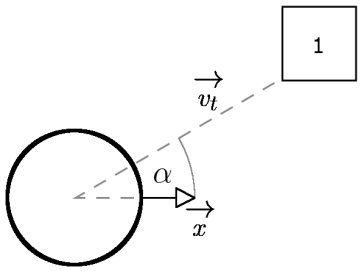

Abbildung 3.6: Berechnung von $f_i(x)$ für die Interest-Map. Das Quadrat ist ein Ziel in der Umgebung, welches vom Finken angeflogen werden soll.

Der berechnete Kosinus wird anschließend in den Bereich $[0, 1]$ abgebildet:

$$f_i(x) = \frac{\cos(\alpha) + 1}{2} = \frac{\frac{\vec{x} \cdot \vec{v_t}}{|\vec{x}| \cdot |\vec{v_t}|} + 1}{2} \tag{3.4}$$

$f_i(x)$ gibt an, wie hoch das Interesse in eine Richtung ist. Wenn mehrere Ziele in der Umgebung sind, soll der Finken das Ziel mit der kürzesten Entfernung bevorzugen. Aus diesem Grund wird $f_i(x)$ im Verhältnis zum weit entferntesten Ziel $\vec{v_{t_{max}}}$ gewichtet:

$$f_i(x) = \frac{\cos(\alpha) + 1}{2} * (1 - \frac{|\vec{v_t}|}{|\vec{v_{t_{max}}}|}) \tag{3.5}$$

Somit erhält ein näher liegendes Ziel einen größeren Interest-Wert $f_i(x)$, als ein weiter entferntes Ziel.

3.4.3 Attraction Map

Die Attraction-Map dient dazu, mehrere Finken in einem Schwarm zu halten. Dazu wird ein Finken von anderen Finken angezogen, die sich in innerhalb eines bestimmten Radius befinden. Die Werte $f_a(x)$ für die Attraction-Map werden wie die Werte $f_i(x)$ der Interest-Map berechnet (siehe Formel 3.4). Die Berechnung erfolgt für alle Finken, die sich innerhalb des Radius $r = 7.5\,m$ befinden. Die Entfernung wird zu dem Finken wird Dabei nicht weiter einbezogen. (siehe Abbildung 3.7). Der Wert $f_a(x)$ gibt an, inwiefern eine Richtung zu einem Schwarm führt.

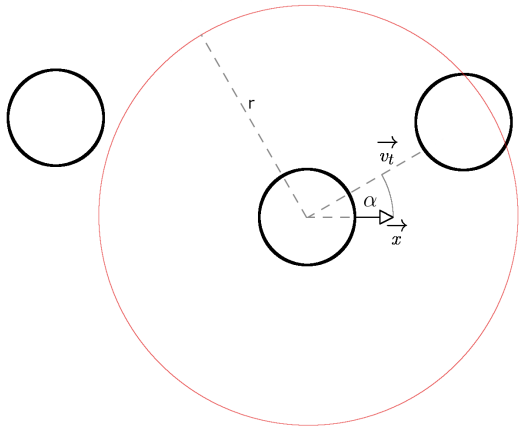

Abbildung 3.7: Berechnung von $f_a(x)$ für die Attraction-Map. der Finken wird von einem benachbarten Finken angezogen. Der Finken außerhalb des Radius r wird nicht berücksichtigt.

Bei der späteren Auswahl einer Bewegungsrichtung mit der Hybrid Method, werden die Werte der Attraction Method mit w_a gewichtet. Ein zu hoher Wert für das Gewicht w_a führt dazu, dass ein Schwarm auf einer Stelle in der Umgebung bleibt. Ein kleiner Wert für w_a lässt den Schwarm instabil werden. Der Schwarm löst sich in so einem Fall auf.

3.4.4 Same Direction Map

Die Same-Direction-Map wird genutzt, um die Bewegungsfreiheit des Finken künstlich einzuschränken. Eine Bewegungseinschränkung hindert den Finken daran, seine Bewegungsrichtung zwischen zwei Iterationsschritten stark ändern zu kann. Damit wird ein ständiges hin und her wechseln von Verhaltensentscheidungen vermieden und der Finken bleibt stabiler in der Luft. Abbildung 3.8 stellt die Funktionsweise schematisch dar. Im aktuellen Iterationsschritt geben die Werte $f_s(x)$ an, wie ähnlich eine Richtung zu der zuletzt ausgewählten Bewegungsrichtung ist. $f_s(x)$ wird folgendermaßen berechnet:

$$f_s(x) = \frac{\cos(\alpha) + 1}{2} = \frac{\frac{\vec{x} \cdot \vec{x_{t-1}}}{|\vec{x}| \cdot |\vec{x_{t-1}}|} + 1}{2} \tag{3.6}$$

Die ε-Bedingung $\varepsilon_s \in [0, 1]$ bestimmt den Umfang des Auswahlbereiches für mögliche Bewegungsrichtungen. In Abbildung 3.8 ist der Auswahlbereich grün dargestellt. ε_s gibt an, wie ähnlich eine Richtung mindestens, zu der zuvor ausgewählten sein muss. Ein zu hoher Wert für ε_s ist nicht empfehlenswert, da es die Richtungsauswahl zu sehr einschränkt.

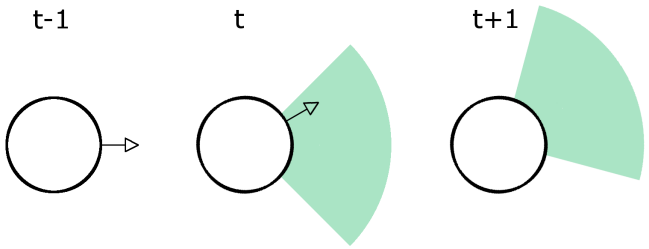

Abbildung 3.8: Einschränkung der Bewegungsmöglichkeit des Finken durch die Same-Direction-Map. Im Zustand $t - 1$ wurde eine Bewegungsrichtung gewählt. Zustand t zeigt den grün dargestellten Auswahlbereich für neue Bewegungsrichtungen und eine neu gewählte Richtung in diesem Bereich. Zustand $t + 1$ zeigt den neun Auswahlbereich für den nächsten Iterationsschritt an.

3.5 Zusammenfassung

An dieser Stelle wird die vorgestellte Bewegungssteuerung noch einmal kurz zusammengefasst. Zuerst wurden in Kapitel 3.1 die Anforderungen an die Bewegungssteuerung genannt. Außerdem wurden die Sensoren des Finken vorgestellt. In Kapitel 3.3 wurde der generelle Ansatz beschrieben. Dazu wurde die Verwendung der Sensoren, die Funktion der Context Maps und die Auswahlmethode einer Bewegungsrichtung erläutert. In den darauf folgenden Kapitel 3.4 wurden die einzelnen Context Maps im Detail erklärt. Es wurde auf die Berechnung der Funktionswerte und die Verwendung der dazugehörigen Gewichte und ε-Bedingungen eingegangen. Im nächsten Kapitel wird auf die Implementierung der Bewegungssteuerung in der Simulationsumgebung V-REP eingegangen.

Kapitel 4

Implementierung

Dieses Kapitel handelt von der Implementierung der Bewegungssteuerung in der Simulationsumgebung V-REP[1]. Zuerst wird ein Überblick über V-REP gegeben. Anschließend wird das virtuelle Modell des Finken und die Einbindung der Bewegungsteuerung vorgestellt.

4.1 Simulation mit V-REP

Für die spätere Evaluation der Bewegungssteuerung, sind praktische Experimente nötig. Experimente mit realen Finken sind in einem so frühen Stadium der Entwicklung aus Kostengründen nicht zu empfehlen. Aus diesem Grund wird die Bewegungssteuerung in einer Simulationsumgebung mit einem virtuellen Modell des Finken getestet. In diesem Kapitel wird die verwende Simulationsumgebung V-REP näher vorgestellt.

V-REP ist eine Simulationsumgebung, die speziell zur Simulation von Robotern vorgesehen ist. Die vielseitige und flexible Systemarchitektur von V-REP ermöglicht skalierbare und portable Lösungen mit komplexen Berechnungen. V-REP bietet eine Vielzahl von unterschiedlichen Komponenten zur Erstellungen einer Simulation an [26]:

- **Joints:**

[1] V-REP: Virtual Robot Experimentation Platform (http://www.coppeliarobotics.com/)

Joints ermöglichen Verbindungen mit verschiedenen Freiheitsgraden zwischen Objekten herzustellen.

- **Shapes:**
  Shapes sind 3D-Gitternetze zur visuellen Repräsentation von Objekten. Sie können Außerdem zur Kollisionsbestimmung und Entfernungsmessung verwendet werden.

- **Proximity sensors:**
  Näherungssensoren zum detektieren von benachbarten Objekten. Die Daten werden zum erstellen der Danger-Map verwendet.

- **Vision sensors:**
  Kamerasensoren, die das Extrahieren von Bildinformationen ermöglichen.

- **Force sensors:**
  Sie messen Zug- und Drehkräfte zwischen Objekten.

- **Graphen:**
  Mit der Hilfe von Graphen können Messwerte aufgezeichnet und visualisiert werden.

- **Kameras:**
  Kameras zeigen eine Szene aus einem bestimmten Blickwinkel. Es können mehrere Kameras in einer Szene verwendet werden um verschiedene Blickwinkel zu bekommen.

- **Lichter:**
  Lichter dienen zur Beleuchtung der Szene. Die Beleuchtung wird mit den Kamerasensoren ebenfalls erfasst.

- **Paths:**
  Mit der Hilfe von Paths werden Weginformationen definiert um Objekte entlang eines Pfades zu bewegen.

- **Dummies:**
  Dummies sind Hilfsobjekte. Sie werden hauptsächlich verwendet um Objekte hierarchisch zusammen zu fassen.

- **Mills:**
  Mills sind Fräsen-Objekte, mit denen andere Objekte gefräst werden. Sie werden mit einem dreidimensionalen Volumen repräsentiert.

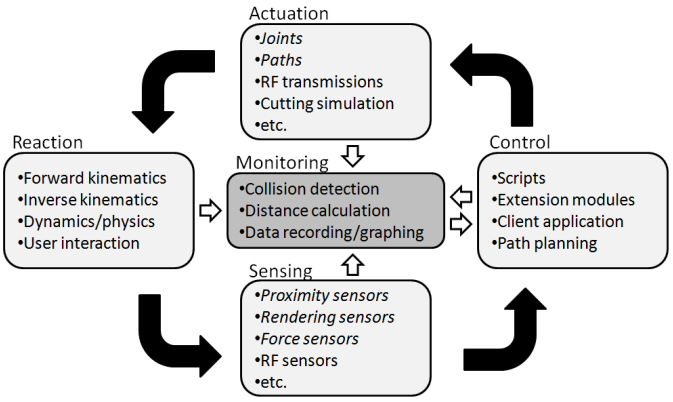

Abbildung 4.1: Simulationsschleife in V-REP [27]

Die Berechnungen der Simulation wird verteilt in unterschiedlichen Modulen durchgeführt. Abbildung 4.1 zeigt die dazugehörige Simulationsschleife. V-REP unterstützt zur Physiksimulation mehrere Physik Engines. In dieser Arbeit wird zur Simulation die kostenlose Bullet Physik Engine[2] genutzt. Zur Steuerung der Simulation und der einzelnen Objekte in der Szene kann auf unterschiedliche Weise durchgeführt werden. V-REP unterstützt zu diesem Zweck sogenannte Child-Scripts und Plugin-Extension-Modules [27]. Child-Scriptts sind Skripte, die in der Skriptsprache Lua[3] geschrieben werden und an Objekte in der Szene gebunden sind. Im Gegensatz zu einer zentralen Steuerung, hat dies den Vorteil, dass voll funktionsfähige Komponenten entstehen, die auch in eine andere Szene kopiert werden können. Mit den genanten Spezifikationen, ist V-REP die geeignete Simulationsumgebung zur Implementierung und Evaluierung der vorgestellten Bewegungssteuerung.

[2]Bullet Physics SDK: http://bulletphysics.org/wordpress/
[3]Lua: http://www.lua.org

4.2 Finken-Quadrokopter Modell

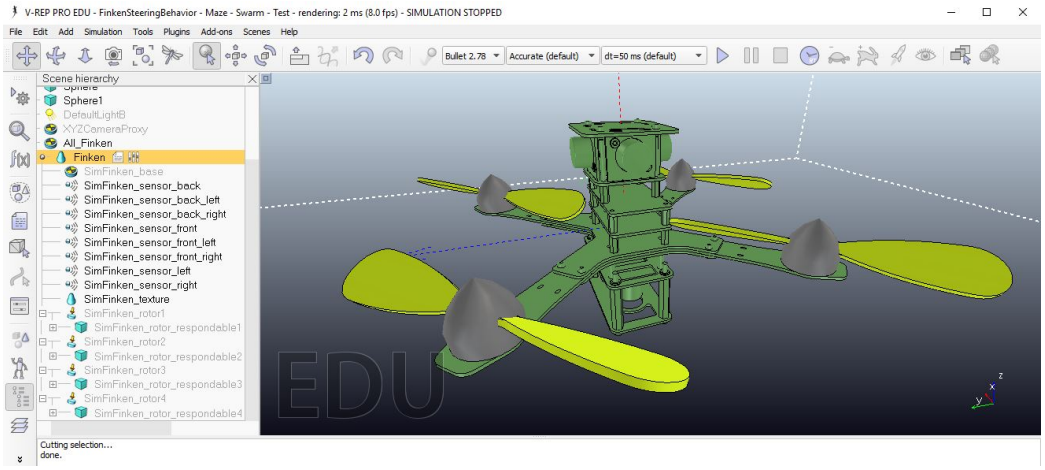

Abbildung 4.2: Virtuelles Finken Modell in V-REP.

In diesem Kapitel wird das virtuelle Modell des Finken in V-REP vorgestellt. Das virtuelle Modell bildet den realen Finken so genau wie möglich ab. Es besteht aus den V-REP Komponenten, die im vorherigen Kapitel 4.1 beschrieben sind. Das visuelle Erscheinungsbild wird dem Finken durch ein dreidimensionales Gitternetz gegeben, welches in V-Rep als Shape importiert wird (siehe Abbildung 4.2). Dieses wird ebenfalls zur Kollisionserkennung bei der Physikalischen Simulation genutzt. Das Finken Modell besitzt im Gegenteil zum realen Finken acht Ultraschallsensoren um die Umgebung abzutasten (siehe Abbildung 3.4). Diese haben eine Reichweite von $7,5$ Meter und einen öffnungswinkel von $30°$. Die Sensoren in der Simulation liefern im Gegensatz zu realen Sensoren, genaue Abstände zu detektierten Objekte. Die Höhen- und Bewegungsmessung wird vereinfachter Weise durch eine Positionsabfrage durchgeführt. Das gleiche gilt für die Entfernungsmessung zu Zielen. Die Funktion der vier Rotoren wird mit Hilfe einer physikalischen Partikelsimulation nachgebildet. Dazu werden an der Position der Rotoren Partikel senkrecht nach unten ausgestoßen. Dadurch wird ein Auftrieb erzeugt, mit dem das Modell an Höhe gewinnt. Die Steuerung des Modells geschieht mit einem Lua-Child-Script. Die Bewegungs- und Rotorensteuerung passiert in ausgelagerten Skripten. Diese werden vom Child-Script eingebunden. Das Child-Skript wird innerhalb der Simulationsschleife zweimal pro Iteration aufgerufen (siehe Abbildung 4.1). Beim Aufruf werden die Sensordaten gelesen und

an das Context Steering Script weitergegeben, welches sich um die Bewegungssteuerung kümmert. Mit den Sensordaten werden die Contex Maps erstellt und anschließend eine Bewegungsrichtung ausgewählt. Die eigentliche Flugsteuerung durch die vier Rotoren wird mit PID-Reglern[28] durchgeführt. Diese steuern den Schub jedes Rotors, um die gewünschte Bewegung des Modells durchzuführen. Abbildung 4.3 zeigt die Zusammenhänge der einzelnen Skripte.

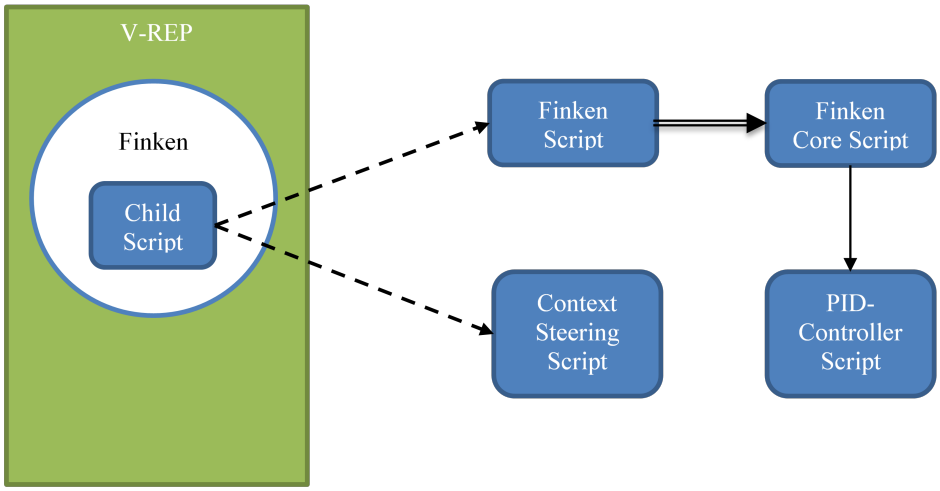

Abbildung 4.3: Übersicht der verwendeten Skripte.

Kapitel 5

Evaluation

In diesem Kapitel wird die vorgestellte Bewegungssteuerung evaluiert. Dazu wird zuerst die verwendete V-REP Szene und die Anfangskonfiguration der Parameter beschrieben. Anschließend wird das Flugverhalten eines einzigen Finken und eines Schwarms in der Szene evaluiert. Dabei sollen die Auswirkungen der Gewichte und der ε-Bedingungen auf das Flugverhalten analysiert werden. Am Ende des Kapitels wird die Stabilität des Schwarms in einer separaten Szene evaluiert.

5.1 Labyrinth Umgebung

Für die Evaluation der Bewegungssteuerung wird eine Labyrinth Umgebung genutzt (siehe Abbildung 5.1). Mit dieser Umgebung wird ein Innenraumszenario nachgebildet. Das Labyrinth hat die Ausmaße von $75m * 75m$. Die einzelnen Gänge sind $5m$ breit, mit Ausnahme zweier Stellen, mit $2.5m$ Breite. Im Labyrinth sind drei Ziele platziert. Diese sollen von den Finken eingesammelt werden. Die Finken starten von einer der drei Startpositionen $S1$, $S2$ oder $S3$. Die Startpositionen sind so gewählt, dass verschiedene Flugrouten beim Einsammeln der Ziele entstehen. Es gibt jeweils mehrere Routen, die zu einem Ziel führen. Zusätzlich sind einige Sackgassen eingebaut. Diese stellen eine besondere Herausforderung an die Bewegungssteuerung dar. Die Finken können bei bestimmten Parameterkonfigurationen, nicht mehr aus den Sackgassen kommen.

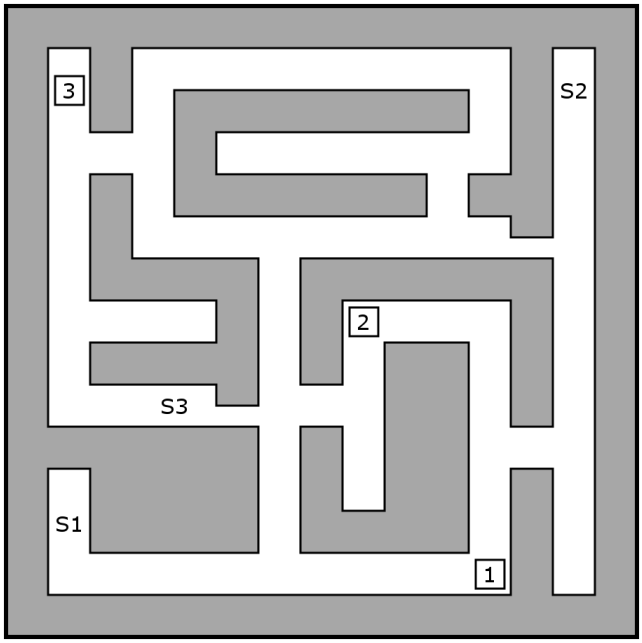

Abbildung 5.1: Schematische Draufsicht auf das Labyrinth mit den drei Startpositionen $S1$, $S2$ und $S3$. Die nummerierten Quadrate stellen Ziele dar, welche vom Finken eingesammelt werden sollen. Die grau markierten Bereiche sind Mauern.

5.2 Feste Parameter

Zur Evaluation werden für jede Konfiguration Simulationen durchgeführt. Dabei gibt es Parameter, die für jeden Simulationsdurchgang gleich bleiben. Diese sind in Tabelle 5.1 aufgeführt. Die Anzahl der Context Maps und der Sensoren bleibt unverändert, ebenso wie die Sensorparameter. Für die Simulation unter V-REP wird Bullet als Physic Engine genutzt, mit den Standartwerten für den Berechnungsmodus $Accurate$ und den Simulationszeitschritt $dt = 50ms$.

	Fester Parameter	**Wert**
Context Steering	Context Map Auflösung	8
	max. Target-Distanz für PID-Regler	$2m$
Sensoren für die Danger-Map	Anzahl Sensoren	8
	Öffnungswinkel	$30°$
	Reichweite	$7.5m$
Simulation in V-REP	Physic Engine	Bullet 2.75
	Berechnungsmodus	Accurate(default)
	Simulationszeitschritt	$dt = 50.0ms$
	max. Simulationsdauer	$t = 15min$

Tabelle 5.1: Feste Parameter für die Evaluation.

5.3 Flugverhalten eines einzelnen Finken

In diesem Kapitel wird die Bewegungssteuerung mit einem einzigen Finken in der Umgebung evaluiert. Dabei sollen die Auswirkungen der ε-Bedingungen auf das Flugverhalten des Finken betrachtet werden. Die Gewichte w_i und w_a spielen in diesem Fall keine Rolle, da es keine Anderen Finken in der Szene gibt, mit denen ein Schwarmverhalten erzeugt werden kann. Aus diesem Grund sind alle Werte in der Attraction-Map $f_a(x) = 0$. Der Finken starrtet von drei verschiedenen Startpunkten (siehe Abbildung 5.1). Die beiden ε-Bedingungen ε_d und ε_s werden bei jedem Simulationsdurchlauf variiert. Für jede Konfiguration werden drei Simulationsdurchläufe durchgeführt und das arithmetische Mittel gebildet. Es wird dabei die benötigte Zeit zum Einsammeln der Ziele gemessen. Außerdem wird die Anzahl der Kollisionen und die Anzahl der gesammelten Ziele festgehalten.

Ein Simulationsdurchlauf wird abgebrochen, sobald der Finken kollidiert oder die Simulationszeit die maximale Simulationsdauer von $15min$ überschreitet.

5.3.1 Ergebnisse

Die folgenden drei Tabellen zeigen die Ergebnisse der Simulationsdurchläufe für jeweils eine der drei Startpositionen.

Startposition	ε_s	ε_d	**Zeit**	**Kollisionen**	**Ziele**
	0	0.6	$15:00\,min$	0	$0/3$
	0.25	0.6	$15:00\,min$	0	$2/3$
	0.5	0.6	$15:00\,min$	0	$2/3$
	0.75	0.6	$07:06\,min$	0	$3/3$
	0.9	0.6	$14:28\,min$	0	$3/3$
	1	0.6	$14:40\,min$	0	$3/3$
$S1$					
	0.75	0	$15:00\,min$	0	$0/3$
	0.75	0.25	$15:00\,min$	0	$0/3$
	0.75	0.5	$15:00\,min$	0	$0/3$
	0.75	0.75	$07:05\,min$	0	$3/3$
	0.75	0.9	$08:53\,min$	0.3	$2.7/3$
	0.75	1	$00:07\,min$	3	$0/3$

Tabelle 5.2: Ergebnisse des Finken von Startposition $S1$.

Startposition	ε_s	ε_d	**Zeit**	**Kollisionen**	**Ziele**
	0	0.6	$15:00\,min$	0	$2/3$
	0.25	0.6	$08:41\,min$	0	$3/3$
	0.5	0.6	$11:18\,min$	0	$3/3$
	0.75	0.6	$11:03\,min$	0	$2.3/3$
	0.9	0.6	$15:00\,min$	0	$1/3$
	1	0.6	$15:00\,min$	0	$1/3$
$\boldsymbol{S2}$					
	0.75	0	$15:00\,min$	0	$0/3$
	0.75	0.25	$15:00\,min$	0	$0/3$
	0.75	0.5	$15:00\,min$	0	$0/3$
	0.75	0.75	$15:00\,min$	0	$1/3$
	0.75	0.9	$05:53\,min$	1	$0.3/3$
	0.75	1	$00:11\,min$	3	$0/3$

Tabelle 5.3: Ergebnisse des Finken von Startposition $S2$.

Startposition	ε_s	ε_d	**Zeit**	**Kollisionen**	**Ziele**
	0	0.6	$15:00\,min$	0	$0/3$
	0.25	0.6	$15:00\,min$	0	$0.3/3$
	0.5	0.6	$15:00\,min$	0	$0/3$
	0.75	0.6	$15:00\,min$	0	$2/3$
	0.9	0.6	$15:00\,min$	0	$0.6/3$
	1	0.6	$15:00\,min$	0	$0.3/3$
$\boldsymbol{S3}$					
	0.75	0	$15:00\,min$	0	$0/3$
	0.75	0.25	$15:00\,min$	0	$0/3$
	0.75	0.5	$15:00\,min$	0	$0/3$
	0.75	0.75	$15:00\,min$	0	$1/3$
	0.75	0.9	$08:15\,min$	0.6	$1.3/3$
	0.75	1	$00:6\,min$	3	$0/3$

Tabelle 5.4: Ergebnisse des Finken von Startposition $S3$.

5.3.2 Einfluss von ε_d

Die Ergebnisse aus den vorherigen Tabellen 5.2-5.4 zeigen, dass die ε-Bedingung ε_d einen erheblichen Einfluss auf die Anzahl der Kollision in den Simulationsdurchläufen hat. Bei einem Grenzwert von $\varepsilon_d \leq 0.75$ ist in keinen der Durchläufe eine Kollision aufgetreten. Erst ab höheren Werten kam es zu Kollisionen mit Wänden. Dabei flog der Finken bis auf ca. $20\,cm$ an die Wände heran.

Kleine Grenzwerte von $\varepsilon_d < 0.5$ führen dazu, dass der Finken auf der Stelle verweilt. Die Korridor ähnlichen Gänge des Labyrinths sind mit einer Breite von $5\,m$ in diesem Fall zu eng für den Finken. Die engen Gänge und die Verteilung der Gefahrenwerte auf die Nachbarrichtungen(siehe Kapitel 3.4.1) führen dazu, dass es in dem Fall keine Bewegungsrichtung gibt, welche die Bedingung $f_d(x) < 0.5$ erfüllt. Die beiden engen Stellen im Labyrinth wurden nur von Finken mit einem Grenzwert ab $\varepsilon_d \geq 0.75$ passiert. Mit unterschiedlichen Werten für ε_d ergaben sich unterschiedliche Flugrouten. Die Ergebnisse zeigen, dass ein sicheres Flugverhalten in einer unbekannten Umgebungen mit feststehenden Objekten erzielt werden kann.

5.3.3 Einfluss von ε_s

Das Labyrinth besitzt einige Sackgassen, in denen sich der Finken festsetzen kann. Mit der ε-Bedingung ε_s lässt sich regeln, wie ähnlich eine neue Bewegungsrichtung zu der letzten mindestens sein muss. Die Ergebnisse zeigen, dass der Finken bei höheren Werten für ε_s eher aus einer Sackgasse raus kommt, als bei kleinen Werten. Dies wird besonders bei den Durchläufen von der Startposition $S1$ in Tabelle 5.2 deutlich. $S1$ befindet sich genau in solch einer Sackgasse. Zu Beginn einer Simulation versucht der Finken nach Oben in Richtung Ziel 2 und Ziel 3 zu fliegen. Die Sackgasse hindert den Finken daran. Bei $\varepsilon_s = 0$ bleibt der Finken über die gesamte Zeit von $15\,min$ in der Sackgasse. Mit ansteigenden Werten für ε_s gelingt es dem Finken immer früher aus der Sackgasse zu gelangen und somit mehr Ziele innerhalb der $15\,min$ zu erreichen. Die besten Ergebnisse werden bei $\varepsilon_s \geq 0.75$ erzielt. Eine visuelle Repräsentation der eingeschränkten Richtungsauswahl mit $\varepsilon_s \geq 0.75$ ist in Abbildung 3.8 in Kapitel 3.4.4 zu sehen. Bei $\varepsilon_s \geq 0.75$ werden nur Richtungen zugelassen, die maximal um $45°$ von der vorherigen Bewegungsrichtung abweichen. Eine zu starke Richtungseinschränkung

verschlechtert die Ergebnisse. Bei $\varepsilon_s \geq 0.9$ wird nur die vorherige Richtung zur Auswahl zugelassen. Dadurch verhält sich der Finken wie ein Ping-Pong Ball. Wird eine Bewegungsrichtung gewählt, so fliegt der Finken so lange in die selbe Richtung, bis die ε-Bedingung ε_d für die Danger-Map verletzt wird und somit eine neue Richtung gewählt wird.

5.4 Flugverhalten eines Schwarms

In diesem Szenario befinden sich drei Finken im Labyrinth. Somit muss ein Finken nicht nur feststehenden Wänden ausweichen, sonder auch den anderen Finken. Sobald ein Finken ein Ziel erreicht hat, werden die restlichen Finken darüber informiert. Aus diesem Grund werden kürzere Zeiten zum Einsammeln erwartet. Die Finken sind zu Beginn der Simulation so verteilt, dass sich auf jeder der drei Startpositionen $S1 - S3$ ein Finken befindet. Wie schon bei dem Szenario mit einem einzigen Finken, werden die ε-Bedingungen ε_s und ε_d zwischen den Durchläufen variiert. Die beiden Gewichte $w_i = 0.6$, $w_a = 0.4$ bleiben dagegen fest. Diese werden im nächsten Szenario untersucht. Es wird die Zeit gemessen, die von den drei Finken benötigt wird, um alle drei Ziele im Labyrinth einzusammeln. Dabei werden die Anzahl der Kollisionen und die Anzahl der eingesammelten Ziele festgehalten. Es werden für jede Parameterkonfiguration drei Durchläufe gestartet und das arithmetische Mittel berechnet.

5.4.1 Ergebnisse

Wie zu erwarten war, wurden die Ziele insgesamt in einer kürzeren Zeit eingesammelt. Das liegt vor allem daran, das die Finken nicht so oft in Sackgassen stecken blieben. Sobald ein Ziel von von einem der Finken eingesammelt wurde, ergab sich für die restlichen Finken eine neue Situatio. Dadurch hat ein Finken, der zuvor in einer Sackgasse feststeckte, ein neues Ziel angepeilt und kam somit aus der Sackgasse raus. Dies ist besonderes bei dem Vergleich zwischen den Ergebnissen aus Tabelle 5.2-5.4 und Tabelle 5.5 zu sehen. Bei der Konfiguration $\varepsilon_s = 0$, $\varepsilon_d = 0.6$ konnte der Schwarm von Finken alle drei Ziele innerhalb der Zeit finden (siehe Tabelle 5.5). In dem Szenario mit nur einem Finken konnte kein

einziges Ziel bei gleicher Konfiguration erreicht werden, da sich der Finken in einer Sackgasse verfangen hat (siehe Tabelle 5.2-5.4). Die Anzahl der Kollisionen ist im Vergleich zum Szenario mit nur einem Finken deutlich angestiegen. Dies ist auf die zusätzlichen Finken in dem Labyrinth zurück zu führen. Die Finken stellen untereinander bewegliche Gefahrenobjekte da. Die Umsetzung der Ausweichbewegung wird vom Finken nicht schnell genug umgesetzt. Somit ist eine frühe Reaktion erforderlich, die mit einem niedrigen Grenzewert $\varepsilon_d \leq 0.6$ für den Gefahrenwert aus der Danger-Map erreicht werden kann.

ε_s	ε_d	**Zeit**	**Kollisionen**	**Ziele**
0	0.6	$08:31\,min$	0	3/3
0.25	0.6	$08:33\,min$	0	2/3
0.5	0.6	$05:25\,min$	0	3/3
0.75	0.6	$05:11\,min$	0	3/3
0.9	0.6	$06:44\,min$	0	3/3
1	0.6	$06:29\,min$	0	3/3
0.75	0	$15:00\,min$	0	0/3
0.75	0.25	$15:00\,min$	0	0/3
0.75	0.5	$15:00\,min$	0	0/3
0.75	0.75	$06:47\,min$	1	3/3
0.75	0.9	$07:28\,min$	2.6	2.3/3
0.75	1	$00:23\,min$	3	0/3

Tabelle 5.5: Ergebnisse des Schwarms mit drei Finken im Labyrinth.

5.5 Stabilität des Schwarms

In diesem Szenario wird die Auswirkung der der beiden Gewichte w_i und w_a analysiert. Dafür wird eine andere Umgebung verwendet, da das Labyrinth mit seinen engen Gängen ungeeignet ist. Abbildung 5.2 zeigt die verwendete Umgebung. In der Umgebung ist ein Schwarm von fünf Finken platziert. Diese haben die Aufgabe, das Ziel 1 einzusammeln. Dabei befinden sich drei eckige Säulen zwischen dem Schwarm und dem Ziel. Die Gewichte w_i und w_a bestimmen, ob ein Finken

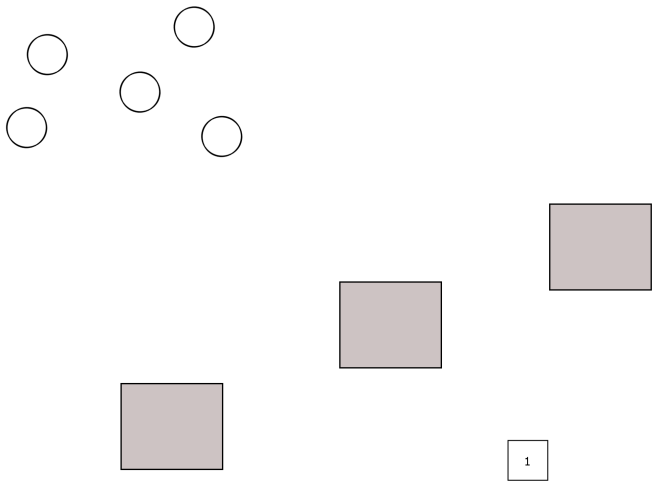

Abbildung 5.2: Umgebung zur Analyse der Stabilität eines Schwarms.

eher eine Richtung zu einem Ziel oder zu einem anderen Finken bevorzugt (siehe Kapitel 3.3). Während der Simulationsdurchläufe werden verschiedene Konfigurationen der beiden Gewichte genutzt. Die beiden ε-Bedingungen bleiben fest $\varepsilon_s = 0.75$, $\varepsilon_d = 0.6$. Es wird die Zeit zum Erreichen des Ziels, die Anzahl an Kollisionen und die Anzahl an Ausreißern festgehalten. Für jede Konfiguration der Gewichte werden jeweils drei Durchläufe simuliert und das arithmetische Mittel berechnet.

5.5.1 Ergebnisse

Die Ergebnisse aus Tabelle 5.6 zeigen die Auswirkung der beiden Gewichte sehr deutlich. Je höher die Gewichtung auf dem Interessenwert ist, desto kürzer ist die Zeit, bis der Schwarm das Ziel erreicht. Außerdem führt eine hohe Gewichtung des Interessenwertes zu mehr Ausreißern, aber dafür zu weniger Kollisionen. Im Gegensatz dazu, führt eine hohe Gewichtung der Werte aus der Attraction-Map zu längeren Zeiten und weniger Ausreißern. Dafür steigt aber die Anzahl der Kollisionen. Eine spannende Beobachtung dabei ist, dass bei $w_a = 1$ weniger Kollisionen entstanden sind als bei $w_a = 0.8$. Dies lässt sich darauf zurückführen, das die Finken bei der Konfiguration $w_a = 1$ gar nicht erst in die Nähe der Säulen und damit in die Nähe enger Stellen kommen. Diese Ergebnisse und die Ergebnisse aus

dem vorherigen Szenario(siehe Kapitel 5.4.1) zeigen, dass die Bewegungssteuerung nicht robust genug ist, um beweglichen Gefahren in besonders engen Umgebungen auszuweichen. Dies kann durch einen niedrigen Grenzwert ε_d für den Gefahrenwert verbessert werden, aber dadurch werden die Abstände der Finken und gleichzeitig die Anzahl der Ausreißer höher. Ein stabiler und kollisionsfreier Schwarm mit kleinen Abständen zwischen den Finken ist mit der vorgestellten Bewegungssteuerung nicht realisierbar.

w_a	w_i	**Zeit**	**Kollisionen**	**Ausreißer**
0	1	$00:18\,min$	0	2.3
0.2	0.8	$00:20\,min$	0	2.3
0.4	0.6	$00:24\,min$	0	2
0.6	0.4	$00:52\,min$	0	1
0.8	0.2	$01:59\,min$	$2,6$	0
1	0	$01:55\,min$	0.6	1

Tabelle 5.6: Ergebnisse der verschiedenen Konfigurationen von w_i und w_a für einen Schwarm mit fünf Finken.

Kapitel 6

Zusammenfassung

6.1 Fazit

In dieser Arbeit wurde eine Bewegungssteuerung für den Finken Quadrokopter vorgestellt, die es dem Finken ermöglicht sich in einer unbekannten Umgebung zu bewegen. Außerdem ermöglicht die Bewegungssteuerung ein Schwarmverhalten der Finken.

Die Bewegungssteuerung basiert auf dem Konzept des Context Steering. Es wurde gezeigt, wie die Sensoren des Finken dazu genutzt werden können, um die einzelnen Context-Maps für das Context Steering zu erstellen. Dabei wurde festgestellt, dass die vier verbauten Abstandssensoren des Finken nicht dazu ausreichen, Hindernisse in der Umgebung zuverlässig zu erkennen. Die Verwendung von acht anstatt vier Abstandssensoren führte dagegen zu ausreichend befriedigenden Ergebnissen.

Das Grundkonzept des Context Steering wurde um zwei weitere Context Maps erweitert. Die Same-Direction-Map wurde zur Verbesserung der Flugstabilität und zur Einschränkung der Bewegungsrichtung hinzugefügt. Die Attraction-Map wurde hinzugefügt, um mehrere Finken in einem Schwarm zusammen zu halten. Für die Auswahl einer Bewegungsrichtung wurden drei Ansätze diskutiert. Dabei hat sich die Hybrid Method als am besten geeignet herausgestellt.

Zur Evaluierung der Bewegungssteuerung wurden drei Szenarien in der Simulationsumgebung V-REP mit verschiedenen Konfigurationen der Parameter simu-

liert. Die Ergebnisse haben die Auswirkungen der einzelnen Gewichte und ε-Bedingungen der Hybrid Method auf das Flugverhalten gezeigt. Das Flugverhalten kann durch die Gewichte und ε-Bedingungen intuitiv angepasst werden und somit verschiedene Flugverhalten des Finken erzeugen.

Die Auswertung der Ergebnisse hat die Stärken der vorgestellten Bewegungssteuerung deutlich gemacht. Die Bewegungssteuerung ist für einzelne Finken in einer Umgebung mit unbeweglichen Objekten sehr gut geeignet. Es konnte ein stabiles Flugverhalten ohne Kollisionen erzeugt werden. Dabei war der Finken sogar in der Lage eigenständig aus Sackgassen zu gelangen, ohne dabei eine Art von Gedächtnis zu besitzen.

Schwächen zeigte die Bewegungssteuerung dagegen bei beweglichen Gefahrenobjekten in der Umgebung. Der Finken konnte in manchen Situationen nicht rechtzeitig auf eine zukommende Gefahr reagieren. Dies liegt vor allem an der Umgebungsabdeckung der acht Abstandssensoren von 66%. Außerdem reagiert der Finken zu träge auf neue Steuerungsbefehle. Ein kollisionsfreier und stabiler Schwarm mit kurzen Abständen zwischen den Finken, konnte somit nicht erzeugt werden. Die Bewegungssteuerung eignet sich für Schwärme mit großen Distanzen zwischen den Finken. Die Stabilität des Schwarms ist dabei jedoch nicht sehr hoch. Dadurch passiert es, dass sich der Schwarm an einer Abzweigung teilen kann oder Ausreißer entstehen.

6.2 Ausblick

Insgesamt gesehen, bildet die vorgestellte Bewegungssteuerung eine gute Grundlage zur Steuerung des Finken. Diese kann in zukünftigen Arbeiten an vielen verschiedenen Stellen erweitert und angepasst werden.

Die Bewegungsteuerung ist von den Fähigkeiten des Finken abhängig. In dem Bereich kann der Finken vor allem an den verwendeten Sensoren verbessert werden. Denkbar wäre zum Beispiel eine höhere Anzahl an Abstandssensoren. Die aktuell verwendeten Ultraschallsensoren können eventuell in Zukunft durch einen Lasersensor ausgetauscht werden. Zum Zeitpunkt dieser Arbeit wurde ein geeigneter und vor allem erschwinglicher Lasersensor von der Firma Scanse vorgestellt [29]. Dieser kann die gesamte Umgebung in bis zu $40\,m$ Entfernung mit einem rotierenden Laser abtasten. Somit können die Abstände und vor allem die Lage von Gefahrenobjekten und Hindernissen genau bestimmt werden. Dadurch würde die Bewegungssteuerung robuster gegen ungewollte Kollisionen werden

Die Bewegungssteuerung an sich bietet ebenfalls potentielle Stellen zur Verbesserung und Erweiterung an. Denkbar sind zum Beispiel zusätzliche Context Maps um neue Flugverhalten zu erzeugen. Ebenso kann die Auswahlmethode einer Bewegungsrichtung ersetzt oder angepasst werden, um möglichst bessere Ergebnisse zu erzielen.

Literaturverzeichnis

[1] M. Kirst, "Multicriteria-optimized context steering for autonomous movement in games," Master's thesis, Otto-von-Guericke Universität Magdeburg, 2015.

[2] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, pp. 664–674, 2012.

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2520–2525.

[4] "Dji phantom 4," https://www.dji.com/de/product/phantom-4, online; Accessed: 05.08.2016.

[5] T. Anai, T. Sasaki, K. Osaragi, M. Yamada, F. Otomo, and H. Otani, "Automatic exterior orientation procedure for low-cost uav photogrammetry using video image tracking technique and gps information," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, 2012.

[6] J. Pestana, J. L. Sanchez-Lopez, S. Saripalli, and P. Campoy, "Computer vision based general object following for gps-denied multirotor unmanned vehicles," in *2014 American Control Conference*. IEEE, 2014, pp. 1886–1891.

[7] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *International Symposium on Robotics Research (ISRR)*, vol. 2, 2011.

[8] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 2815–2821.

[9] S. Grzonka, G. Grisetti, and W. Burgard, "A fully autonomous indoor quadrotor," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 90–100, 2012.

[10] E. Masehian and Y. Katebi, "Robot motion planning in dynamic environments with moving obstacles and target," *International Journal of Mechanical Systems Science and Engineering*, vol. 1, no. 1, pp. 20–25, 2007.

[11] N. Gageik, T. Müller, and S. Montenegro, "Obstacle detection and collision avoidance using ultrasonic distance sensors for an autonomous quadrocopter," *University Of Würzburg, Aerospace Information Technology (Germany), Würzburg September*, 2012.

[12] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *Control and Automation, 2008 16th Mediterranean Conference on*. Ieee, 2008, pp. 1258–1263.

[13] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *AIAA guidance, navigation and control conference and exhibit*, 2008, pp. 1–14.

[14] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH computer graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[15] ——, "Steering behaviors for autonomous characters," in *Game developers conference*, vol. 1999, 1999, pp. 763–782.

[16] A. Fray, "Context steering: Behavior-driven steering at the macro scale," in *Game AI Pro 2: Collected Wisdom of Game AI Professionals*. CRC Press, 2015, pp. 183–193.

[17] S. B. Damelin and W. Miller Jr, *The mathematics of signal processing*. Cambridge University Press, 2012, no. 48.

[18] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12, pp.78-96.

[19] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE transactions on Automatic Control*, vol. 8, no. 1, pp. 59–60, 1963.

[20] S. Gass and T. Saaty, "The computational algorithm for the parametric objective function," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 39–45, 1955.

[21] Y. Y. Haimes, L. Ladson, and D. A. Wismer, "Bicriterion formulation of problems of integrated system identification and system optimization," p. 296, 1971.

[22] R. E. Wendell and D. Lee, "Efficiency in multiple objective optimization problems," *Mathematical Programming*, vol. 12, no. 1, pp. 406–414, 1977.

[23] H. Corley, "A new scalar equivalence for pareto optimization," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 829–830, 1980.

[24] C. Fritsche and A. Klein, "On the performance of hybrid gps/gsm mobile terminal tracking," in *2009 IEEE International Conference on Communications Workshops*. IEEE, 2009.

[25] "Finken-iii," http://www.is.ovgu.de/SwarmLab/Robots.html, online; Accessed: 20.07.2016.

[26] M. F. E. Rohmer, S. P. N. Singh, "V-rep: a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[27] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform v-rep: a versatile 3d robot simulator," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 51–62.

[28] L. M. Argentim, W. C. Rezende, P. E. Santos, and R. A. Aguiar, "Pid, lqr and lqr-pid on a quadcopter platform," in *Informatics, Electronics & Vision (ICIEV), 2013 International Conference on*. IEEE, 2013, pp. 1–6.

[29] "sweep lidar sensor der firma scanse," http://scanse.io/, online; Accessed: 10.08.2016.