Technische Universität Hamburg-Harburg

Untersuchung von Konzepten zur Formationseinnahme und Einhaltung in einer UAV-Simulationsumgebung

Bachelor Arbeit

Februar 2012

vorgelegt von:
Christian Matzat

Erstprüfer:
Prof. Dr. Ralf Möller

Zweitprüfer:
Prof. Dr. Herbert Werner

Betreuer:
Dipl.-Ing. Karsten Martiny

Inhaltsverzeichnis

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Hamburg, den 10.Februar 2012

Abstrakt

Im Rahmen dieser Ausarbeitung werden Konzepte vorgestellt und weiterentwickelt, um UAVs den Flug innerhalb einer Formation zu ermöglichen. Die Bedingungen, die für die Konstruktion einer Formation erfüllt sein müssen, werden genauso wie auch der eigentliche Ablauf der Formationsbildung erläutert. Steuerungsalgorithmen sorgen für die Einhaltung der Formation während des Fluges. Die Auflösung der starren Formation erlaubt den UAVs das rechtzeitige Ausweichen vor Hindernissen. Die auf Grundlagen aus dem Bereich der diskreten Mathematik beruhenden Modelle, ermöglichen die Erweiterung der bereits in einem Simulator implementierten UAVs hin zu der Nutzung von Formationen.

Kapitel 1

Einleitung

Die Entwicklung von UAVs hat in den letzten Jahren große Fortschritte gemacht, sodass mittlerweile das Anwendungsspektrum für UAVs im zivilen Bereich sehr vielfältig ist. Es reicht von der Luftbildarchäologie, über die Untersuchung von Sturmschäden bis hin zu der Messdatensammlung in der Erdatmosphäre. Das Institut "Software Technology Systems" der Technischen Universität Hamburg-Harburg hat einen Simulator entwickelt, in dem Quadrokopter, Hubschrauber mit vier Rotoren, für die Waldbrandaufklärung genutzt werden. In dieser Bachelor-Arbeit werden Konzepte untersucht, die den UAVs das Halten eines Formationsfluges und koordinierte Bewegungen während der Ausführung von Aufgaben ermöglichen sollen. Die UAVs sollen in naher Zukunft alle Voraussetzungen erfüllen, die von intelligenten, autonom handelnden Agenten erwartet werden. Die Agenten kommen ohne eine externe Steuerung aus und handeln vor allem initiativ und weniger reaktiv. Agenten kooperieren bei der Ausführung von Aufgaben, die sie alleine nicht angehen könnten [22]. Wenn es Alternativen bei der Zielerfüllung gibt, versuchen die Agenten ihren Nutzen zu maximieren. Äquivalent bedeutet dies, dass Kosten, messbar durch die benötigte Zeit oder den Treibstoffverbrauch, reduziert werden sollen. Das Fliegen innerhalb einer Formation hat für die beteiligten UAVs unterschiedliche Vorteile. Angefangen bei einer höheren Ausfallsicherheit bis hin zu der Kostenminimierung und Robustheit. Der Rechenaufwand für das Konstruieren und Aufrechterhalten einer Formation, sowie die hierfür benötigte Zeit fallen kaum ins Gewicht. Die Gefahr, dass ein UAV sich aufgrund von fehlerhaft erfassten Messdaten verirrt, ist bei einem einzigen UAV höher, als wenn eine ganze Gruppe UAVs Messdaten auswertet, um diese zur Entscheidungsfindung zu verwenden. Die Wahrscheinlichkeit für die Falscheinschätzung eines Hindernisses ist auch niedriger. Nachrichtenübertragungs- und Detektionsfehler können auftreten, wenn Sensoren von Hitze, Eis oder Qualm gestört oder beschädigt werden. Die Zuverlässigkeit und Aussagekraft von Messdaten wird

verbessert, wenn diese aus verschiedenen, räumlich versetzten Blickwinkeln erfasst werden. Auf diese Weise aufgenommene Bodenmessdaten ermöglichen die Erstellung eines 3-dimensionalen Boden-Gitternetzes.

Das Arbeiten der UAVs innerhalb eines verteilten Systems ermöglicht die Aufteilung der Tätigkeiten, die parallel bearbeitet werden können. Dem Agenten i ist bekannt, wo sich das UAV j befindet, das beispielsweise die Sensormessdaten, die i gesammelt hat, auswerten kann. Diese Daten können auf kurzem und zuverlässigen Wege gesandt werden. Weitere Vorteile des emergenten Systems lassen sich in [3] finden.

Es ist günstiger kleine, transportable Sensoren auf mehrere UAVs zu verteilen [5], anstatt ein großes, schweres und alles beherrschendes UAV zu konstruieren. Der Schaden wäre immens, würde solch ein UAV beschädigt werden oder verloren gehen. Bei der Auswahl von Konzepten steht im Vordergrund, dass der Energiebedarf möglichst gering sein soll. Die Auslastung der CPU hat einen vernachlässigbar kleinen Einfluss auf die Batterie, deren Laufzeit etwa 30 Minuten beträgt. Die Algorithmen werden daher so konzipiert, dass die Anzahl an berücksichtigten Agenten bei der Berechnung von Werten maximiert wird. Der Algorithmus terminiert schneller, als wenn nur wenige Parameter berücksichtigt werden könnten. Die "Cyclic stop-and-go strategy", die in Kapitel 3 eingeführt wird, ist hierfür ein gutes Beispiel.

In dem 2. Kapitel werden Formationseigenschaften definiert. Darauf aufbauend werden in Kapitel 3 Algorithmen vorgestellt und Konzepte erarbeitet, damit unter anderem Agenten einer bereits bestehenden Formation zugewiesen werden können. Es wird die generische Konstruktion einer Formation unter Berücksichtigung von Kriterien wie minimaler Starrheit erläutert. Voraussetzungen für die Formationseinhaltung einer sich unter Umwelteinflüssen bewegenden Formation werden erarbeitet. In dem Kapitel 4 steht das Ausweichen vor Hindernissen im Mittelpunkt und in Kapitel 5 wird diskutiert, wie die erarbeiteten Konzepte auf die im Simulator implementierten UAVs übertragen werden können.

1.1 Quadrokopter

Der Quadrokopter, siehe Abbildung 1.1, ähnelt am ehesten einem Helikopter, da dieser ebenfalls senkrecht startet und landet. 2 der 4 Rotoren sind linksdrehend und die anderen beiden sind rechtsdrehend, damit es, wie in [24] beschrieben, zum Ausgleich des Drehmoments kommen kann. Eine den Quadrokopter auszeichnende Fähigkeit ist das stabil in der Luft Schweben. Hierzu wertet die Steuerelektronik kontinuierlich die Lagesensoren aus und korrigiert.

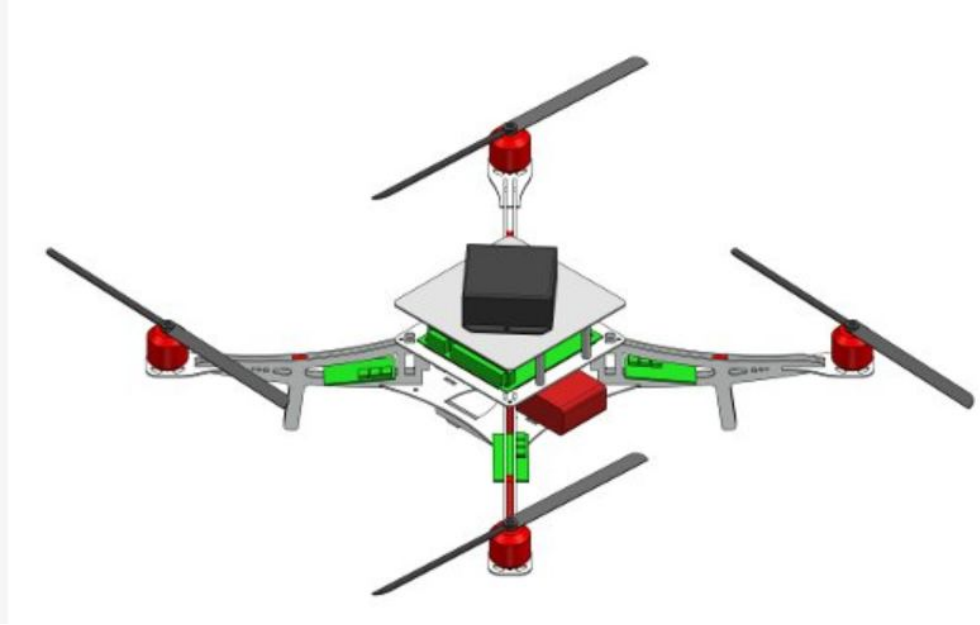


Abbildung 1.1: Quadrokopter(aus [25])

1.2 Waldbrandaufklärung

Bei der Waldbrandaufklärung wird versucht, ein Feuer bereits in der Entstehungsphase zu erkennen. Ein frühzeitiges Eingreifen von Seiten der Feuerwehr erhöht deutlich die Chancen, dass das Feuer noch eingedämmt werden kann und nicht außer Kontrolle gerät. In [23] ist beschrieben, dass in waldbrandgefährdeten Gebieten bereits Überwachungstürme errichtet worden sind, von denen aus Kameras Bilder von der Umgebung aufzeichnen. Spezielle Softwareprogramme identifizieren aufsteigenden Qualm. UAVs, die optimalerweise nahe des Überwachungsturms stationiert sind, werden zu der Stelle gesandt, wo der Ursprung des Qualms vermutet wird. Ein frühzeitiger Überblick über das Ausmaß des Feuers ermöglicht eine gezieltere Koordination der anlaufenden Löscharbeiten. Es lassen sich auch Fehlalarme rechtzeitig erkennen. Ein Fehlalarm kann ausgelöst werden, wenn Staub, der von arbeitenden Maschinen aus der Landwirtschaft aufgewirbelt wird, fälschlicherweise für Qualm gehalten wird. Nebel und aufsteigender Wasserdampf können ebenfalls mit Qualm verwechselt werden. Die UAVs umfliegen innerhalb ihrer Formation das Feuer und bestimmen Parameter, wie die von Wind und Boden abhängende primäre Ausbreitungsrichtung, sowie die Stärke und Giftigkeit des Qualms. Elektromagnetische Wellen im Mikrowellenbereich können Rauch und Staub durchdringen, sodass auch bei einer sehr starken Rauchentwicklung Rückschlüsse auf den tatsächlich brennenden Bereich möglich sind.

UAVs auf gut Glück einen Waldbrandgefährdeten Bereich überfliegen zu lassen, ist ebenfalls sinnvoll, auch wenn noch kein konkreter Brandherd ausgemacht wurde, da die UAVs einen großen Bereich abdecken können.

1.3 Simulations-Software

Die Kontrolle der Agenten übernimmt AURIS [12], eine an der TU Hamburg-Harburg entwickelte und in C++ geschriebene Steuerungs-Software. AURIS steht für "Autonomous Robot Interaction Simulation". Über eine TCP/IP-Verbindung erhalten die Agenten, für deren Darstellung in der Simulation die Unreal Engine 3 sorgt, Anweisungen von AURIS. Der Simulator des STS-Instituts verwendet als Middleware zwischen Grafik Engine und dem Kontrollprogramm AURIS USARSim. Die Abkürzung steht für "Unified System for Automation and Robot Simulation". Die Software baut auf der Unreal Engine 3 auf. USARSim stellt Roboter, Sensoren und nützliche Tools zur Verfügung. Es ermöglicht aber kein eigenständiges Verhalten der Roboter. Dieses wird erst von AURIS realisiert. Die Befehle an die simulierten UAVs können theoretisch auch von physischen UAVs verstanden werden. Die erläuterten Zusammenhänge werden in Abbildung 1.2, entnommen [12], deutlich. USARSim ist die Schnittstelle zwischen den n Agenten auf der Client-Seite und der Unreal Engine 3 auf der Server-Seite. Um ein weiteres UAV im Simulator nutzen zu können, muss AURIS ebenfalls ein weiteres Mal gestartet werden. Das Verwenden einer Middleware ermöglicht ein zukünftiges Austauschen einer der beiden voneinander entkoppelten Schichten Simulation und Kontrollprogramm [12].

Die Abbildungen 1.3 a) - d) vermitteln einen Eindruck vom Simulator. Die erste Abbildung zeigt ein Szenario, wo UAVs Brände detektieren sollen. Die gezeigte Welt, die von einer zentral gelegenen Brücke charakterisiert wird, ist nur eine von weiteren möglichen Szenarien, die als Umgebung genutzt werden können. Die Stärke der UnrealEngine ist nicht die Darstellung von Organischem, wie Bäumen und Gräsern. Die zweite Abbildung zeigt ein noch nicht vollständig ausgebrochenes Feuer, da von diesem bislang nur starker Qualm aufsteigt . Die Kameraaufnahmen der UAVs besitzen, wie man sieht, einen hohen Detailgrad. Das UAV erkennt den Brand, indem es diese Aufnahmen auswertet. Ein noch nicht abgehobenes UAV wird in c) gezeigt. Deutlich erkennbar sind die Kamera und die 4 Rotorblätter. Gesteuert wird das UAV vom Kontrollprogramm AURIS, welches wie in der letzten Abbildung ersichtlich ist, das Verhalten des UAVs protokolliert und verwaltet. Position, Laufzeit und die Sensorübersicht können abgefragt werden.

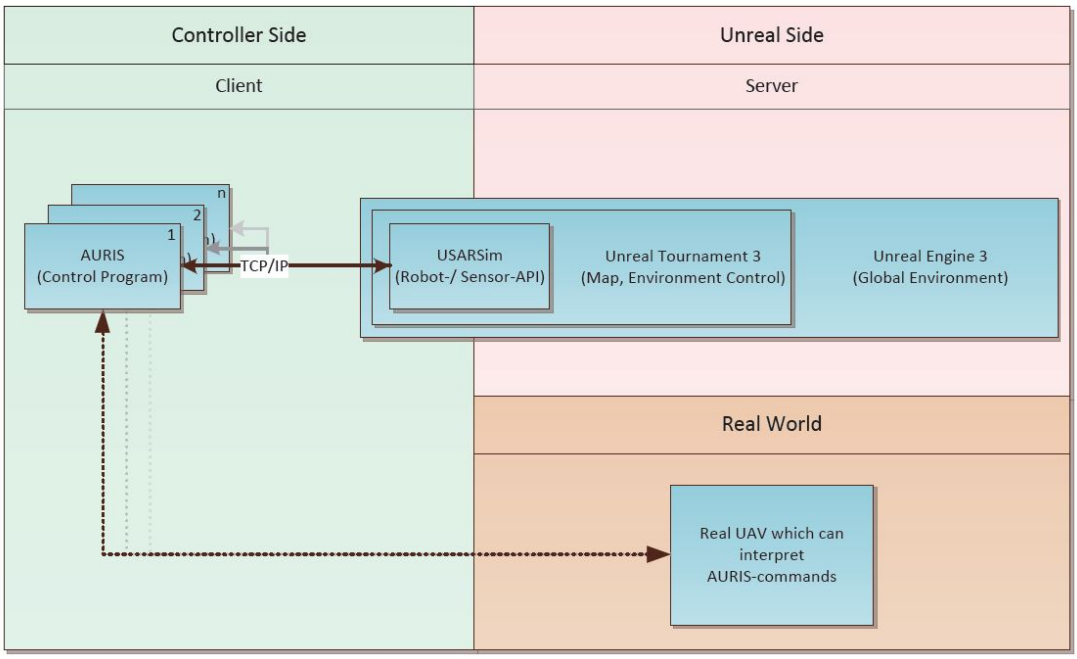


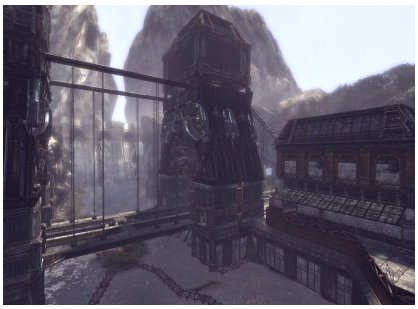
Abbildung 1.2: Softwarezusammenhänge (aus [12])

1.4 Formation

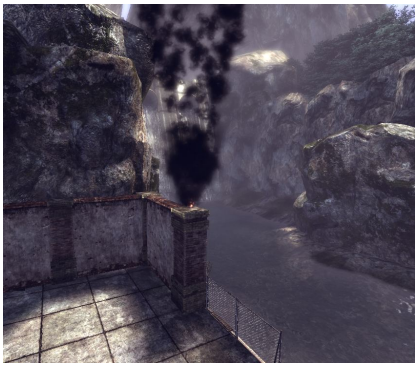
Während des Formationsfluges können die simulierten UAVs die Bodenmessdaten effizienter sammeln. Mehrere Messungen können zeitlich parallel aus untereinander abgestimmten Perspektiven durchgeführt werden und ein UAV muss nicht den kompletten Bereich alleine abdecken. Die für die Aufklärung und Analyse des potentiellen Waldbrandgebietes benötigte Zeit wird reduziert. Wenn 2 UAVs zeitlich nacheinander das gleiche Areal erfassen, sinkt die Fehlerwahrscheinlichkeit der Messdaten, sofern das Areal sich zwischenzeitlich nicht verändert haben kann. Wenn bekannt ist, dass das Areal dynamischen Umwelteinflüssen, wie Funkenflug ausgesetzt ist, können die zu unterschiedlichen Zeitpunkten aufgenommenen Bilder verglichen werden. Von den Funken verbrannte Stellen lassen sich beispielsweise erkennen.

Der Zustandsgraph in Abbildung 1.4 zeigt einen Ausschnitt der Formations-Interaktionsmöglichkeiten. Die Abbildung 1.5 zeigt die einzelnen Zustände. Die Gründe für Formationsänderungen und deren Ablauf werden im Rahmen dieser Arbeit untersucht.

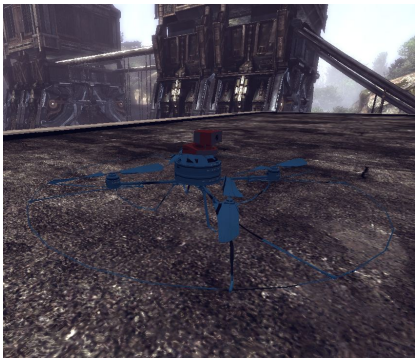
Die Formationen werden der internen Datenbank entnommen und sind daher nicht dynamisch mittels den im Kapitel 3 vorgestellten Henneberg-Operationen erzeugt worden.



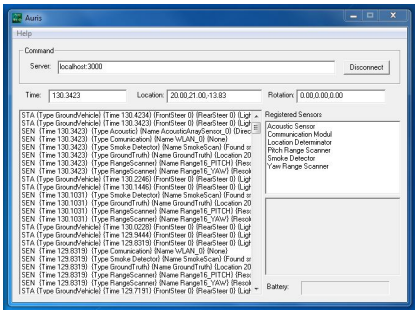
(a) mögliches Szenario



(b) Feuer und Qualm



(c) abflugbereites UAV



(d) Kontrollprogramm AURIS

Abbildung 1.3: Simulator

1. Zustand q_a: Die UAVs bewegen sich unkoordiniert.
2. Zustand q_b: Jeweils 3 UAVs finden sich zu einer Dreiecksformation zusammen.
3. Zustand q_c: Die beiden Dreiecksformationen werden miteinander über einen "Z-Link" verschmolzen.
4. Zustand q_d: Die Formation q_d eignet sich für den Flug zu einem Zielgebiet.
5. Zustand q_e: Die Formation q_e wird gewählt, wenn Daten redundant oder zu einem Zeitpunkt t_0 aus unterschiedlichen Blickwinkeln erfasst werden sollen.

Vor dem Hintergrund, dass eine für den Einsatzbereich optimale Formation gewählt werden sollte, ist eine Formationsänderung von q_c zu q_d oder q_e möglich. Alle Formationsänderungen sind reversibel.

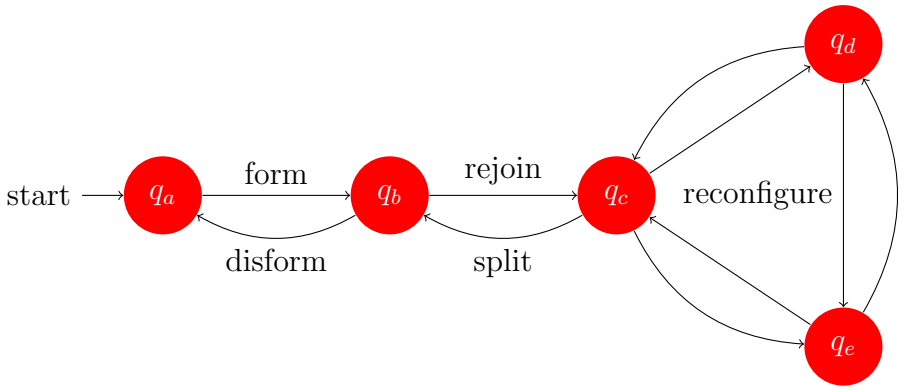


Abbildung 1.4: Formationen-Zustandsgraph (aus [7])

1.5 Agententypen und Architektur

Die kognitiven Agenten führen alle Aktionen unter Maximierung des Nutzens aus. Abbildung 1.6 verdeutlicht das Modell, das von "Utility-Based Agents" ausgeht.

Die Sensoren ermöglichen den Agenten Kenntnisse über die Umwelt. - What the world is like now

Zu der Umwelt sind auch die Zustände der anderen UAVs zu zählen. - What it will be like if I do action A

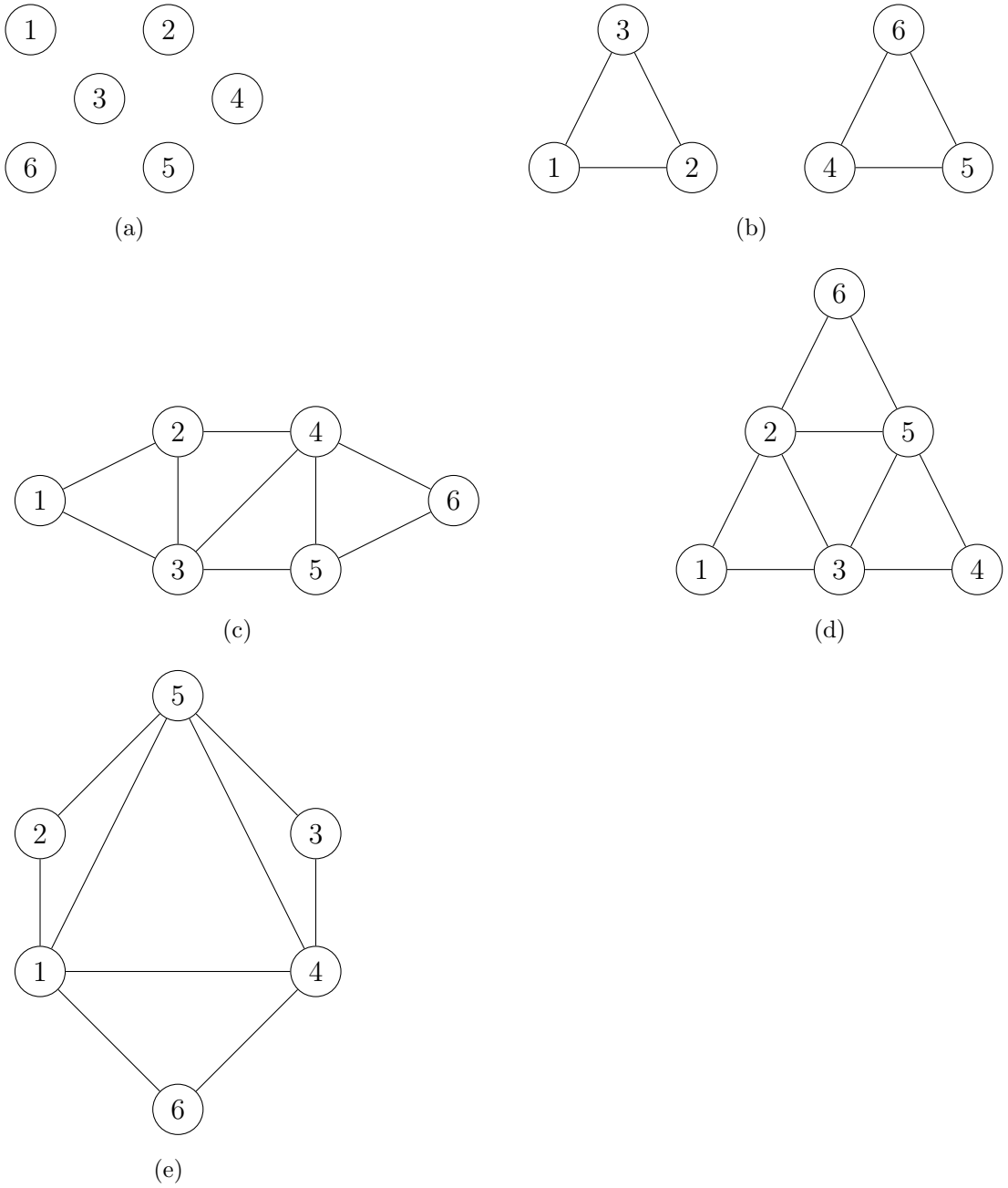


Abbildung 1.5: Zustände des dazugehörigen Zustandsgraphen, der die Aktionsmöglichkeiten einer Formation zeigt.

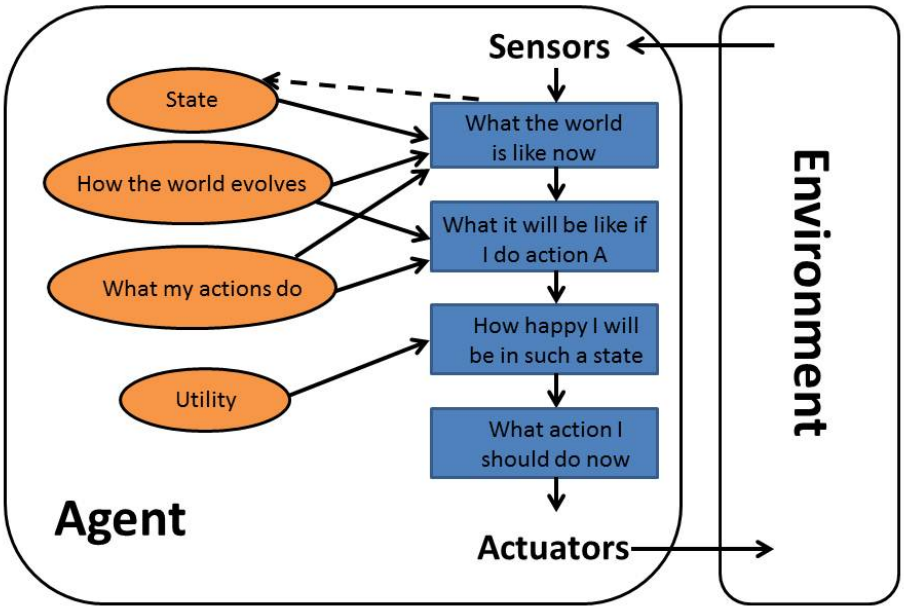


Abbildung 1.6: Utility-Based Agents (aus [22])

Der Grad der Zufriedenheit ist umso höher, desto niedriger die Kosten sind, um den neuen Zustand zu erreichen oder die Aufgabe zu erfüllen. - How happy I will be in such a state -

Beispiel: Positionszuweisung. Die ungarische Methode ist der Weg, über den der Grad an Nutzen maximiert wird.

Eine Aktion wird erst ausgeführt, wenn die optimale Ausführungsweise ermittelt wurde. - What action I should do now

Die folgenden beiden Beispiele verdeutlichen, weshalb es Sinn ergibt, für die Agenten die erläuterte Architektur zu wählen. Bei der Kollisionsverhinderung müssen die Agenten abwägen, ob es langfristig Erfolg versprechend ist, sich um die aus einer Formel geschlussfolgerten 70° zu drehen. Möglicherweise taucht nun ein anderes Hindernis direkt vor einem der Agenten auf und die Formation würde nur ein kurzes Stück fliegen können, bis sie sich wieder drehen müsste. Dieses Beispiel verdeutlicht die Notwendigkeit des vorausschauenden Handelns und das notwendige Verwerfen von Alternativen, die im übernächsten Ausführungsschritt die Situation weiter verschlechtern würden.

Die Umwelt ist nicht statisch. Feuer breitet sich weiter aus und die Länge der Bahn, entlang der die Formation um das Feuer fliegt, wird größer. Abhängig

von dieser Entwicklung besteht die Möglichkeit, die Formation aufzuteilen, damit eine Teilformation gegen den Uhrzeigersinn und die andere Teilformation im Uhrzeigersinn das Feuer umfliegt.

1.6 Weitere Anwendungsbereiche

Formationen bieten sich an, wenn UAVs ein Gelände absichern sollen. Diese Aufgabe wird als "Security patrol" bezeichnet. Wenn die Gruppe etwas entdeckt, lösen sich einige Agenten aus der Formation heraus und untersuchen das Entdeckte, während die anderen UAVs ihre alten Ziele weiterverfolgen.

Agenten nutzen Sensoren, mit denen Hochfrequente Radiowellen [5] empfangen werden können. Die Koordinaten des Ursprungs der Radiowellen lassen sich abschätzen und daher ebenfalls die neuen Zielkoordinaten, um das gesuchte Ziel zu erreichen. Die Formation bewegt sich in diese Richtung, während die Abstandsmesssensoren versuchen das Ziel zu erkennen.

Kapitel 2

Grundlagen und Eigenschaften von Formationen

In diesem Abschnitt wird beschrieben, wie sich eine Formation graphentheoretisch darstellen lässt. Außerdem werden mathematische Methoden vorgestellt, um Graphen auf zuvor eingeführte Formationseigenschaften, wie Starrheit oder Flexibilität, zu überprüfen.

2.1 Graph als abstrakte Struktur

Vor allem aus dem Bereich der diskreten Mathematik stammen Notation, Begriffe und Definitionen. Eine Formation lässt sich mittels eines Graphen $G = (V, E)$ beschreiben. V ist die Knotenmenge und E ist die Kantenmenge.

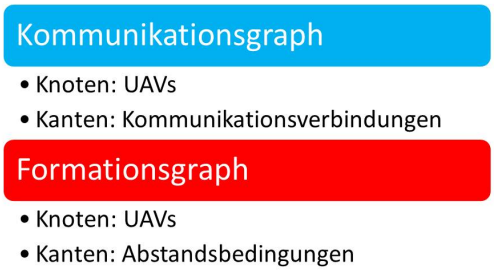


Abbildung 2.1: Kommunikations- und Formationsgraph

Die Knoten des Graphen entsprechen den UAVs und die ungerichteten Kanten stellen die geforderten Abstände dar. Wenn die Kante e_{ij} gerichtet ist, bedeutet dies, dass das UAV i dafür verantwortlich ist, dass die Abstandsbedingung eingehalten wird. Ausschließlich i wird aktiv, wenn Sensoren zur Abstandsbestimmung genutzt werden können. Alle in dieser Arbeit vorgestellten Methoden beschränken sich zuallererst auf den $\mathbb{R}^2$. Es gibt Fälle, bei denen eine Erweiterung für den $\mathbb{R}^3$ sehr einfach möglich ist. Auf diese Ausnahmen wird hingewiesen. In der Literatur [12] wird ebenfalls davon ausgegangen, dass um im $\mathbb{R}^3$ beispielsweise nach Zielen auf dem Boden zu suchen, mehrere 2-dimensionale Schichten übereinander gelegt werden müssen. Diesen Schichten werden unterschiedliche Höhen zugewiesen. Alle in dieser Arbeit in Beispielen dargestellten Formationen befinden sich im $\mathbb{R}^2$.

Das "Leader-UAV" wählt die Formation in Abhängigkeit von dem Missionsziel, Umwelteinflüssen und Gefahrenquellen aus. In einem eigenen Abschnitt wird später erläutert, wie jedes UAV die für ihn wichtigen Abstandsforderungen erhält.

Der Kommunikationsgraph entspricht von der Modellierung her dem Formationsgraphen. Unterschiedlich ist die Bedeutung der Kanten. Bei dem Kommunikationsgraphen handelt es sich bei den Kanten um Kommunikationsverbindungen, also Wege, über die die UAVs mittels W-LAN kommunizieren. Einen übersichtlichen Vergleich zeigt Abbildung 2.1.

2.2 Abstandskorrektur zwischen 2 benachbarten UAVs

In diesem Abschnitt wird erläutert, wie die Regelung der Abstände innerhalb einer Formation abläuft. Das System (G, p), im Englischen "Framework" genannt, setzt sich zusammen aus dem Formationsgraphen und der Abbildung $p : V \to \mathbb{R}^m$ [11]. m ist die Dimension des Raumes. Den Knoten V des Formationsgraphen werden Koordinaten im m-dimensionalen Raum zugewiesen. (G, p) ist die Realisierung von G im $\mathbb{R}^m$ [6].

$x_i,\ i = 1, 2, 3$ ist der Ortsvektor des Agenten i. Der geforderte normierte Abstand

zwischen UAV 1 und UAV 2 sei d_{12}, zwischen Agent 2 und 3 sei dieser d_{23} und zwischen UAV 3 und 1 d_{31}.

$$D = \{d_{ij} \in \mathbb{R} | d_{ij} > 0,\ i, j = 1, ..., n,\ i \neq j\} \tag{2.1}$$

D ist die Menge, welche die Abstandsvorgaben enthält. Der Abstand muss immer größer 0 sein, damit 2 Knoten nicht identisch sind. Bei ungerichteten Kanten ist $d_{ij} = d_{ji}$.

Wenn alle Abstandsbedingungen erfüllt sind, ist die geforderte Formation realisiert [2]. Entsprechend obiger Durchnummerierung sind die mittels Sensoren ermittelten relativen Abstände zwischen den Agenten $z_{12} = x_1 - x_2, z_{23} = x_2 - x_3 und z_{31} = x_3 - x_1$. Die geforderte Formation ist erreicht, sofern der Fehler, also die Differenz zwischen gefordertem und momentanem Abstandswert, gegen 0 konvergiert ist [1].

$$\lim_{t \to \infty} \|z_{ij}(t)\| - d_{ij} = 0,\ i, j = 1, 2, 3,\ i \neq j \tag{2.2}$$

Es wird keine Obergrenze für die Höhe der initialen Abstandsfehler definiert. Jedoch reduzieren bereits nah beieinanderliegende Ist- und Soll-Werte die Konvergenzzeit. Die Abbildung 2.2 *b*) zeigt eine Dreiecksformation, deren Abstandsbedingungen nicht mehr erfüllt sind, nachdem beispielsweise ein starker Windstoß die einzelnen Agenten von ihrer Trajektorie abgebracht hat.

Jedes UAV besitzt eine Masse. Der Gesamtmassenmittelpunkt der Formation, Schwerpunkt, soll sich auf einer vordefinierten Bahnkurve, Trajektorie bewegen. Es wird idealerweise erst einmal davon ausgegangen, dass keine Hindernisse die UAVs zu einem Ausweichmanöver veranlassen.

Abbildung 2.2: *a*) zeigt eine nicht gestörte Formation, wohingegen *b*) eine inkorrekte Formation zeigt

2.3 Modellierung der Flugbahn

Die Flugbahn ist in mehrere unterschiedlich lange, lineare Teilstücke unterteilt. Zu durchfliegende Kurven werden ebenfalls von mehreren kurzen, linearen Teilstücken modelliert. Die Geschwindigkeit wird beim Durchfliegen eines Kurvenabschnitts angepasst, damit das UAV aufgrund der Zentrifugalkraft nicht aus der Kurve gedrängt wird. Zwei weitere Möglichkeiten gibt es, eine Kurve zu durchfliegen. Während das UAV entlang der Geraden n fliegt, soll es sich bereits drehen, sodass es sich weiter geradeausbewegen kann, sobald es den Startpunkt der Geraden $n + 1$ erreicht. Ein Anhalten des UAVs, um die Drehung ausführen zu können, würde somit entfallen. Die logische Schlussfolgerung ist, dass eine Interpolation der Geradenanfangspunkte ebenfalls möglich ist, um eine Bewegungskurve zu erhalten. Erfolgversprechend ist die Spline-Interpolation. Der Vorteil von Splines ist, dass alle Übergänge krümmungsruckfrei sind.

2.4 Starrheit

Eine Formation bleibt erhalten, wenn alle Agenten den Abstand zu bestimmten Nachbarn nicht ändern. Die Abstände zwischen nicht zueinander adjazenten UAVs bleiben dann ebenfalls konstant. Sind alle geforderten, vordefinierten Konsistenzbedingungen, also die Abstandsbedingungen d_{ij}, für eine sich bewegende Formation erfüllt, so wird von Starrheit gesprochen [5]. Die einzuhaltenden Abstände, dargestellt mittels gewichteten Kanten, können auch formal in Nebenbedingungen festgehalten werden. Die folgende Aussage, entnommen [6], definiert "Starrheit". Ein System ist starr, wenn nur die trivialen Bewegungen "Rotation um die z-Achse" und "Translation im Raum" möglich sind. Eine Formation bewegt sich auf einer starren Flugbahn, wenn nicht nur zueinander adjazente UAVs einen unveränderlichen Abstand einhalten, sondern diese Bedingung ebenfalls für jedes beliebige Knotenpaar des Graphen der Formation gilt [2]. Innerhalb einer starren Formation kann es zu keiner Kollision zweier Agenten kommen und die benachbarten UAVs bleiben gewiss innerhalb des kreisförmigen Kommunikationsbereiches. Bei nicht starren Formationen könnte sich ein UAV zu weit vom Nachbarn entfernen, sodass es über W-LAN nicht mehr erreicht wird. Laut [2] ist ein Graph minimal starr, wenn dieser nur solange starr bleibt, wie keine Kante entfernt wird und damit auch keine Abstandsforderung verloren geht. Mindestens eine Kante geht verloren, wenn die Knotenmenge um ein Element reduziert wird. Laut [6] ist das Gegenteil von "minmialer Starrheit" "redundante Starrheit". In dem Abschnitt "Henneberg Folge: Erweiterung eines minimal starren Graphen", wird ein Verfahren gezeigt, das einen Graphen mit n Knoten

liefert, der über eine minimale Anzahl an Verbindungskanten verfügt. Ein starrer Graph sollte nur für Formations- und nicht für Kommunikationsgraphen gefordert werden. Die UAVs innerhalb des Kommunikationsgraphen können aufgrund des vorliegenden minimal starren Formationsgraphen ihre relative Positionierung zueinander nicht ändern. Bei Formationsgraphen nimmt der Rechenaufwand ab, da weniger Abstandsmessungen vorgenommen werden müssen. Diese gewonnene Zeit kann daher für ein erneutes Überprüfen der Abstände genutzt werden, sodass der Abstandsfehler kleiner wird. Bei dem Kommunikationsgraphen lässt sich der W-LAN-Funkverkehr auf ein Minimum reduzieren, wenn mittels Dijkstra-Algorithmus der kürzeste Weg zu jedem Knoten berechnet wird.

Fehlende Redundanz führt auf der anderen Seite zu einer höheren Wahrscheinlichkeit, dass Übertragungsfehler zu Stande kommen, die dann weitergegeben werden. Diese Gefahr der Fehlerfortpflanzung muss auf eine andere Weise wieder kompensiert werden. In [1] wird diskutiert, ob für die Abstandseinhaltung innerhalb einer Formation eine symmetrische oder eine asymmetrische Kontrollstruktur zu Grunde gelegt werden sollte. Liegt einer Formation eine symmetrische Kontrollstruktur zu Grunde, ist der Graph nicht gerichtet, siehe Abbildung 2.3. Ein gerichteter Graph, Digraph genannt, lässt auf eine asymmetrische Kontrollstruktur schließen. Nach einem Vergleich der beiden potentiellen Kontrollstrukturen wird entschieden, welche für das UAV-Projekt unter den Aspekten Zuverlässigkeit und Effizienz die geeignetere ist.

Abbildung 2.3: in a liegt der Formation eine symmetrische Kontrollstruktur zu Grunde und in b eine asymmetrische

2.4.1 Symmetrische Kontrollstruktur

Bei einem der Formation zu Grunde liegenden ungerichteten Graphen sendet Agent i seine auf dem GPS-Signal beruhende Standortinformation, während Agent j ebenfalls seine Standortinformation sendet. Der Empfänger erhält die Position des Senders. Die zweite Möglichkeit besteht darin, dass beide Agenten mittels Abstandsmessungen über Sensoren die gegenseitige relative Positionierung ermitteln.

2.4.2 Asymmetrische Kontrollstruktur

Agent j sendet seine Standortinformation, die Agent i empfängt. Agent i sendet keine Informationen. Diese klare Rollenverteilung, Sender und Empfänger, wird bei einem gerichteten Graphen dargestellt, indem die Kante an einem Knoten i beginnt und am benachbarten Knoten j, dargestellt mit Pfeilspitze endet. Die Pfeilspitze zeigt zum Sender. Wenn i ausschließlich den Abstand über Sensoren bestimmt, bekommt davon der Nachbar j nichts mit. j ist daher unbeteiligt.

2.4.3 Vergleich der Kontrollstrukturen

Bei einer asymmetrischen Kontrollstruktur muss den beteiligten Agenten klar sein, ob sie Sender oder Empfänger sind. Der Graph muss nicht zyklisch sein, damit eine asymmetrische Kontrollstruktur möglich ist. Bei jeder Formation ohne "Leader-UAV" ist der Ausgangsgrad von jedem Knoten größer 0. Die Formation, dargestellt in der Abbildung 2.3 b) besitzt kein "Leader-UAV". Die asymmetrische Kontrollstruktur hat gegenüber der symmetrischen den Vorteil , dass die Anzahl an Kommunikationsvorgängen halbiert ist. Daher ist die Vermutung, dass die Fehleranfälligkeit im Gegenzug zugenommen hat.

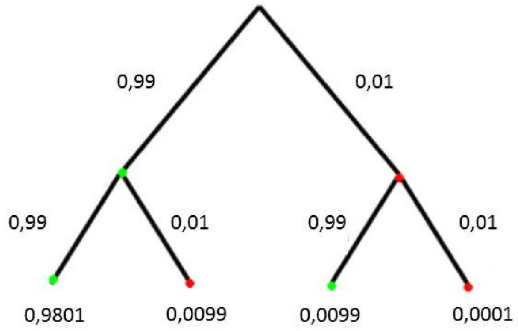


Abbildung 2.4: Baumdiagramm

Es folgt ein Beispiel, das wenige Parameter berücksichtigt und daher nur ein qualitatives Urteil darstellen kann. Abbildung 2.4 zeigt ein Baumdiagramm für die Berechnung der Fehlerwahrscheinlichkeit bei der Datenübertragung mittels symmetrischer Steuerung. Angenommen wird, dass die Wahrscheinlichkeit für eine fehlerhafte Übertragung bei $p = 0,01$ liegt. Wegen fehlender Redundanz liegt bei einer asymmetrischen Kontrollstruktur bei 1 maligem Senden eine Fehlerwahrscheinlichkeit von $0,01$ vor. Die Wahrscheinlichkeit dafür, dass bei 1 maligem Korrigieren des Abstandes bei einer symmetrischen Kontrollstruktur mindestens ein Fehler bei einem der beiden sendenden Agenten auftritt ist $0,01 \cdot 0,01 + 2 \cdot (0,01 \cdot 0,99) = 0,0199$. Bei einer symmetrischen Steuerung kommt es also häufiger zu Fehlern, da es mehr Akteure gibt. Die symmetrische Steuerung arbeitet schneller, benötigt dafür jedoch einen doppelt so großen Aufwand. Im Grunde führt diese Betrachtungsweise mit dem Schwerpunkt, zu entscheiden, welche Kontrollstruktur weniger fehleranfällig ist, nicht zu einer eindeutigen Entscheidung. Die beiden Akteure der symmetrischen Steuerung erzeugen keine Redundanz. In den Berechnungen kann der Einfluss und die Folge des Übertragungsfehlers nicht berücksichtigt werden. Bei nicht gesendeten Daten bewegt sich das auf eine Nachricht wartende UAV nicht. Wenn Bits verfälscht werden, kann dies dazu führen, dass falsche Positions-Angaben empfangen werden.

2.5 Dreiecksformation

Eine Dreiecksformation, der eine asymmetrische Kontrollstruktur zu Grunde liegt, hat gegenüber komplexeren Formationen einen entscheidenen Vorteil. Das "Leader-UAV" kann eindeutig identifiziert werden, wenn der Graph nicht zyklisch ist und die Formation ist sehr einfach erweiterbar, ohne dass das Aussehen sich ändert, da ein UAV einfach eine Kante auftrennen kann.

Die geforderte Dreiecksformation wird entweder nur über Abstandsbestimmungen realisiert [5] oder alternativ werden 2 Winkel sowie ein Abstand gefordert. In der Literatur findet letzterer Ansatz keine Beachtung. Ausschließlich Abstandsforderungen werden stattdessen gestellt, da die Hinzunahme des Winkels genauere Sensoren voraussetzt, die mit der benötigten Genauigkeit für den Awendungsbe-

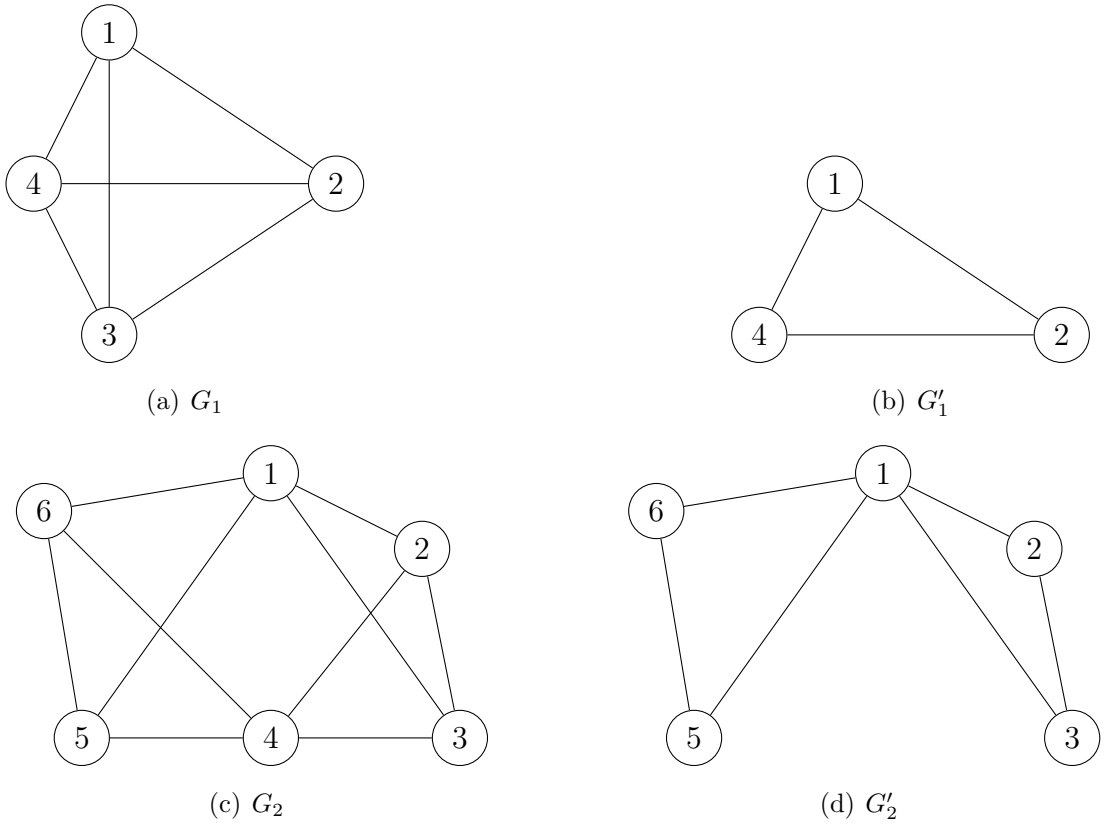
reich nicht zur Verfügung stehen. Jedes UAV versucht die von ihm geforderten Bedingungen zu erfüllen.

Formationen, deren Mitglieder unempfindlich auf Umwelteinflüsse, wie Regen oder Wind reagieren, sind anderen vorzuziehen. Ein Merkmal einer robusten Formation ist ebenfalls, dass diese auf den Verlust von einem oder mehreren Gruppenmitgliedern so reagiert, dass die Starrheit erhalten bleibt. Darauf wird im Abschnitt "In der Anwendung oft verwendete Formationstypen" eingegangen.

2.6 Redundante Starrheit

Ein Graph ist redundant starr, wenn das Entfernen einer Kante zu keinem Verlust der Starrheit führt [19], [9]. Ein Synonym für redundant starr ist 2-rigid. Als "circuit" wird ein Graph bezeichnet, wenn das Entfernen einer Kante den Graph minimal starr werden lässt. Ein starrer Graph ist "birigid", wenn dieser auch starr bleibt, wenn ein Knoten und alle zu diesem Knoten inzidenten Kanten entfernt werden.

Die Anzahl an Kanten, die mindestens entfernt werden muss, damit ein Graph nicht länger starr ist, wird mit $R_e(G)$ abgekürzt. $R_e(G)$ ist bei nicht starren Graphen 0. Im Umkehrschluss wird ein Graph als "n-Kanten-starr" bezeichnet, sofern ein Entfernen von $n-1$ beliebigen Kanten nicht zu einem Verlust der Starrheit führt. $R_e(G)$ ist bei nicht starren Graphen und bei Graphen, deren Knotenanzahl $V < 3$ ist 0. Analog zu dieser Kantenorientierten Betrachtungsweise ist auch eine Knotenorientierte möglich. Die Anzahl an Knoten, die mindestens entfernt werden muss, damit ein Graph nicht länger starr ist, wird mit $R_v(G)$ abgekürzt. Eingeschränkt wird dieser Satz von der Forderung, dass der entstehende Graph G' mindestens über 3 Knoten verfügen muss. $R_v(G)$ ist bei nicht starren Graphen und bei Graphen, deren Knotenanzahl $V < 3$ ist 0. Ein Graph wird als "n-Knoten-starr" bezeichnet, sofern ein Entfernen von $n-1$ beliebigen Knoten nicht zu einem Verlust der Starrheit führt und G' über mindestens 3 Knoten verfügt. Zwei Bespiele, entnommen [19], für die Überprüfung eines Graphen auf n-Knoten-Starrheit sind in Abbildung 2.5 dargestellt. Wenn aus dem Graphen G_1 von Abbildung 2.5 a) ein Knoten entfernt wird, dieser also auf n=2 hin getestet wird, entsteht ein starrer Graph, nämlich eine Dreiecksformation. Wird hingegen dem Graphen G_2 ein Knoten entnommen, so ist G_2' nicht länger starr. G_1 ist also 2-Knoten-starr und G_2 ist nicht 2-Knoten-starr.



(a) G_1 (b) G_1'

(c) G_2 (d) G_2'

Abbildung 2.5: Überprüfung von Formationen auf 2-Knoten-Starrheit im $\mathbb{R}^2$

2.7 Prüfen einer Formation auf Starrheit

Es gibt zwei Ansätze, um nachzuweisen, ob eine Formation starr ist oder nicht. Die Eigenschaften der Starrheitsmatrix A, aufgestellt für den Formationsgraphen G, lassen eindeutige Rückschlüsse zu. Der andere Weg führt über die Überprüfung von Lamans Theorem. Im folgenden wird der mathematische Weg [1], der Erkenntnisse aus der Linearen Algebra nutzt, zuerst erläutert.

2.7.1 Aufstellen und Auswerten der Starrheitsmatrix

Der Formationsgraph $G = (V, E, p)$ ist nicht notwendigerweise planar. p enthält die Koordinaten der Knoten V im m-dimensionalen Raum.

Die Starrheitsmatrix ist eine $E \times nm$ Matrix. Die Variable n steht für die Anzahl der Knoten V. Die Formation bewegt sich im m-dimensionalen Raum. Die Anzahl der Kanten von $G = (V, E)$ spiegelt sich in der Anzahl der Zeilen wieder. Die Anzahl der Spalten ist das Produkt aus Anzahl der beteiligten Agenten und Dimension des Raumes. i ist der Index des Knotens. Die Einträge einer Zeile hängen von genau einer Kante $g = e_{(i,j)}$ und deren inzidenten Knoten ab [6]. Die Kanten können beliebig mit dem Index g durchnummeriert werden und somit werden auch die Zeilen der Matrix in beliebiger Reihenfolge angeordnet. Die Elemente $a_{g,(i-1)m+1}$ bis $a_{g,im}$ enthalten die Differenzen $(p_i - p_j)^T$ und die Differenzen $(p_j - p_i)^T$ werden in die Elemente $a_{(g,j-1)m+1}$ bis $a_{g,jm}$ eingetragen. Alle anderen Einträge sind 0.

$$(p_i - p_j)^T = (x_1 - x_2,\ y_1 - y_2,\ z_1 - z_2) \tag{2.3}$$

ist der Differenzvektor im $\mathbb{R}^3$. Man beachte, dass die Anzahl der benötigten Spalten abhängig von der Dimension ist.

Im $\mathbb{R}^2$ sind beispielsweise für die Kante $g = e_{1,2}$ die Elemente $a_{g,(i-1)m+1} = a_{g,(1-1)\cdot 2+1} = a_{g1}$ und $a_{g,im} = a_{g,1\cdot 2} = a_{g2}$ sowie $a_{g,(j-1)m+1} = a_{g,(2-1)2+1} = a_{g3}$ und $a_{g,jm} = a_{g,2\cdot 2} = a_{g4}$ auszuwählen.
Der Rang und die Dimension der Starrheitsmatrix sagen aus, ob die Formation starr ist. Wenn der Rang der Matrix mindestens $2\|V\| - 3$ ist, dann ist der Graph im $\mathbb{R}^2$ starr. Im $\mathbb{R}^3$ kommt für jeden Agenten eine Spalte hinzu und der Rang muss mindestens $3\|V\| - 6$ sein, damit Starrheit vorliegt. Die allgemeine Formel lautet

$$\text{rank}_c = m \cdot n - \frac{m \cdot (m+1)}{2} \text{ bzw. } m \cdot n - \left(\frac{m+1}{2}\right) \tag{2.4}$$

Wenn der Rang der Matrix mindestens gleich rank_c ist, dann ist der Graph starr. Um zusätzlich minimale Starrheit zu erfüllen, muss der Rang der Matrix genau gleich rank_c sein [6], [13].

2.7.2 Formationsbeispiele im $\mathbb{R}^2$ für starre, minimal starre und flexible Formationen

Die folgenden Beispiele sind [5] entnommen.

Abbildung 2.6: flexible Formation

Die Formation in Abbildung 2.6 a) ist nicht starr, sondern flexibel, da die Knoten 4 und 1 entlang der z-Achse gedreht werden können, wenn auf sie eine leichte Kraft ausgeübt wird, siehe Abbildung 2.6 b). Von dieser Formationsänderung bekommen die Agenten nichts mit, weil die Abstandsvorgaben weiterhin erfüllt werden. Der Winkel zwischen Knoten 3 und 4 ändert sich und somit ist eine andere Formation entstanden. In der Abbildung 2.6 b) sind die Positionen der Kanten und Knoten vor der Rotation rot eingezeichnet.

Bei der Formation in Abbildung 2.7 liegt Starrheit vor, da ein Anstoßen der Konstruktion zu keiner Änderung führt, die nicht von den Agenten wieder korrigiert wird. Sofern die Kante zwischen Knoten 4 und Knoten 2 entfernt wird, liegt wieder die flexible Formation aus 2.6 a) vor. Daher ist diese Formation nicht nur starr, sondern sogar minimal starr.

Die Formation in Abbildung 2.8 ist starr, jedoch nicht minimal star. Dies erkennt man daran, dass es 2 sich überschneidende Kanten im Zentrum der Formation gibt, von denen auch eine ausreicht, um die minimal starre Formation aus Abbildung 2.7 zu erhalten

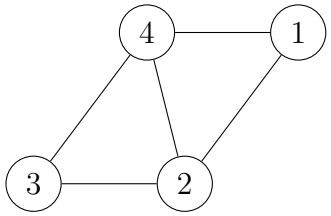


Abbildung 2.7: minimal starre Formation

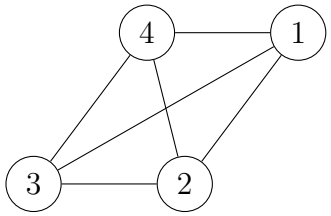


Abbildung 2.8: starre Formation

Die Überlegungen sind auch algebraisch überprüfbar. Für 2.6 a) sieht die Starrheitsmatrix wie folgt aus.

$$\begin{bmatrix} x_1 - x_2 & y_1 - y_2 & x_2 - x_1 & y_2 - y_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_2 - x_3 & y_2 - y_3 & x_3 - x_2 & y_3 - y_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_3 - x_4 & y_3 - y_4 & x_4 - x_3 & y_4 - y_3 \\ x_1 - x_4 & y_1 - y_4 & 0 & 0 & 0 & 0 & x_4 - x_1 & y_4 - y_1 \end{bmatrix} \tag{2.5}$$

Für die Einträge in der ersten Zeile werden die zu der Kante e_{12} inzidenten Knoten 1 und 2 betrachtet. Der Rang der Matrix wird bestimmt, indem die Matrix unter Verwendung des Gaußschen Elimininationsverfahren in eine äquivalente Matrix umgeformt wird. Die Zeilen, die nicht ausschließlich 0-en als Komponenten aufweisen, werden gezählt, um den Rang zu bestimmen.

Der Rang der Starrheitsmatrix ist 4 und die Anzahl der Knoten n bzw. V ist ebenfalls 4.
Das Einsetzen der Variablen in die Gleichung (2.4) liefert

$\operatorname{rank}_c = 2V - 3 = 2 \cdot 4 - 3 = 5$
Die Matrix ist nicht starr, da der Rang der Starrheitsmatrix zu klein ist. Der Rang hätte mindestens 5 sein müssen, ist jedoch nur 4.

Für die Formation aus Abbildung 2.8 wird die Starrheitsmatrix aufgestellt.

$$\begin{bmatrix} x_1-x_2 & y_1-y_2 & x_2-x_1 & y_2-y_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & x_2-x_3 & y_2-y_3 & x_3-x_2 & y_3-y_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_3-x_4 & y_3-y_4 & x_4-x_3 & y_4-y_3 \\ x_1-x_4 & y_1-y_4 & 0 & 0 & 0 & 0 & x_4-x_1 & y_4-y_1 \\ x1-x3 & y1-y3 & 0 & 0 & x3-x1 & y3-y1 & 0 & 0 \\ 0 & 0 & x_2-x_4 & y_2-y_4 & 0 & 0 & x_4-x_2 & y_4-y_2 \end{bmatrix} \tag{2.6}$$

Im Vergleich zu der Starrheitsmatrix, aufgestellt für die Formation aus Abbildung 2.6 a), besteht diese Matrix aus 2 zusätzlichen Zeilen aufgrund der 2 zusätzlichen Kanten des Formationsgraphen. Die Anzahl der Knoten V ist weiterhin 4.

Der Rang der Starrheitsmatrix ist 6.
Wenn die Variablen in die Bedingung (2.4) eingesetzt werden, liefert die kurze Rechnung folgendes.

$\text{rank}_c = 2V - 3 = 2 \cdot 4 - 3 = 5$
Die Matrix ist starr, jedoch nicht minimal starr, da dafür der Rang um 1 zu groß ist. Der Rang der Matrix ist 6, für minimale Starrheit hätte dieser jedoch nur 5 sein dürfen.

2.8 Laman-Graphen

In diesem Abschnitt wird ein zweites Verfahren erläutert, das nachweist, ob es sich bei dem Formationsgraphen um einen minimal starren Graph handelt. Minimal starre Graphen sind Laman-Graphen. Allgemein unterscheidet man Graphen danach, ob es sich um "sparse graphs" oder um "dense graphs" handelt. Laman-Graphen sind zu der Familie der "sparse graphs" zu zählen. Da Formationsgraphen mit so wenig Kanten wie möglich auskommen sollen, gibt es diesbezüglich keine Widersprüche.

Es wird überprüft, ob es sich bei dem Graphen um einen Laman-Graphen handelt. Wenn dies zutrifft, handelt es sich ebenfalls um einen minimal starren Graphen. In [8] wird vorgeschlagen, den Graphen daraufhin zu prüfen, ob mit diesem eine "3Tree2-partition"konstruiert werden kann. Diese Konstruktion ist die Bedingung für einen Laman-Graphen.

2.8.1 3Tree2-partition

Der Formationsgraph wird in 3 Bäume $T_1, T_2 und T_3$ partitioniert. Deren Kantenmengen sind disjunkt [13]. Jeder Knoten ist Element von genau 2 Bäumen, siehe Abbildung 2.9. Es handelt sich bei der "3Tree2-partition" um eine "proper 3Tree2-partition", wenn sich die Bäume T_i in ihrem Span unterscheiden. Sonst wird die "3Tree2-partition" als "non-proper" bezeichnet, siehe Abbildung 2.10. Unter Ausnutzung von Erkenntnissen des Mathematikers Tay wird in [26] bewiesen, dass die 3Tree2-partition "proper" sein muss, wenn der Formationsgraph G = (V,E) starr, genauer genommen "generically rigid", sein soll.

Eine 3Tree2-partition wurde gefunden, wenn die Knotenanzahl und die Kantenanzahl von jedem Baum wie folgt in Relation stehen:

Jeder Baum T erfüllt

$$E(T) = V(T) - 1 \tag{2.7}$$

Diese Forderung wird für den Formationsgraphen in Abbildung 2.9 überprüft.

Partition	Anzahl an Knoten	Anzahl an Kanten
blau	4	3
rot	6	5
grün	2	1

Die Anzahl an Knoten in jedem Baum ist immer um 1 größer als die Anzahl an Kanten.

In [13] wird geschlussfolgert, dass das Verhältnis zwischen Kanten und Konten bei Graphen die eine 3Tree-partition erlauben, genau

$$E(G) = 2V(G) - 3 \tag{2.8}$$

sein muss. Für das betrachtete Beispiel gilt $E = 9$ und $V = 6$ und damit wird diese Bedingung erfüllt.

2.8.2 Lamans Theorem

Der Formationsgraph $G = (V, E)$ ist starr, wenn auf ihn Lamans Theorem anwendbar ist. Das Theorem wurde 1970 aufgestellt.

Lamans Theorem zur Überprüfung von G auf Starrheit: Es existiert ein Untergraph $G' = (V, E')$ mit V Knoten und

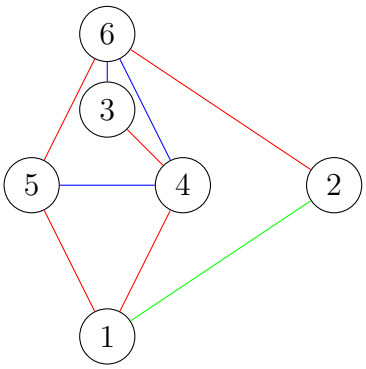
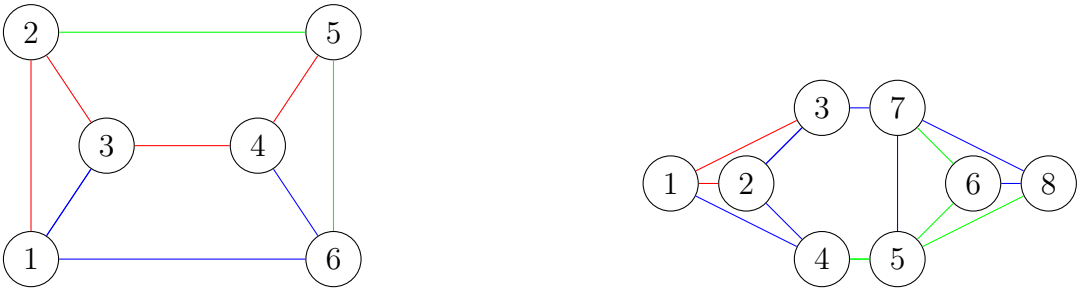


Abbildung 2.9: Die unterschiedlich gefärbten Kanten zeigen die drei aus dem Graphen gebildeten Bäume der 3Tree2-partition.



(a) (b)

Abbildung 2.10: die linke Abbildung zeigt eine "proper 3Tree2-partition", wohingegen rechts eine "non-proper 3Tree2-partition" dargestellt wird

$$E' = 2V - 3 \tag{2.9}$$

Kanten.

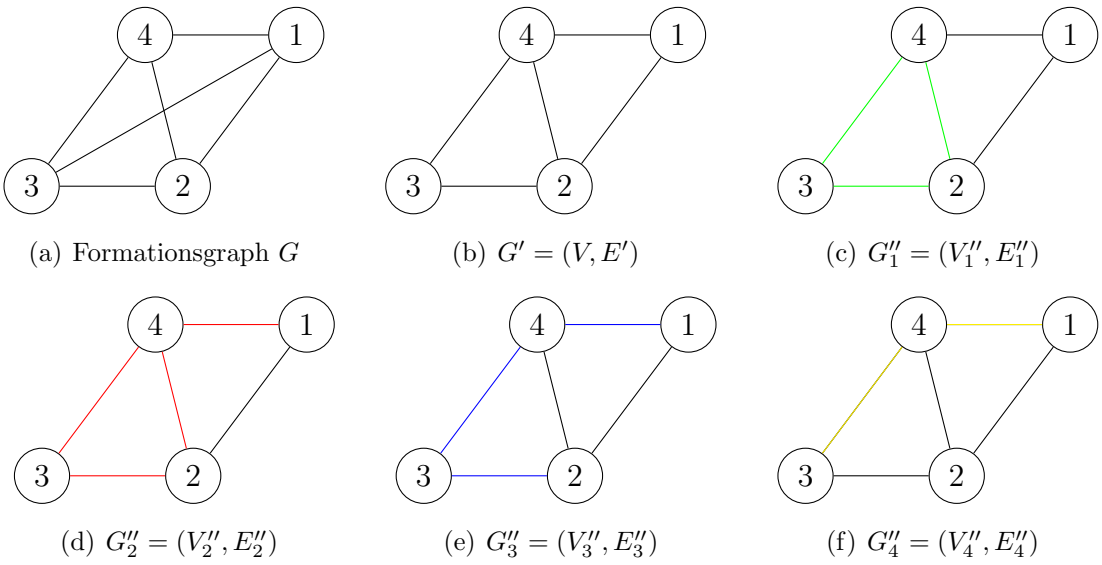
Jeder mögliche Teilgraph $G'' = (V'', E'')$ von G' erfüllt die Kantenbedingung

$$E'' \leq 2V'' - 3, \tag{2.10}$$

wenn V'' die Endpunkte der Kanten E'' in G'' sind.

Wie in [2] definiert ist, sind die Gleichungen (2.9) und (2.10) die Bedingungen für das Vorliegen von minimaler Starrheit. In diesem Fall wird G' daraufhin untersucht.

Eine notwendige und hinreichende Bedingung für starre Graphen ist nach [14], dass es einen Teilgraphen G' geben muss, der minimal starr ist. Diese Aussage spiegelt sich in Lamans Theorem wieder.



(a) Formationsgraph G (b) $G' = (V, E')$ (c) $G''_1 = (V''_1, E''_1)$

(d) $G''_2 = (V''_2, E''_2)$ (e) $G''_3 = (V''_3, E''_3)$ (f) $G''_4 = (V''_4, E''_4)$

Abbildung 2.11: Laman's Theorem.

Die folgenden Berechnungen werden für den Graphen $G = (V, E)$ aus Abbildung 2.11 a) durchgeführt. Lamans Theorem wird genutzt, um zu überprüfen, ob G starr ist oder nicht.

Der Formationsgraph besitzt $V = 4$ Knoten. Es soll gezeigt werden, dass es einen Untergraphen gibt, für den (2.9) erfüllt wird. In (2.9) wird die Anzahl an Knoten, hier 4, eingesetzt:

$E' = 2V - 3 = 2 \cdot 4 - 3 = 5$

Der Graph G', den es geben soll, muss über genau 5 Kanten verfügen. In Abbildung 2.11 b) ist der Graph G' dargestellt.

Die Abbildungen 2.11 c) bis f) zeigen 4 von allen zu testenden Untergraphen $G_i'' = (V_i'', E_i'')$. Für alle Untergraphen, die es gibt, muss (2.10) gelten.

G_i''	V_i''	E_i''	Rechnung
1	3	3	$E'' \leq 2V'' - 3 = 2 \cdot 3 - 3 = 3$
2	4	4	$E'' \leq 2V'' - 3 = 2 \cdot 4 - 3 = 5$
3	4	3	$E'' \leq 2V'' - 3 = 2 \cdot 4 - 3 = 5$
4	3	2	$E'' \leq 2V'' - 3 = 2 \cdot 3 - 3 = 3$

Der Graph G_1'' darf maximal 3 Kanten besitzen und diese Forderung wird, wie man in Abbildung 2.11 c) sieht, auch erfüllt, da genau 3 Kanten vorliegen. Der Graph G_2'' darf maximal 5 Kanten besitzen und diese Forderung wird, wie man in Abbildung 2.11 d) sieht, auch erfüllt, da nur 4 Kanten vorliegen.

2.9 Persistente Formation

Wenn der Formation eine asymmetrische Kontrollstruktur zu Grunde liegt, kann der Fall eintreffen, dass die Formation starr ist, jedoch nicht alle Abstandsbedingungen eingehalten werden können. Die Abstandsbedingungen sind dann nicht erfüllbar mit den momentanen Positionen der UAVs im Raum. Es können nicht alle vordefinierten Konsistenzbedingungen erfüllt werden. Der Formationsgraph ist dann nicht persistent. Das Beispiel in der Abbildung 2.12, entnommen [5], verdeutlicht dies. Die gezeigte Formation, die im Rahmen dieser Arbeit schon häufiger verwendet wurde, ist bekanntlich starr. Der Agent 4 ist in der Lage die Abstände zu den Agenten 1 und 2 einzuhalten. Für die Erfüllung der Abstandsbedingung zwischen Agent 4 und 3, ist ebenfalls Agent 4 verantwortlich. Da aber Agent 3 ausschließlich den Abstand zu einem Nachbarn regeln muss, ist das UAV frei in seiner Bewegung. Wenn auf Agent 3 eine leichte Kraft ausgeübt wird, korrigiert der Agent den Abstand, bewegt sich dabei jedoch zu einer neuen Position, die auf einem Kreis liegt, dessen Mitelpunkt Agent 2 ist. Die Folge ist, dass Agent 3 einen Abstand zu Agent 4 erzeugt, den Agent 4 nicht korrigieren kann, ohne dass die Abstände zu den Agenten 1 und 2 wieder fehlerhaft werden. Dieser von Agent 4 allein nicht korrigierbare Abstand wird in der Abbildung mittels rot eingefärbter Kante e_{43} dargestellt. Die Folge ist, dass die Lyapunovfunktion nicht gegen 0 konvergieren kann. Wenn eine symmetrische Kontrollstruktur vorliegen täte, würden beide UAVs an der Korrektur arbeiten und das Abstandsproblem

lösen können.

(a) Agent 4 erfüllt seine verantwortlichen Abstandsbedingungen

(b) Agent 3 erfüllt d_{32} auch von einer anderen Position aus

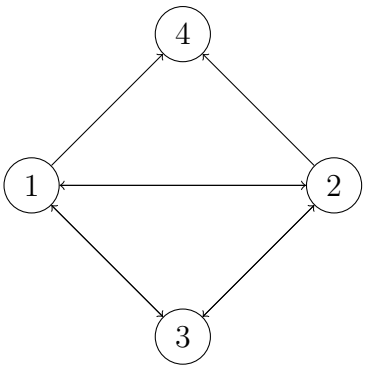
Abbildung 2.12: starrer, nicht persistenter Graph

Die folgenden Überlegungen beziehen sich nur auf den 2-dimensionalen Raum. Ein Knoten mit einem Ausgangsgrad $d_G^-(v)$ von 2, hat einen Freiheitsgrad von 0 [14]. Wenn der Ausgangsgrad $d_G^-(v)$ 1 ist, ist ebenfalls der Freiheitsgrad 1 und wenn der Ausgangsgrad 0 ist, dann besitzt der Knoten 2 Freiheitsgrade. Ein Graph erfüllt alle vordefinierten Konsistenzbedingungen, wenn der Ausgangsgrad $d_G^-(v)$ von jedem Knoten nicht größer als 2 ist. Starrheit muss ebenfalls vorliegen, damit Persistenz erfüllt ist. Bei einem Graphen, der mindestens einen Knoten besitzt, dessen Ausgangsgrad $d_G^-(v) \geq 3$ ist, kann trotzdem Persistenz vorliegen. Die Persistenz bleibt erhalten, wenn man Kanten entfernt, sodass der Ausgangsgrad von keinem einzigen Knoten größer als 2 ist.

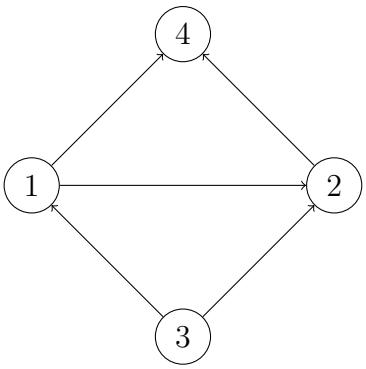
Ein Graph ist persistent, wenn die Summe der Freiheitsgrade der einzelnen Knoten nicht größer als 3 ist [14]. Ein Graph ist minimal persistent, wenn das Entfernen einer Kante zum Verlust der Persistenz-Eigenschaft führt.

Die angesprochene Methode, mit der sich feststellen lässt, ob ein Formationsgraph G persistent ist, wird in [14] erläutert. Wenn ein Graph Knoten besitzt, deren Ausgangsgrad größer als 2 ist, werden Kanten entfernt, sodass der Ausgangsgrad nicht mehr größer als 2 ist. Alle hierdurch konstruierbaren Subgraphen müssen starr sein, denn erst dann ist G persistent.

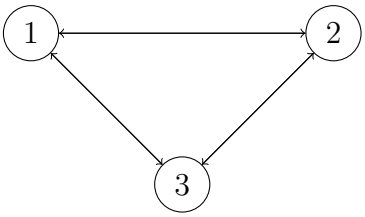
Für den Graphen G, dargestellt in Abbildung 2.13 a), lässt sich ein minimal persistenter Teilgraph G' finden, siehe 2.13 b). Um G' zu erhalten, wurde sowohl von Knoten 2, als auch von Knoten 1 je eine ausgehende Kante entfernt, da deren Ausgangsgrad $d_G^-(v) = 3$ gewesen ist. G ist dennoch nicht persistent, weil der Teil-



(a) G



(b) G'



(c) G''

Abbildung 2.13: Überprüfung eines Graphen G auf Persistenz

graph G'' in 2.13 c) nicht starr ist. G'' erhält man, indem von den Knoten 1 und 2 von G jeweils eine ausgehende Kante entfernt wird. Starrheit von jedem möglichen Untergraphen wurde, wie weiter oben beschrieben, gerade als Voraussetzung für Persistenz von G gefordert. Der Graph G'' ist nicht starr, da der Knoten 4 sich frei bewegen kann, ohne dass eine Abstandsbedingung berücksichtigt werden müsste.

Es gilt im Allgemeinen, dass ein Graph selbst nicht persistent sein muss, um einen minimal persistenten Untergraphen besitzen zu können. G' ist minimal persistent, jedoch wurde nachgewiesen, dass G nicht persistent ist, siehe Abbildung 2.13. Die Aussage steht im Gegensatz zu der Überprüfung von G auf Starrheit, da es hier sowohl eine notwendige als auch eine hinreichende Bedingung ist, dass es ausreicht, wenn G einen minimal starren Teilgraphen besitzt. In diesem müssen alle Knoten aus G wieder enthalten sein. Dies ist Lamans Theorem.

Ein in nicht polynomialer Zeit laufender Algorithmus prüft, wie im Beispiel beschrieben, die Persistenz eines Graphen.

Kapitel 3

Steuerung von sich bewegenden UAVs

In diesem Abschnitt werden Methoden vorgestellt, die für sich bewegende Formationen konzipiert sind. Der Formationsgraph ist nicht länger stationär, sondern dynamisch. Angesprochen wird die Zuordnung von eintrittswilligen Agenten in eine bestehende Formation und die Korrektur der Formation, nachdem Umwelteinflüsse diese gestört haben.

3.1 Positionszuweisung eines UAVs beim Beitreten einer Formation

Die Agenten entnehmen der internen Datenbank alle Informationen über zur Verfügung stehende Formationen. Die Knoten des in den Raum zu legenden Ziel-Formationsgraphen sind den UAVs bekannt. Entweder wurden die Koordinaten bereits vor dem Start den UAVs mitgeteilt oder sie erhalten sie über Funk von der Basis. Die Funkreichweite entspricht einem Vielfachen der W-LAN-Reichweite. Eine dritte Möglichkeit besteht darin, dass ein "Leader-UAV" den Formationskonstruktionsbereich festlegt. Die UAVs verfügen über eine Karte, auf der sie ihre momentane Position verzeichnen. Damit nicht jedes UAV ein eigenes Koordiantensystem, beziehungsweise eine eigene Basis des Vektorraumes nutzt, werden GPS-Daten verwendet. Jede mögliche Zuteilung eines Agenten j zu einer Zielkoordinate i verursacht unterschiedliche Kosten c_{ij}, da der zum angestrebten Formationspunkt zurückzulegende Weg unterschiedlich lang ist. Für jedes UAV

gibt es eine individuelle optimale Position innerhalb der Formation. Da für 2 UAVs die gleiche Zielposition optimal sein kann, wird sich eines der beiden UAVs mit der nächstbesten Zielposition begnügen müssen. Das gewichtete Zuordnungsproblem entspricht einer Permutation über $[n] = \{1, \ldots, n\}$ [2]. Die n Elemente umfassende Menge der Agenten wird bijektiv auf sich selbst abgebildet.

$$\pi : [n] \rightarrow [n] \tag{3.1}$$

Es handelt sich um eine Permutation, da theoretisch alle Kombinationen bei der Zuordnung der UAVs zu Zielknoten möglich sind. Bei bijektiven Abbildungen wird jedem Element aus der Definitionsmenge genau ein Element aus der Zielmenge zugeordnet. Das Formationsproblem

$$\|x_{\pi(i)} - x_{\pi(j)}\| = d_{ij} \text{ für alle } i, j = 1, ..., n, i \neq j \tag{3.2}$$

soll mittels bijektiver Abbildung gelöst werden. Die Gleichung fordert, dass die Abstandsbedingungen für jedes UAV i und j innerhalb einer Formation erfüllt werden. Die Ungarische Methode ist ein Algorithmus mit einer Komplexität von $\mathcal{O}(n^3)$. Der gewichtete bipartite Graph $K_{n,n} = (V, E, w)$, wobei $V = X \cup Y$ ist, setzt sich aus $2n$ Knoten zusammen. Der Graph ist bipartit, da die Knoten in zwei disjunkte Teilmengen, X und Y, unterteilt sind. Ein Agent ist der Menge X zuzuordnen, wenn dieser seine Position innerhalb der Formation finden will. Eine freie Stelle innerhalb der Formation befindet sich in der Menge Y. Untereinander dürfen Knoten, die zu der gleichen Menge gehören, nicht mittels Kante verbunden werden. Die Menge M enthält Elemente aus der Kantenmenge $E = X \times Y$. Es gilt daher

$$M \subseteq E \tag{3.3}$$

Jeder Knoten aus Y ist höchstens zu einer dieser Kanten inzident. Eine Zuordnung ist erreicht, die Paarung ist also komplett, wenn $\text{card}(M) = n$, siehe Abbildung 3.1. Jedes UAV aus der Menge X weiß, wohin es sich in Y bewegen soll. Die Aufgabe ist es nun, die beste Lösung des gewichteten Zuordnungsproblems zu finden.

$$\min_\pi \sum_{i=1}^{n} c(\pi(i), i) \tag{3.4}$$

Die aufsummierten Kosten sollen minimiert werden . Der Algorithmus bricht ab, sobald jedes UAV seinen Platz in der Formation gefunden hat und die Kosten minimiert sind. Weitere Details und der Algorithmus, beschrieben von Kuhn und Munkres, der direkt mit dem bipartiten Graphen arbeitet, ist in [2] beschrieben. Innerhalb dieser Ausarbeitung wird ein alternativer Algorithmus beschrieben, der nicht direkt mit dem gewichteten bipartiten Graphen arbeitet, sondern stattdessen mit Matrizen.

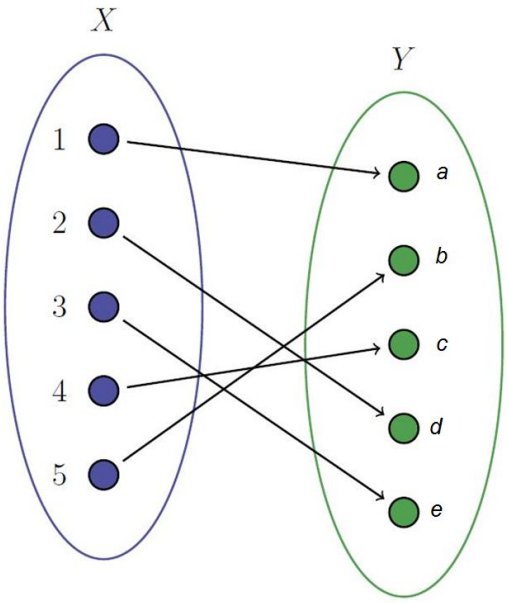


Abbildung 3.1: bipartiter Graph, der eine zulässige Lösung zeigt

3.1.1 Abstandsbestimmung zu allen möglichen Zielpunkten

Unter Verwendung einer gewichteten Adjazenzmatrix A des bipartiten Graphen wird das Zuordnungsproblem gelöst. $n - k$ Plätze der angestrebten Formation sind bereits besetzt. Daher stehen k freie Plätze zur Verfügung. Jedes UAV bestimmt den Abstand zu jeder freien Position innerhalb der Formation. Die k Abstände werden spaltenweise in eine Matrix B der Dimension $m + 1 \times k$ geschrieben. m ist die Dimension des Raumes. In das jeweilige Element $B_{m+1,i}$ wird der euklidische Abstand geschrieben. Der Index i steht für die Zielposition. Die korrekte Abstandsbestimmung wird erschwert, wenn sich ein Hindernis zwischen dem Agenten und dem Zielpunkt befindet.
Den gemessenen Abständen werden Kosten $c_{ij} \in \mathbb{N}$, zugeordnet. Hierzu sortiert jedes UAV j intern die Abstände und ordnet diese Präferenzwerten $P \in \mathbb{N} \setminus \{0\}$ zu. Ganzzahlige Präferenzwerte zwischen 1 und k werden vergeben. Ein niedriger Wert steht dabei für einen kleinen Abstand und k ist weiterhin die Anzahl freier Plätze. Bei der Präferenzwertberechnung werden auch die in die Matrix B aufgenommenen Richtungskomponenten des Abstands berücksichtigt. Wenn beispielsweise das UAV sich stark in z-Richtung bewegen soll, also massiv an Höhe gewinnen muss, so verschlechtert sich der temporäre Präferenzwert, da ein UAV horizontale Bewegungen schneller als vertikale ausführen kann.

Der Abstand zwischen Agent u und dem Zielpunkt 3 sei das 0,9 fache des Ab-

stands, der zwischen dem Agenten w und dem Zielpunkt 4 liegt. Beide UAVs würden den gleichen Präferenzwert 1 zuordnen, wenn kein kleinerer Abstand vorliegt. Um zu verhindern, dass 2 UAVs nach der Abstandsdiskretisierung den gleichen Abstand zu haben scheinen, wird die Präferenzwertberechnung modifiziert. Entfernungen werden in Intervalle unterteilt, die wiederum mit der Entfernung zunehmend größeren Präferenzwerten zugeordnet sind. Ausreißer, die viel zu weit entfernt sind und deren Beitreten in die Formation ineffizient sein würde, könnten von vornherein aussortiert werden, da deren Präferenzwerte alle oberhalb einer festgelegten Grenze ϵ sein würden. Stünde nur ein Präferenzwert zwischen 1 und k zur Verfügung, wären die Ausreißer nicht aufgefallen. Aus Kosten der Höhe 5 von maximal 5 ließe sich schließen, dass dieser Abstand relativ groß ist, doch dahinter könnte ein Ausreißer stecken, der z.B. eine 20-mal so große Distanz zurücklegen müsste, wie im Durchschnitt die anderen UAVs. Bevor der Algorithmus, der als Übergabeparameter die Kosten, sowie X und Y benötigt, gestartet wird, muss nun eine neue, dem Anwendungszweck der gerade verworfenen Formation entsprechende Formation mit $n-1$ Knoten ausgewählt werden. Alle Agenten bis auf den Ausreißer bestimmen erneut ihre Abstände zu den Zielkoordinaten.

Bei einer Verkleinerung der Intervalllänge wird die Lösung genauer. Konvergenz gegen Null ist nicht sinnvoll, da approximativ gleiche Abstände in der Anwendung eher selten sind.

Die tatsächlichen Abstände können nicht vom Wert her direkt für die Kosten genutzt werden, da der Algorithmus aufgrund der Kommazahlen ansonsten viel zu lange benötigen würde. Ein Abstand $d_{ij} = 1{,}42$ LE kann also nicht Kosten von $c_{ij} = 1{,}42$ entsprechen. LE steht für Längeneinheiten. Als Vereinfachung ist es vorstellbar, dass die Nachkommastellen einfach abgeschnitten werden. Der Algorithmus würde in diesem Fall jedoch eine längere Bearbeitungszeit in Anspruch nehmen

Das Beispiel verdeutlicht, wie das UAV j den Abständen zu den freien Positionen i Präferenzwerte zuordent. ϵ betrage 80 LE und die Intervalllänge sei 7 LE.

Agent j	Zielkoordinate i	d_{ij} in LE	$d_{ij} : 7$	Präferenzwert P
1	a	24.10	3.44	4
2	b	70.42	10.06	11
3	c	61.23	8.75	9
4	d	30.31	4.33	5
5	e	6.23	0.89	1

Da es das 0.te Intervall nicht gibt, wird nach dem Entfernen der Nachkommastellen, noch die 1 hinzuaddiert.

Die Matrix A, in welche die Präferenzwerte eingetragen werden, ist genau dann quadratisch, wenn die Anzahl der freien Plätze der Anzahl an Bewerbern entspricht. Dies ist nicht der Fall, wenn l Agenten, $l > k$, um die k freien Plätze konkurrieren. $l - k$ Zeilen bzw. Spalten werden mit Nullen aufgefüllt. Außerdem können sich, wie bereits erwähnt, nur Agenten anschließen, die bereits in der Nähe sind. Im Rahmen dieser Arbeit wird vom idealen Fall ausgegangen, bei dem $l = k$ ist, also die Anzahl an Bewerbern und die Anzahl an freien Plätzen übereinstimmt. Angenommen die UAVs können eigenständig bestimmen, wo die Formation zu Stande kommen soll. Dann muss auch dieses Problem gelöst werden. Aufgrund der resultierenden unterschiedlichen Abstände ist dieses ebenfalls gewichtet. Für diese Bestimmung reicht die Ungarische Methode nicht mehr aus, da nun zuallererst die in der Teilmenge Y zu speichernden Koordinaten bestimmt werden müssen.

3.1.2 Verwendung der ungarischen Methode

Der Algorithmus, der ohne Berücksichtigung eines Anwendungsbereiches in [10] dargestellt ist, wird mittels eines Beispiels [10], erläutert. Das minimale Überdeckungssystem wird bestimmt und danach wird eine Kostentransformation durchgeführt. 5 Agenten konkurrieren um genausoviele zur Verfügung stehende freie Plätze einer zu bildenden Formation. Das Element a_{ij} enthält den Präferenzwert für den zurückzulegenden Abstand zwischen dem Agenten j und der freien Position i. Der zurückzulegende Abstand zwischen UAV 4 und Zielpunkt 3 würde Kosten von 5 verursachen, siehe (3.5). Der zurückzulegende Abstand zwischen UAV 2 und Zielpunkt 4 würde Kosten von 7 verursachen, siehe ebenfalls (3.5).

Zeilen und Spalten-Bezeichnungen sind vertauschbar. Es handelt sich dann um die Transponierte der Matrix.

Nach dem Eintragen der Kosten in die 5×5 Matrix, besitzt der Algorithmus alle zum Starten notwendigen Informationen

$$\begin{array}{ccccc} 3 & 4 & 5 & 4 & 6 \\ 4 & 4 & 2 & 3 & 6 \\ 5 & 1 & 6 & 5 & 1 \\ 6 & 7 & 7 & 10 & 8 \\ 4 & 5 & 3 & 5 & 6 \end{array} \tag{3.5}$$

Zeilenminima bestimmen und in $5 + 1$ Spalte eintragen

$$\begin{matrix} 3 & 4 & 5 & 4 & 6 & 3 \\ 4 & 4 & 2 & 3 & 6 & 2 \\ 5 & 1 & 6 & 5 & 1 & 1 \\ 6 & 7 & 7 & 10 & 8 & 6 \\ 4 & 5 & 3 & 5 & 6 & 3 \end{matrix} \tag{3.6}$$

Subtraktion des i-ten Zeilenminima von den Elementen der i-ten Zeile
Spaltenminima bestimmen und in $5+1$ Zeile eintragen

$$\begin{matrix} 0 & 1 & 2 & 1 & 3 \\ 2 & 2 & 0 & 1 & 4 \\ 4 & 0 & 5 & 4 & 0 \\ 0 & 1 & 1 & 4 & 2 \\ 1 & 2 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 & 0 \end{matrix} \tag{3.7}$$

Subtraktion des j-ten Spaltenminima von den Elementen der j-ten Spalte.
Innerhalb einer jeden Zeile und Spalte kommt folglich mindestens einmal eine 0 vor

$$\begin{matrix} 0 & 1 & 2 & 0 & 3 \\ 2 & 2 & 0 & 0 & 4 \\ 4 & 0 & 5 & 3 & 0 \\ 0 & 1 & 1 & 3 & 2 \\ 1 & 2 & 0 & 1 & 3 \end{matrix} \tag{3.8}$$

Das Ziel ist die Bestimmung der "maximalen Anzahl unabhängiger 0-er". In jeder Zeile und jeder Spalte kommt dann höchstens eine 0 vor.
Eine Matrixkomponente, die 0 ist, wird markiert, wenn das folgende Kriterium erfüllt ist. Es gibt keine andere Zeile oder Spalte, die weniger 0-en enthält. Alle weiteren 0-en, welche die Zeile und die Spalte, die die markierte 0 schneiden, enthalten, werden gelöscht. Dieses Vorgehen wird solange fortgesetzt, bis jede 0 der Matrix entweder gelöscht oder markiert ist.

$$\begin{array}{ccccc} 0 & 1 & 2 & 0 & 3 \\ 2 & 2 & & 0 & 4 \\ 4 & 0 & 5 & 3 & 0 \\ 0 & 1 & 1 & 3 & 2 \\ 1 & 2 & 0 & 1 & 3 \end{array} \tag{3.9}$$

$$\begin{array}{ccccc} 0 & 1 & 2 & & 3 \\ 2 & 2 & & 0 & 4 \\ 4 & 0 & 5 & 3 & 0 \\ 0 & 1 & 1 & 3 & 2 \\ 1 & 2 & 0 & 1 & 3 \end{array} \tag{3.10}$$

$$\begin{array}{ccccc} & 1 & 2 & & 3 \\ 2 & 2 & & 0 & 4 \\ 4 & & 5 & 3 & 0 \\ 0 & 1 & 1 & 3 & 2 \\ 1 & 2 & 0 & 1 & 3 \end{array} \tag{3.11}$$

Der Algorithmus terminiert, wenn die maximale Anzahl unabhängiger 0-er gleich der Zeilen bzw. Spaltenanzahl ist, also der Dimension der quadratischen Matrix entspricht. In der Matrix wurde bislang 4 Mal die 0 markiert. Die Dimension der quadratischen Matrix ist jedoch 5. Da eine markierte 0 zu wenig ist, muss das minimale Überdeckungssystem bestimmt werden.

3.1.2.1 Minimales Überdeckungssystem

Die zu bestimmenden Überdeckungslinien sollen alle Komponenten, die 0 sind, überdecken. Die minimale Anzahl an Überdeckungslinien entspricht der maximalen Anzahl unabhängiger 0-er. In diesem Beispiel müssen also 4 Überdeckungslinien gefunden werden.

Wenn es keine markierte 0 in einer Zeile gibt, wird in die $5 + 1$ Spalte ein $\times$

für diese Zeile geschrieben.

$$\begin{array}{ccccccc} & 1 & 2 & & 3 & \times \\ 2 & 2 & & 0 & 4 & \\ 4 & & 5 & 3 & 0 & \\ 0 & 1 & 1 & 3 & 2 & \\ 1 & 2 & 0 & 1 & 3 & \end{array} \tag{3.12}$$

Wenn eine Zeile bereits markiert ist und eine gelöschte 0 enthält, wird die Spalte markiert, die als gemeinsames Element mit der Zeile die gelöschte 0 hat

$$\begin{array}{ccccccc} \times & & & \times & & \\ & 1 & 2 & & 3 & \times \\ 2 & 2 & & 0 & 4 & \\ 4 & & 5 & 3 & 0 & \\ 0 & 1 & 1 & 3 & 2 & \\ 1 & 2 & 0 & 1 & 3 & \end{array} \tag{3.13}$$

Jede Zeile wird markiert, die eine markierte 0 besitzt, welche auch in einer markierten Spalte steht

$$\begin{array}{ccccccc} \times & & & \times & & \\ & 1 & 2 & & 3 & \times \\ 2 & 2 & & 0 & 4 & \times \\ 4 & & 5 & 3 & 0 & \\ 0 & 1 & 1 & 3 & 2 & \times \\ 1 & 2 & 0 & 1 & 3 & \end{array} \tag{3.14}$$

Die letzten beiden Schritte werden solange wiederholt, bis weder Zeilen noch Spalten markiert werden können.

$$\begin{array}{ccccccc} \times & & \times & \times & & \\ & 1 & 2 & & 3 & \times \\ 2 & 2 & & 0 & 4 & \times \\ 4 & & 5 & 3 & 0 & \\ 0 & 1 & 1 & 3 & 2 & \times \\ 1 & 2 & 0 & 1 & 3 & \times \end{array} \tag{3.15}$$

Es werden die Elemente gestrichen, die zu einer *nichtmarkierten* Zeile gehören. Ebenso werden die Komponenten einer *markierten* Spalte gestrichen.

$$\begin{array}{cccccc} \times & & \times & \times & & \\ & 1 & \not{2} & & 3 & \times \\ \not{2} & 2 & & \not{0} & 4 & \times \\ \not{4} & & \not{5} & \not{3} & \not{0} & \\ \not{0} & 1 & \not{1} & \not{3} & 2 & \times \\ \not{1} & 2 & \not{0} & \not{1} & 3 & \times \end{array} \tag{3.16}$$

Die Forderung, dass die minimale Anzahl an Überdeckungslinien gleich der maximalen Anzahl unabhängiger 0-er sein soll, ist erfüllt. Es gibt 4 Matrixelemente, die eine rot markierte 0 enthalten und ebensoviele Überdeckungslinien sind eingezeichnet, welche die 0-en überdecken.

3.1.2.2 Kostentransformation

Aus allen nicht herausgestrichenen Elementen wird das Minimum gesucht. Alle doppelt überdeckten Komponenten sind hier grün gefärbt.

$$\begin{array}{cccccc} \times & & \times & \times & & \\ & 1 & \not{2} & & 3 & \times \\ \not{2} & 2 & & \not{0} & 4 & \times \\ \not{4} & & \not{5} & \not{3} & \not{0} & \\ \not{0} & 1 & \not{1} & \not{3} & 2 & \times \\ \not{1} & 2 & \not{0} & \not{1} & 3 & \times \end{array} \tag{3.17}$$

Das Minimum, 1, wird zu allen doppelt überdeckten Elementen hinzuaddiert und von allen überhaupt nicht überdeckten subtrahiert.

$$
\begin{array}{cccccc}
\times & & \times & \times & & \\
 & 0 & \not{2} & & 2 & \times \\
\not{2} & 1 & & \not{0} & 3 & \times \\
\not{5} & & \not{6} & \not{4} & \not{0} & \\
\not{0} & 0 & \not{1} & \not{3} & 1 & \times \\
\not{1} & 1 & \not{0} & \not{1} & 2 & \times
\end{array}
\tag{3.18}
$$

3.1.2.3 neue Iteration

Die maximale Anzahl unabhängiger 0-er wird erneut bestimmt, siehe Verfahren auf Seite 36.

$$
\begin{array}{ccccc}
0 & 0 & 2 & 0 & 2 \\
2 & 1 & 0 & 0 & 3 \\
5 & 0 & 6 & 4 & 0 \\
0 & 0 & 1 & 3 & 1 \\
1 & 1 & 0 & 1 & 2
\end{array}
\tag{3.19}
$$

$$
\begin{array}{ccccc}
0 & 0 & 2 & 0 & 2 \\
2 & 1 & & 0 & 3 \\
5 & 0 & 6 & 4 & 0 \\
0 & 0 & 1 & 3 & 1 \\
1 & 1 & 0 & 1 & 2
\end{array}
\tag{3.20}
$$

Die Anzahl unabhängiger 0-er stimmt nun mit der Anzahl an Zeilen bzw Spalten der quadratischen Matrix überein. Beide Werte sind hier 5. Die Abbruchbedingung für den Algorithmus ist somit erfüllt, vergl. auf Seite 37 mit der Matrix (3.11), bei der noch eine markierte 0 zu wenig vorlag.

$$
\begin{array}{ccccc}
 & 0 & 2 & & 2 \\
2 & 1 & & 0 & 3 \\
5 & & 6 & 4 & 0 \\
0 & & 1 & 3 & 1 \\
1 & 1 & 0 & 1 & 2
\end{array}
\tag{3.21}
$$

3.1.2.4 Auswertung der Matrix

Agent X	Kosten	Zielkoordiante Y
1	6	4
2	4	1
3	3	5
4	3	2
5	1	3

Agent 1 bewegt sich in Richtung Formationspunkt 4, da das Element a_{41} der Matrix (3.21) gleich 0 ist. Agent 3 bewegt sich in Richtung Formationspunkt 5, da das Element a_{53} der Matrix (3.21) gleich 0 ist.

Man erkennt beispielhaft an den Agenten 1 und 2, dass sich einige Agenten nun zu Zielkoordinaten hin bewegen müssen, denen nicht der höchste interne Präferenzwert zugewiesen wurde. Dies wird bei dem Vergleich der Tabelleneinträge mit der Interpretation der Matrix (3.5) deutlich. Dennoch gibt es keine bessere Lösung.

Der Verlauf des in [10] verwendeten Algorithmus ist nicht eindeutig, da beim Bestimmen der maximalen Anzahl unabhängiger 0-er, siehe Matrix (3.9), mehrere Elemente die Forderungen gleichzeitig erfüllen. Abhängig von der Wahl sieht die Matrix bei jedem kommenden Schritt anders aus. Die Lösung ist unabhängig von der Wahl, ebenso der Rechenaufwand, also die Anzahl an Matrixmanipulationen.

3.1.3 Realisierbarkeit der Ungarischen Methode

Der Algorithmus muss entweder von jedem UAV ausgeführt werden, oder zentral vom "Leader-UAV". Die UAVs kommen alle zu der gleichen Lösung und entnehmen die relevanten Informationen. Die Alternative ist, dass das "Leader-UAV" nach erfolgreichem Ausführen des Algorithmus jedem UAV eine Nachricht sendet, welche die letzte Matrix (3.21) oder nur die Nummer der Zielposition enthält. Die Reichweite des W-LANs ist, wie bereits erwähnt, begrenzt. Wenn die UAVs zuerst das Rendezvous-Problem lösen sollten, würde sich die Frage ergeben, warum noch die Ausführung der Ungarischen Methode Sinn ergibt, wenn sich eh alle UAVs einem Rendezvous-Punkt nähern. Man könnte fordern, dass die W-LAN-Reichweite unbegrenzt ist, jedoch würden dann alle Überlegungen und Konzepte nicht immer nutzbar sein. Daher muss ein leistungsstärkerer Nachrichtenkanal als der des W-LANs genutzt werden. Hierfür müsste ein neuer Funkkanal den UAVs zur Verfügung gestellt werden. Der Nachteil von leistungsfähigeren Funkantennen ist nicht nur das zusätzliche Gewicht, sondern der deutlich höhere Energiebedarf

im Vergleich zum W-LAN. Es wird weiterhin W-LAN genutzt, wenn über den Kommunikationsgraphen Nachrichten versandt werden sollen.

3.2 Henneberg Folge: Erweiterung eines minimal starren Graphen

Wenn ein UAV die Absicht hat, sich einer bereits bestehenden Gruppe anzuschließen, muss die Formation um einen Agenten erweitert werden. Dieser Ausbau darf nicht zu dem Verlust der minimalen Starrheit führen. Die folgenden beiden Methoden verhindern diesen Eigenschaftsverlust, sind jedoch unterschiedlich effizient und abhängig von der Anwendung.

Die Henneberg Folge, erläutert in [6], erzeugt eine Sequenz von Graphen $G_2, G_3, \ldots, G_{|V|}$. Im Index steht die Anzahl der beteiligten Knoten. Die Anwendungsbedingung ist, dass G_2 sowohl *komplett*, als auch starr ist und G_{i+1} wird erreicht mittels Anwendung entweder der Henneberg-Operation "Knotenaddition" oder der "Kantenteilungs"-Operation auf G_i.

Die in [5] erläuterte Henneberg Folge ist nur für 2-dimensionale Graphen definiert. Eine Henneberg Folge besteht aus dem Hintereinanderausführen von "Knotenadditionen" und "Kantenteilungen". Eingeführt wurden die Henneberg-Operationen von dem deutschen Mathematiker Lebrecht Henneberg.

Angaben für spezielle Formationserweiterungen, also Erweiterungen die auf die Verwendung einer Henneberg-Operation verzichten, müssen in der Datenbank der beitretenden UAVs hinterlegt sein. Da die Formationen in der Regel nicht allzu groß sind, würde der hierfür benötigte Speicherplatz nicht ins Gewicht fallen. Eine Erweiterung, ausgehend von einer mathematischen Vorgehensweise, wird als generisch bezeichnet. Der Vorteil gegenüber einer generischen Erweiterung mittels Henneberg-Operationen besteht darin, dass die Formationserweiterung unter Berücksichtigung von Umwelteinflüssen, Zielen und der Formationshierarchie möglich ist. Die Henneberg Folge könnte weniger gut berücksichtigen, dass beispielsweise ein neues "Leader"-UAV oder ein "First-follower" der Formation beitritt. Ein geeigneter Bereich in der bisherigen Formation müsste für die Formationserweiterung geschickt ausgewählt werden. Angenommen die Formation besteht aus 5 Mitgliedern und ein "Leader-UAV" will der Formation beitreten. Die Ausführung einer einfachen Henneberg-Operation wird dem "Leader-UAV" nicht gerecht. Es ist sinnvoll, die Formation aufzulösen und einen "wheel graph", siehe Abbildung 4.10 im Abschnitt "In der Anwendung oft verwendete Formationen" auf Seite 77, zu konstruieren. Alle UAVs, ausgenommen das "Leader-UAV",

führen die Ungarische Methode aus und bewegen sich zu ihren Zielpunkten. Das "Leader-UAV" kennt seine Zielposition, die direkt im Zentrum des Formationsgraphen liegt.

3.2.1 Voraussetzungen für das Anwenden der Henneberg-Operationen

Knotenadditionen können immer ausgeführt werden, wenn der Graph aus 2 Knoten besteht, die mittels Kante verbunden sind. Die beiden Agenten müssen einen geforderten Abstand einhalten und die Voraussetzung für eine beliebig oft erweiterbare Formation ist geschaffen. Kantenteilungen sind hingegen erst möglich, wenn der Graph beispielsweise aus einem Subgraphen besteht, der eine Dreiecksformation enthält.

3.2.2 Knotenaddition

Der in die Formation einzugliedernde Knoten wird über je eine Kante mit 2 zueinander adjazenten Knoten verbunden.

$$G' := G|_{(v_j,v_k)} \oplus \{v_l\} \tag{3.22}$$

Auf den Formationsgraphen $G = (V, E)$ wird die Knotenaddition angewendet. Das Beispiel in Abbildung 3.2 dient der Veranschaulichung. In (3.22) wird der Knoten v_l hinzugenommen. Es werden Kanten zu v_j und v_k erstellt [7]. Das Zeichen $\oplus$ steht für einen Henneberg-Operator.

$V' = V \bigcup \{l\}$

$E' = E \bigcup \{e_{lj}, e_{lk}\}$ Für die Knoten gilt $j, k \in V$

Eine Folge τ von Knotenadditionen kann wie folgt dargestellt werden

$\tau = (\{v_1\} \oplus \{v_2\}) \oplus \{v_3\}$

Die in runden Klammern stehende Operation wird zuerst ausgeführt [7]. Eine Dreiecksformation wird konstruiert, indem nacheinander 3 Agenten sich zu einer Formation zusammenschließen. Diese Schreibweise impliziert im Allgemeinen nicht die Anwendung der Henneberg-Operationen. Der Grund ist, dass die

(a) $G = (V, E)$ (b) $G' = (V', E')$

Abbildung 3.2: Knotenaddition

Voraussetzung für die Anwendung einer Knotenaddition, wie beschrieben, das Vorhandensein einer Kante ist. Diese Bedingung kann bei einer initial zu konstruierenden Formation natürlich nicht erfüllt sein.

3.2.3 Kantenteilung

Eine Kante wird bei der Kantenteilung herausgenommen und im Gegenzug werden 2 Kanten eingefügt, die inzident zu einem neuen Knoten sind. Jeweils ein Eckpunkt von den beiden Kanten ist jeweils einer der beiden Knoten, zwischen denen zuvor die Verbindung aufgehoben wurde. Eine weitere Kante wird eingefügt, die nun den der Formation beigetretenen Knoten mit einem bereits zuvor bestehenden verbindet.

Auf den Formationsgraphen $G = (V, E)$ wird die Kantenteilung angewendet, siehe hierfür auch das Beispiel in Abbildung 3.3.

$V' = V \bigcup \{v\}$

$E' = E \bigcup \{e_{vj}, e_{vk}, e_{vm}\} \setminus \{e\}$

Für die Knoten gilt $j, k, m \in V$ und mindestens zwei von diesen sind adjazent zueinander in G. Die Kante e ist entweder e_{jk}, e_{jm} oder e_{km}.

(a) $G = (V, E)$ (b) $G' = (V', E')$

Abbildung 3.3: Kantenteilung

3.2.4 Auswahl der Henneberg-Operationen

Der nach Anwendung einer Henneberg Operation entstandene Graph $G' = (V', E')$ ist weiterhin minimal starr, jedoch nicht unbedingt planar. Da die entstehende Gesamtanzahl an Kanten und Knoten in G' unabhängig von der gewählten Operation ist, sind beide Operationen gleichwertig. Beide Operationen erzeugen gleich viele Abstandsbedingungen. Dennoch sollte nicht nur ausschließlich eine Operation verwendet werden, da andernfalls die Formation allzu generisch aufgebaut wird. Es ist zu klären, für welche Operation sich das beitretende UAV entscheidet. Möglich ist, dass mittels einer 2 Variablen umfassenden Zufallsmenge die Operation bestimmt wird. Es gibt 2^k Möglichkeiten, wie die Formation nach Hinzunahme von k Agenten aussehen kann. Nachdem eine Operation von dem beitrittswilligen UAV oder dem "Leader-UAV" ausgewählt wurde, werden die an der Operation zu beteiligenden Agenten informiert. Diese können nun Abstandsbedingungen aufheben und neue zu dem ankommenden UAV konstruieren. Das beitrittswillige UAV kommuniziert mit bereits in der Formation vorhandenen UAVs über Funk. Das UAV wechselt erst zum W-LAN-Kommuniaktionskanal, wenn der Formationsgraph eingehalten wird. Der Kommunikationsgraph wird ebenfalls angepasst.

3.2.5 Umgekehrte Verwendung der Henneberg-Operationen

Beim Rückwärtsanwenden der Operationen wird genau das Gegenteilige vom Beschriebenen durchgeführt. So werden bei der umgekehrten Kantenteilung ein Knoten, sowie 3 Kanten herausgenommen und hierfür wird eine neue Kante eingefügt. Dieses entgegengesetzte Vorgehen findet Anwendung, wenn der Verlust oder das Ausscheiden eines Agenten zu kompensieren ist. Das Verfahren kann

$n-2$ mal ausgeführt werden, bis nur noch ein mittels Kante verbundenes Knotenpaar übrigbleibt.

3.2.6 Konstruktion eines persistenten Graphen

In [4] wird darauf eingegangen, wie minimal persistente Graphen konstruiert werden können. Während der Konstruktion eines minimal starren Graph unter Verwendung der Henneberg-Operationen wird bei jeder Hinzunahme einer Kante, dieser auch eine Richtung zugewiesen. Die hierbei zu beachtende Regel ist, dass der Ausgangsgrad $d_G^-(v)$ eines Knotens maximal 2 sein darf. Dies gilt für den R^2, wohingegen im R^3 der Ausgangsgrad $d_G^-(v)$ nicht größer als 3 werden darf.

3.3 X-replacement

Eine weitere auf Graphen anwendbare Operation, um einen Agenten in eine bestehende Formation aufzunehmen, ist die "X-replacement"-Operation [13]. Es handelt sich hierbei um keine klassische Henneberg-Operation. Bedingung für die Anwendung auf einen nichtplanaren Formationgraphen sind zwei sich schneidende Kanten. In dem Schnittpunkt wird ein neuer Knoten v eingefügt.

Der Graph G besitzt einen Teilgraphen F, mit der Eigenschaft, dass $e_{12}, e_{34} \in E$ und der einzufügende Knoten v_5 ist $\notin V$, siehe Abbildung 3.4. Nach Anwendung der "X-replacement"-Operation besitzt G' die Knoten $V' = V \cup \{v_5\}$ und die Kanten $E' = (E \setminus \{e_{12}, e_{34}\}) \cup \{e_{5i} | i \in \{1,2,3,4\}\}$. An der Starrheit ändert sich mit der Ausführung der Operation nichts. Mit v_5 kann bei gerichteten Graphen einer Formation ein "Leader"-UAV beitreten, wenn dessen Ausgangsgrad 0 ist.

(a) G (b) G'

Abbildung 3.4: Ausführung der X-Replacement-Operation

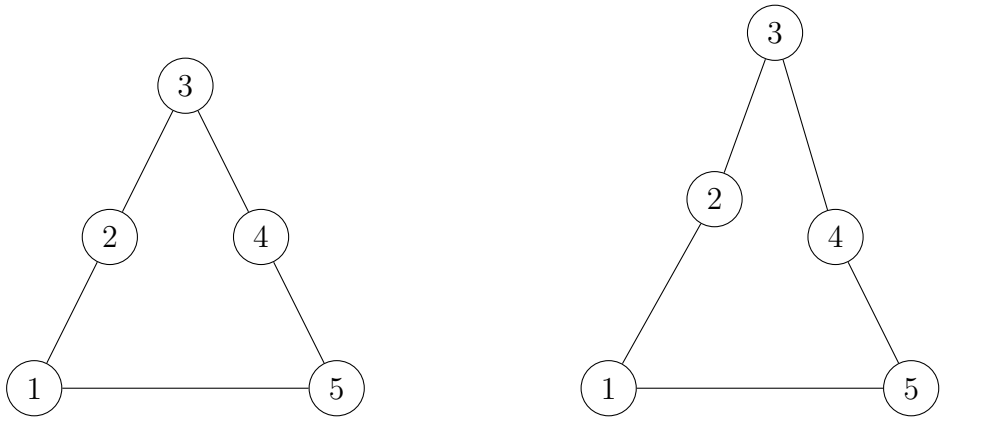
3.4 Einhaltung der Abstandsbedingungen einer sich bewegenden Formation

In diesem Abschnitt geht es darum, dass die Agenten in einer Formation die Abstände untereinander einhalten. Zunächst wird davon ausgegangen, dass der minimal starre Graph stationär ist. Erst später erhält die Formation dynamische Eigenschaften, die der Bewegung gerecht werden.

Die neuen Agenten haben sich zu den mittels Ungarischer Methode bestimmten Zielpositionen bewegt und sind nun Mitglieder der Formation. Die Abstandsvorgaben $\|x_i - x_j\| = d_{\text{ij}}$ zwischen den Agenten innerhalb der geforderten Formation sind dennoch noch nicht erfüllt. Nachdem die ersten UAVs bereits ihre Position erreicht haben, warten sie solange, bis auch die letzten UAVs ihre Position erreicht haben. Da die wartenden UAVs währenddessen den Umwelteinflüssen wie Wind und Regen ausgesetzt sind, ändert sich deren Position leicht. Auch die ankommenden UAVs werden beim Anflug zur gewünschten Zielposition abgelenkt und erreichen daher trotz Korrekturen nicht die exakten Zielkoordinaten. Im Simulator werden diese Einflüsse unter Hinzunahme von Gaußverteilten Zufallsvariablen berücksichtigt.

Die "Cyclic stop-and-go strategy" ist eine Methode, um mehrere Abstandskorrekturen innerhalb der Formation zeitlich zu koordinieren. Alle Agenten sind dahingehend zu steuern, dass sie ihre geforderten Abstandsbedingungen $\|x_i - x_j\| = d_{\text{ij}}$ erfüllen. Damit nicht alle UAVs gleichzeitig ihre Abstände zueinander korrigieren wollen und sich bewegen, sind Regeln aufzustellen, welche den UAVs klar definierte Zeiträume zum Agieren zuweisen. Um Effizienz zu gewährleisten, sind die UAVs Teilmengen so zuzuordnen, dass eine maximalmögliche Anzahl an UAVs gleichzeitig agieren darf. Die Idee des in [4] vorgestellten Algoritmus beruht darauf, dass stationäre Agenten ausschließlich den Abstand untereinander bestimmen können. Kommunikationswege zwischen benachbarten UAVs stehen daher nicht zur Verfügung und relative Positionsangaben sind nicht möglich. Die Forderungen werden im Kontext der eigenen Arbeit aufgehoben. Die "Cyclic stop-and-go strategy" kann auch auf eine sich bereits bewegende Formation angewendet werden, da die Zeit, die für die Erfüllung der Auftragsziele zur Verfügung steht, nicht mit Aktionen, wie der Formationseinhaltung, verschwendet werden soll. Eine zeitliche Überlagerung der beiden Vorgänge, Translation und Positionskorrektur, lässt sich berücksichtigen, indem das System sich mit der konstanten Geschwindigkeit v bewegt [4].

Die Lyapunov Funktion ist für einen Agenten, der den Abstand zu mehreren Nachbarn korrigieren will, ein Anhaltspunkt dafür, welche Auswirkungen eine Positionsänderug hat. Der Lyapunov Funktionswert wird neu berechnet und mit



(a) geforderte Abstände eingehalten (b) Abstandsbedingungen nicht erfüllt

Abbildung 3.5: Störung der Formation

dem alten Wert verglichen. Eine Lyapunov Funktion, welche die Standorte aller Agenten berücksichtigt, ist in [4] wie folgt dargestellt.

$$W(p_1(t), p_2(t), ..., p_n(t)) = \sum_{e_{ij} \in E} \left(\delta_{ij}^2(t) - d_{ij}^2\right)^2 \tag{3.23}$$

Die Agenten versuchen, die Potentialfunktion (3.23) gegen 0 laufen zu lassen. $p_i \in R^2$ ist der Ortsvektor des Agenten i. $\delta_{ij}(t)$ ist der zum Zeitpunkt t von Agent i bestimmte Abstand zu Agent j und d_{ij} ist der zeitinvariante, geforderte Abstand.

Die Lyapunov Funktion, welche nur die Standorte aller benachbarten Agenten des Agenten A berücksichtigt, ist in [4] wie folgt dargestellt.

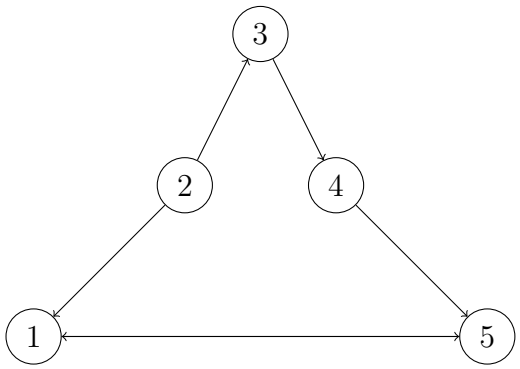
$$W_A(p_1, p_2, ..., p_n) = \sum_{e_{Aj} \in E} \left(\delta_{Aj}^2 - d_{Aj}^2\right)^2 \tag{3.24}$$

Die Gleichung ist nicht mehr abhängig von der Zeit t, da die von einem UAV bestimmte Potentialfunktion unabhängig von der bisherigen Laufzeit des Algorithmus ist.

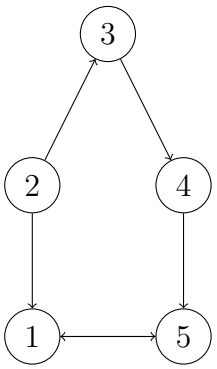
Das folgende Beispiel verdeutlicht die Notwendigkeit des Verwendens der cyclic stop-and-go strategy, siehe hierzu auch Abbildung 3.5. Der Dreiecks-Formation liegt ein Graph zu Grunde, dessen Kontrollstruktur symmetrisch ist. Die Formation, die in 3.5 a) wie gefordert vorliegt, wird gestört, sodass die Struktur, die in 3.5 b) dargestellt ist, vorliegt. Agent 1 erkennt nach Auswertung der Abstands-

messung zu Agent 2, dass der Abstand größer ist, als gefordert. Der Abstand zwischen Agent 1 und Agent 5 ist hingegen korrekt. Um den aufsummierten Abstandsfehler ${\epsilon_{15}}^2 + {\epsilon_{12}}^2$ so weit wie möglich zu minimieren, bewegt sich Agent 1 in Richtung Agent 2 und lässt nun einen entstehenden Fehler zwischen ihm und Agent 5 zu. Der Wert der Lyapunov Funktion nimmt ab. Es ist wahrscheinlich, dass Agent 2 den Fehler ϵ_{23} korrigieren möchte. Der Versuch des Agenten 1, einem Erfüllen der Abstandsbedingung d_{12} näher zu kommen, ist somit fürs erste fehlgeschlagen und muss unter Berücksichtigung der neuen Position von Agent 2 wiederholt werden.

Unter Verwendung einer vollständigen asymmetrischen Kontrollstruktur könnte zumindest der Fall, dargestellt in Abbildung 3.6, nicht eintreten. Das Aussehen des geforderten Formationsgraphen ist in Abbildung 3.5 a) zu sehen. Die Formation wird gestört, siehe Abbildung 3.6 a). Die dort dargestellte Kontrollstruktur, die sowohl aus einer symmetrischen, als auch aus einer asymmetrischen Kontrollstruktur besteht, soll nun genutzt werden. Die Abstandskorrektur der Agenten 1 und 5 führt zu einer Verschlechterung des Lyapunov Funktionswertes (3.23), der die Standorte aller UAVs berücksichtigt. Wenn die Agenten 5 und 1 sich gleichzeitig entgegen kommen, kann der entstehende Fehler betragsmäßig höchstens dem vorangegangenen Fehler entsprechen. Jedoch werden nun die Abstandsforderungen d_{21} und d_{45} nicht mehr erfüllt, siehe 3.6 b).



(a) Abstand zwischen Agent 1 und 5: 1,5 d_{15}



(b) Abstand zwischen Agent 1 und 5: 0,5 d_{15}

Abbildung 3.6: nichtkoordinierte Abstandskorrektur

3.4.1 Idee und Ablauf der cyclic stop-and-go strategy

Die an der Formation beteiligten Agenten werden nach Regeln, die weiter unten beschrieben werden, in Untermengen aufgeteilt. Diese Untermengen werden in

einen Bewegungs-Ablaufplan eingetragen, welcher eine zyklische Abfolge enthält.

$$\{\vartheta_1, \vartheta_2, \vartheta_3, ...\vartheta_m,\ \vartheta_1, ...\} \tag{3.25}$$

Die UAVs der gerade ausgewählten Untermenge ϑ_i erhalten die Möglichkeit, den Abstand zu ihren Nachbarn zu ändern. Der Agent bestimmt alle Abstände und bestimmt die neuen Koordinaten, damit die Summe der quadrierten Abstandsfehler minimiet wird, also der Wert der Lyapunov-Funktion (3.24) verkleinert wird. Der Bewegungs-Ablaufplan ist zyklisch, denn nachdem $v_j \in \vartheta_1$ den Abstandsfehler zu $v_k \in \vartheta_2$ verbessert hat, während $v_i \in \vartheta_3$ gewartet hat, ist nun der der zuvor ansatzweise korrigierte Abstand zwischen v_i und v_j wieder schlechter geworden. Das Beispiel in Abbildung 3.5 verdeutlicht dies.

Das Ziel besteht darin, so wenig Teilmengen wie möglich zu verwenden, damit ein Zyklus sehr kurz ausfällt. Der Bewegungsablaufplan ist nur abhängig von den Knoten V und den Kanten E und ist unabhängig von der zu Grunde liegenden Kontrollstruktur. Dies liegt daran, dass Nachbarn stationär sein sollen und der Nachbar ähnlich wie bei einer zu Grunde liegenden asymmetrischen Steurerung nichts von der Korrektur mitbekommt. Für die Einteilung der Agenten in m Teilmengen sind drei Regeln zu berücksichtigen:

1. In jeder Untermenge $\{\vartheta_i\}$ sind nur nichtzueinander adjazente Knoten enthalten
2. $\vartheta = \bigcup_{i=1}^{m} \vartheta_i$ Alle Agenten werden berücksichtigt.
3. Jedes UAV wird eindeutig einer Untermenge zugeordnet

Die Regeln garantieren, dass jedes UAV berücksichtigt wird und genau einmal innerhalb einer Sequenz den Abstand zu seinen Nachbarn korrigieren darf. Erst während der nächsten zyklisch wieder auftretenden Abfolge kann jedes UAV seine Abstandsfehler erneut korrigieren. Ein effizientes Vorgehen, das die Henneberg Operationen für die Unterteilung und die Berücksichtigung der drei Regeln nutzt, wird später beschrieben.

Der Algorithmus wird im Idealfall solange durchlaufen, bis alle Abstandsbedingungen erfüllt sind. Der Vereinfachung halber ist davon auszugehen, dass alle Abstandsbedingungen nach k Durchläufen gleichzeitig erfüllt werden können, denn sonst konvergiert die Lyapunov-Funktion gegen einen Wert, der ungleich 0 ist. Die Formation soll also realisierbar sein.

Um die Anzahl an Untermengen m so klein wie möglich wählen zu können, werden Erkenntnisse aus dem Bereich des Knotenfärbens genutzt. Auf den Formationsgraphen wird das Knotenfärbungsverfahren angewendet. Die Agenten werden anhand ihrer Färbung den Knotenteilmengen ϑ_i zugeordnet. Die erste Regel ist erfüllt, da es keine 2 adjazenten Knoten gibt, die gleich gefärbt sind. Somit gibt es auch keine zwei Nachbarn, die sich gleichzeitig bewegen dürfen. Jeder Knoten wird gefärbt und jede Knotenfärbung ist eindeutig. Somit sind Regel 2 und 3 erfüllt. Die minimal benötigte Anzahl an Farben wird auch Knotenfärbungszahl oder chromatische Zahl genannt. Von der Knotenfärbungszahl kann man daher auf die Anzahl der benötigten Knotenteilmengen m schließen. Allgemein ist die Knotenfärbungszahl bei minimal starren Graphen, $V > 2$, entweder drei oder vier. Bei einfachen Formationen, wie der Kolonnenformation, siehe Abbildung 3.7, kommt man sogar mit 2 Farben aus.

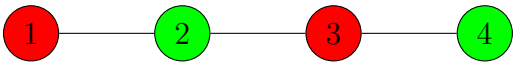


Abbildung 3.7: Kolonnenformation

3.4.2 Aktivierung der Teilmengen

In diesem Abschnitt wird die Idee vorgestellt, wie ein Agent erfährt, dass seine Teilmenge gerade in der zyklischen Abfolge ausgewählt ist und er aktiv werden darf.
Die den Agenten zur Verfügung stehende Taktdauer muss so gewählt werden, dass auf der einen Seite genug Zeit für Berechnungen und Korrekturen bleibt, aber auf der anderen Seite nicht zu lange auf Agenten gewartet wird, die eine überdurchschnittlich lange Strecke zurücklegen müssen. Wenn die zur Verfügung stehende Zeit abgelaufen ist, wird ein Timeout ausgelöst, auch wenn noch nicht jeder Agent den Wartezustand erreicht hat und signalisiert hat, dass er seine Korrektur beendet hat. Um auf einen globalen Takt und die dafür notwendige Zeitsynchronisation verzichten zu können, bietet sich folgende in dieser Arbeit überlegte Arbeitweise für die Agenten an. Mit dieser Herangehensweise wird die Annahme aus der Literatur [4], dass nur Abstandsmessungen zwischen den Agenten möglich sein sollen, aufgehoben.
UAVs vermerken in einem Register, ob sie gerade arbeiten oder warten. Wenn ein bestimmtes Flag im Statusregister gesetzt ist, arbeitet das UAV. Es löscht dieses erst, sobald es die Korrektur beendet hat und in den Wartezustand gewechselt ist. Das "Leader-UAV" wird vom UAV über die bestehenden W-LAN Kommunikationswege benachrichtigt, wenn dieses die Arbeit beendet hat. Die kürzesten Kommunikationswege wurden, nach der Konstruktion des Kommunikationsgraphen, mittels Dijkstra-Algorithmus bestimmt. Ein "Leader-UAV" übernimmt in-

nerhalb einer Formation verwaltungstechnische Aufgaben und gibt Ziele vor. Fast alle benachbarten UAVs orientieren sich an dem "Leader-UAV", beispielsweise bei der Geschwindigkeitsbestimmung. Das "Leader-UAV" kennt den Bewegungs-Ablaufplan und somit die Anzahl der einer Untermenge zugewiesenen Agenten. Wenn von jedem UAV der gerade aktiven Untermenge die "Ready-Nachricht" eingegangen ist, werden die UAVs der nächsten Untermenge aktiviert. Läuft hingegen der Timer ab, bevor alle "Ready-Nachrichten" eingegangen sind, wird an alle UAVs der momentan aktiven Untermenge die Anweisung gesandt, die Arbeit zu unterbrechen. Alternativ könnten die UAVs auch immer ihre ID im W-LAN-Datagramm mitschicken, damit das "Leader-UAV" weiß, von welchem UAV es noch keine Rückmeldung gab. Daher können gezielt diese UAVs zum Beenden ihres Korrekturvorgangs aufgefordert werden. Die Kommunikationswege über W-LAN sind nur bedingt zuverlässig. Das 3-Wege-Handshake-Protokoll wird genutzt, um Nachrichtenverluste zu erkennen und zu vermeiden. Dieses Protokoll sichert zu, dass sowohl die eigentliche Nachricht, als auch die Bestätigungsnachricht ankommen.

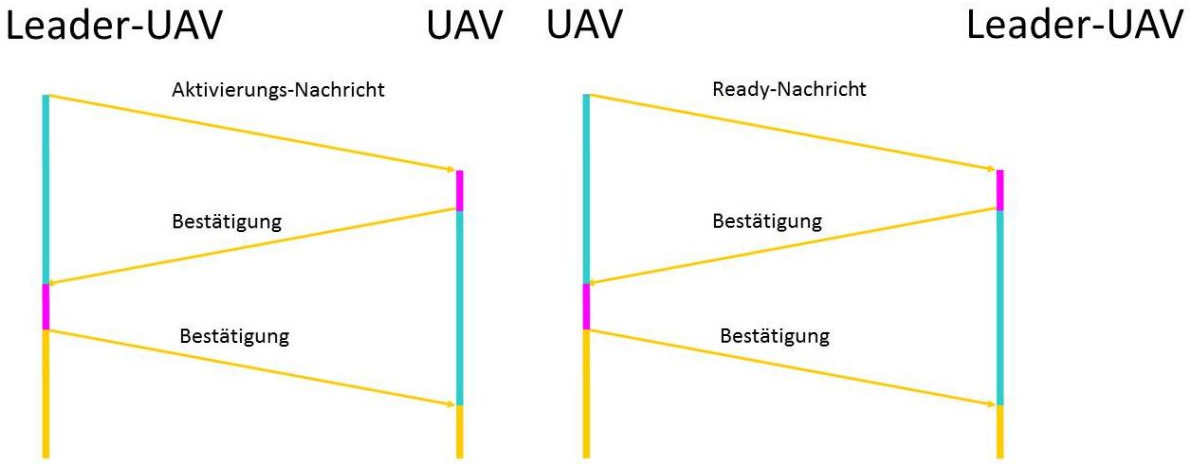
Das UAV sendet an das "Leader-UAV", dass es in den Wartezustand gewechselt ist, siehe Abbildung 3.8 b). Das "Leader-UAV" sendet, nachdem es die Nachricht empfangen hat, eine Bestätigung. Das UAV sendet ebenfalls nach Eingang dieser Bestätigung eine Bestätigung. Wenn nachdem k mal die Wartezeit abgelaufen ist und somit k mal das "Leader-UAV" die Nachricht wiederholt gesandt hat, immer noch keine Rückmeldung vom UAV empfangen wird, kann davon ausgegangen werden, dass das UAV nicht mehr erreicht werden kann. Das UAV ist entweder verloren gegangen oder dessen W-LAN ist ausgefallen. Das UAV, welches nicht mehr antwortet, ist nicht unbedingt verantwortlich für das Problem, da möglicherweise stattdessen ein UAV verloren gegangen ist, welches die Nachricht weiterleiten sollte. Möglich ist auch, dass das "Verbindungs-UAV" ausgefallen ist, während das UAV auf die Bestätigung vom "Leader-UAV" wartet. In diesem Fall würde das UAV ebenfalls maximal k mal wiederholt die "Ready-Nachricht" senden. Im schlimmsten Fall ist das W-LAN des "Leader-UAVs" ausgefallen.

Das "Leader-UAV" entnimmt dem Bewegungs-Ablaufplan, welche UAVs als nächstes aktiviert werden sollen. Diesen wird eine Nachricht gesandt, siehe Abbildung 3.8 a). Analog zu dem oben beschriebenen Vorgehen wird auch hier sowohl mit einer 1. Bestätigung, gesandt vom UAV, als auch einer 2. Bestätigung, gesandt vom "Leader-UAV", gearbeitet.

Sofern ein "Leader-UAV" ein UAV nicht mehr erreicht, wird versucht, über Funk Kontakt aufzunehmen. Wenn der Agent sich meldet, muss identifiziert werden, ob das W-LAN ausgefallen ist. Wenn es am W-LAN nicht liegen kann, wird Funkkontakt zu jedem UAV aufgenommen, das die Nachricht hätte weiterleiten müssen. Auch hier wird überprüft, ob das komplette UAV verloren gegangen ist,

oder nur das W-LAN ausgefallen ist. Beim Verlust eines UAVs muss die Formation geändert werden und der Bewegungs-Ablaufplan kann fürs erste verworfen werden.

Mit dieser vorgeschlagenen Herangehensweise hebt sich diese Arbeit von [4] ab, wo der Algorithmus von einem synchronen Verhalten hin zu einem asynchronen, zufallsbedingten Verhalten entwickelt werden sollte.



(a) "Leader-UAV" sendet Nachricht (b) UAV sendet Nachricht

Abbildung 3.8: 3-Wege-Handshake-Protokoll

3.4.3 Verbesserung des Algorithmus

Der Algorithmus wird effizienter, wenn die UAVs häufiger als einmal die Möglichkeit erhalten, bei einem Sequenzdurchlauf die Abstände zu korrigieren.
Die Regeln müssen folgende Annahmen berücksichtigen. Mindestens einer Untermenge wird jeder Agent zugeordnet. Der Vorteil ist, dass Agenten mehr als einer Untermenge zugeordnet sind und daher mehrmals beim Durchlaufen eines Zyklus aufgerufen werden. Von Vorteil ist dies vor allem für Agenten, die wenige Nachbarn haben. Der hieraus entstehende Nachteil ist jedoch, dass die Knotenfärbung nicht mehr der zugeteilten Untermenge entspricht.

alte Regeln:

1. In jeder Untermenge ϑ_i sind nur nichtzueinander adjazente Knoten enthalten

2. Alle Agenten werden berücksichtigt. $\vartheta = \bigcup_{i=1}^{m} \vartheta_i$
3. Jedes UAV wird eindeutig einer Untermenge zugeordnet

neue Regeln:

1. In jeder Untermenge ϑ_i sind nur nichtzueinander adjazente Knoten enthalten
2. Alle Agenten werden mindestens einmal berücksichtigt. $\vartheta = \bigcup_{i=1}^{m} \vartheta_i$
3. Agent v_k darf nicht zwei im Bewegungs-Ablaufplan aufeinanderfolgenden Untermengen ϑ_i und ϑ_{i+1} zugeordnet werden. Mindestens ein Nachbar von v_k muss ϑ_{i+1} zugeordnet sein, wenn v_k ebenfalls der Teilmenge ϑ_{i+2} zugeordnet werden soll.

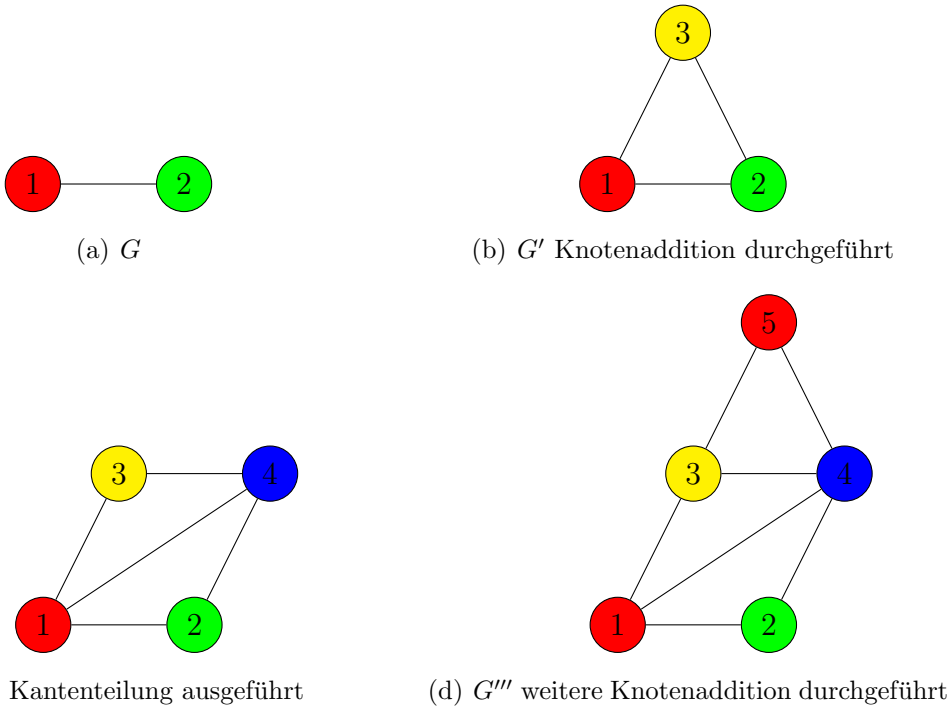
Die 1. Regel bleibt bestehen. Die 3.Regel garantiert, dass ein Agent sich erst wieder bewegen darf, wenn zwischenzeitlich mindestens ein Nachbar die Möglichkeit hatte, sich zu bewegen. Andernfalls würde das UAV wieder die gleiche Potentialfunktion bestimmen, die es dann auch nicht verbessern könnte. Ein Wegfall der 3. Regel hätte jedoch den Vorteil, dass UAVs, welche zuvor in ihrem Korrekturvorgang frühzeitig aufgrund des Timeouts unterbrochen worden sind, nun die Korrektur abschließen können. Die UAVs, auf die dies zutrifft, können den Lyapunov Funktionswert weiter verbessern. Der tatsächliche Nutzen dieser Verbesserung ist abhängig von der Formation und ist daher wie im nächsten Abschnitt deutlich werden wird, für den Simulator eher ungeeignet.

Der Algorithmus nähert sich schneller einem Lyapunov Funktionsgrenzwert, wenn die Kontrollstruktur des Formationsgraphen symmetrisch ist, als im asymmetrischen Fall [4]. Der Grund dafür ist, dass alle adjazenten Knoten in die Berechnung der Lyapunovfunktion miteinbezogen werden können, als wenn ein Agent nur eine Teilmenge der Nachbarn berücksichtigen dürfte.

3.4.4 Einteilung der Agenten in Untermengen

Im folgenden werden zwei Möglichkeiten vorgestellt, mit denen die Einteilung der Agenten in Untermengen ϑ_i gelingt. Der Beweis in [4] dafür, dass 4 Farben für die Knotenfärbung von jedem minimal starren Graphen ausreichen, liefert als Nebenprodukt eine Methode für die Einteilung der Agenten in disjunkte Mengen. Die bereits weiter oben eingeführten Henneberg-Operationen werden verwendet.

In Abbildung 3.9 wird anhand eines Beispiels die Agentenzuordnung mittels Knotenfärbung verdeutlicht. 4 Farben $\{c_i\}, i = 1, ..., 4$ mit c_1 = rot, c_2 = grün, c_3 = gelb, c_4 = blau stehen zur Verfügung, um den jeweiligen Knoten, der bei der Konstruktion hinzugefügt wird, zu färben. Beim Verwenden der Henneberg-Operationen bestimmt das beitretende UAV seine Farbe, nachdem es über Funk von den Nachbarn deren Farben zugesandt bekommen hat. Abhängig davon, welche Farbe nicht dabei ist, wählt es eine aus den verbleibenden. Der Vorteil hierbei ist, dass bei der Farbzuweisung kein "Leader-UAV" benötigt wird. Der Funkkanal wird anstelle des W-LANs genutzt, da der Kommunikationsgraph noch nicht das neue UAV berücksichtigt. Es ist nicht nötig, während der Formationskonstruktion, die aus mehreren Henneberg-Operationen besteht, bei jeder Hinzunahme eines Knotens einen temporären Kommunikationsgraphen zu konstruieren.



(a) G (b) G' Knotenaddition durchgeführt

(c) G'' Kantenteilung ausgeführt (d) G''' weitere Knotenaddition durchgeführt

Abbildung 3.9: Knotenfärbung während der Konstruktion einer Formation

Alle in der Menge $\{c_i\}$ enthaltenen Farben werden auch benötigt. Bei der Durchführung weiterer Knotenadditionen und Kantenteilungen werden ebenfalls keine weiteren Farben benötigt.

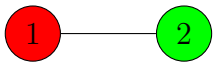
Das Beispiel zeigt, dass die Knotenfärbung mittels Henneberg-Operationen keine ideale Lösung liefert. Der Knoten 3 könnte, nach Ausführung der 3. Henneberg-Operation, der gleichen Teilmenge zugeordnet werden, in der bereits Knoten 2 ist, siehe Abbildung 3.9 d). Da die Kante e_{32} nach Ausführung der Kantenteilung, siehe 3.9 c), weggefallen ist, wird die Umfärbung des Knotens 3 ermöglicht. Wenn die

Formation nicht mehr erweitert wird, hätte ein nachträgliches Umfärben den Vorteil, dass eine Farbe weniger benötigt wird. Allgemein nimmt beim nachträglichen Färben der Knoten die Anzahl an möglichen Farbkombinationen mit der Anzahl an Knoten exponentiell zu. Bei der Färbung der Knoten von Formationsgraphen können, im Gegensatz zu der Anwendung, bei der Länder gefärbt werden, Sackgassen nicht auftreten [27]. Die nachträgliche Knotenfärbung eines Graphen ist NP-hart und für den Anwendungszweck des Simulators daher zu aufwendig.

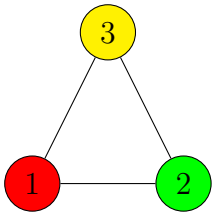
Die Alternative ist, dass die UAVs anhand der ID Teilmengen zugeordnet werden. Die Zuordnung muss nicht gleichmäßig stattfinden, da wie man auch dem Beispiel in der Abbildung 3.9 entnehmen kann, die Farben unterschiedlich häufig verwendet werden. So wird die Farbe blau ausschließlich einmal für den Knoten 4 benötigt. Man könnte nach den folgenden 4 Kriterien einteilen: gerade, ungerade, teilbar durch 3, teilbar durch 5. Diese initiale Zuordnung der Farben ist für die meisten Formationen möglich, kann jedoch dazu führen, dass die Formation wieder teilweise auseinander genommen werden muss, wenn die Vorgaben nicht mehr bei der Hinzunahme eines weiteren noch wartenden Agenten erfüllt werden können. Da die Formation weiterhin unter Verwendung der Henneberg Operationen konstruiert wird, kann es nicht vorkommen, dass zu wenig Farben zur Verfügung stehen. Wenn die initiale Farbzuordnung zu mindestens einem Konflikt geführt hat, muss das "Leader-UAV" eingreifen, nachdem es über den Kommunikationsgraph von benachbarten UAVs alarmiert wurde, welche die gleiche Farbe haben. Die identifizierten Sackgassen müssen behoben werden. Vorschläge für das Vorgehen hierfür, können [27] entnommen werden. Die Abbildung 3.10 zeigt eine Formationskonstruktion, bei der initial die Knotenfarbe vergeben ist. Ein UAV arbeitet in der Reihenfolge, wie die Kriterien in der Tabelle gelistet sind, die Bedingungen ab, bis ein Kriterium für die eigene ID zutrifft. Wenn das UAV mit der ID 6 nach einiger Zeit mittels Knotenaddition der Formation in einem weiteren Schritt beitritt, Abbildung 3.10 e), kommt es zu einem Konflikt.

Kriterium	Farbe	UAV
teilbar durch 3	gelb	3
teilbar durch 5	blau	5
gerade	grün	2,4
ungerade	rot	1

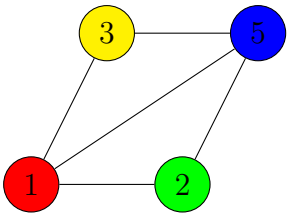
In der Anwendung müsste ermittelt werden, welche Kriterien am ehesten zu konfliktfreien Formationen führen und ob sich diese initiale Farbzuordnung tatsächlich auszahlt. Daher ist dieses entwickelte Verfahren in der jetzigen Form, im Hinblick auf die Effizienz, stark in Frage zu stellen.



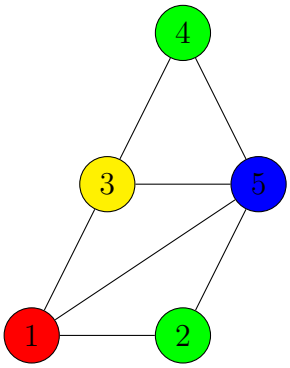
(a) G



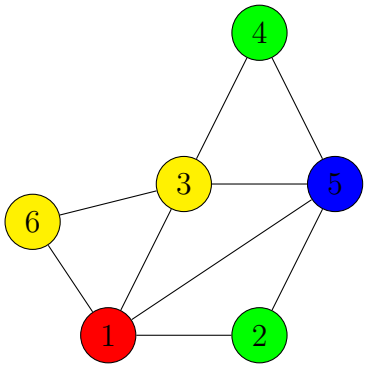
(b) G' Knotenaddition durchgeführt



(c) G'' Kantenteilung ausgeführt



(d) G''' weitere Knotenaddition durchgeführt



(e) G^4 weitere Knotenaddition durchgeführt

Abbildung 3.10: Konstruktion einer Formation mit initialer Knotenfärbung

3.4.4.1 Starten der "cyclic stop-and-go strategy"

Die "cyclic stop-and-go strategy"wird von den Agenten sofort genutzt, wenn alle Positionen innerhalb der Formation belegt sind. Das Erreichen dieses Zustands wird von dem "Leader-UAV", sobald dieses seine Position erreicht hat, mittels "3-Wege-Handshake-Protokoll" wiederholt überprüft. Wenn keine Rückmeldung kommt oder ein direkter Nachbar nicht mittels Sensoren erkannt wird, werden diese UAVs noch als fehlend vermerkt und die "Cyclic stop-and-go strategy"wird noch nicht gestartet.

Der Schwachpunkt der Überlegungen ist die Nutzung von Abstandsmesssensoren. In [12] ist die Funktionweise der für den STS-Simulator vorgeschlagenen akkustischen Sensoren beschrieben. Störquellen werden in [12] ebenfalls berücksichtigt und es wird ein Ansatz unter Verwendung einer Gaußverteilung geliefert, um die Sensormessdaten zu stören. Wie bereits erläutert, werden daher relative Abstände mittels W-LAN-Kommuniaktion ermittelt.

3.5 GPS

In diesem letzten Abschnitt des 3. Kapitels wird erläutert, woran UAVs sich während des Fluges orientieren, auf welche Mittel sie dafür zurückgreifen und wie eine eindeutige Positionsbestimmung ermöglicht wird. Abstände, die für die Ausführung der ungarischen Methode benötigt werden, können ebenfalls nur bestimmt werden, wenn das UAV eindeutig weiß, wo es sich befindet, wo das Ziel ist, und wie es dort optimal hinkommt.

Das "Leader-UAV" verfügt über die Missionszielkoordianten und gibt den UAVs Zielkoordianten vor, wenn diese die Formation verlassen sollen oder ihre Position innerhalb der Formation ändern sollen. Um die eigene relative Position zum Ziel zu betimmen, nutzt das UAV das über Satellit gesendete GPS-Signal, um seinen Standort zu bestimmen [16]. GPS ist die Abkürzung für Global Positioning System. Auch in Gebieten, in denen eigentlich die GPS-Signale nicht abgeschirmt werden oder anderweitig gestört werden könnten, kann es dennoch kurzzeitig zu Ausfällen kommen. Den UAVs bleibt bislang nichts anderes übrig als zu landen, wenn nach Ablauf einer bestimmten Zeit das Signal noch immer nicht wieder empfangen wird. Ein Blindflug ist ein zu großes Risiko. In [16] wird ein erweiterter Kalman Filter konstruiert, über den die Abstandsbestimmung zu UAVs in der näheren Umgebung möglich ist. Diese Orientierungshilfe führt, wie die Beispiele in Abbildung 3.11 und Abbildung 3.12 zeigen, bei ungünstigen Umständen nicht zu einer eindeutigen Positionsbestimmung. Die beiden Beispiele werden später

erläutert. Das UAV führt mehrere Messungen mit den optoelektronischen Sensoren hintereinander aus, um zu erkennen, ob die betrachteten UAVs sich ebenfalls bewegen. Wenn der 2-dimensionale Fall ausschließlich betrachtet wird, ist es hinreichend, wenn das UAV sich an mindestens 2 anderen UAVs orientiert, deren GPS-Signal nicht gestört ist. Die Abstandsmessungen werden ausgewertet, um die eigene Position zu bestimmen.

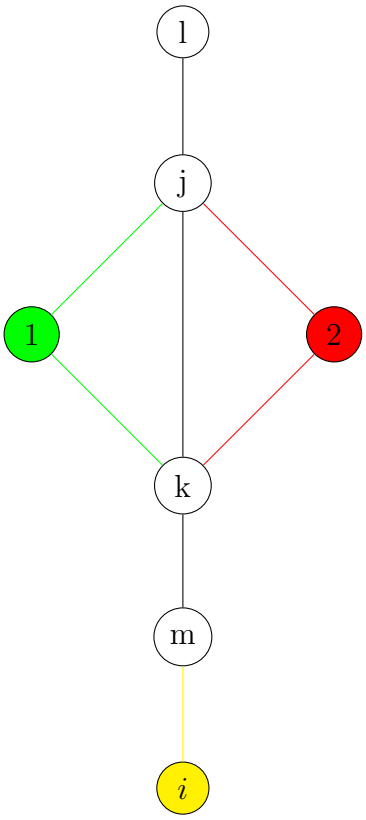


Abbildung 3.11: Agent i kann nicht eindeutig bestimmen, wo er sich befindet

3.5.1 "Flip ambiguity Problem"

Für die folgenden beiden Beispiele, die das "Flip ambiguity Problem" verdeutlichen sollen, wurden 2 Abbildungen aus [16] weiterentwickelt. In der genannten Quelle war ursprünglich von keiner zu Grunde liegenden Formation ausgegangen worden. Stattdessen war das Ziel, dass die Gruppe lose beieinander bleibt. Auch hier sollten mehrdeutige Positionsbestimmungen umgangen werden, da die UAVs Viodeoaufzeichnungen durchführen sollten. Wenn ein UAV der Auffassung ist, dass es den Bereich A gerade überfliegt, jedoch tatsächlich den Bereich B aufzeichnet, führt dies zu falschen Messungen.

Beispiel 1: Die Agenten l, j, k, i und m sind Mitglieder einer Kolonnenformation, siehe Abbildung 3.11. Agent i erhält vom "Leader-UAV" l den Befehl, eine neue Position innerhalb der Formation einzunehmen. Die gelb markierte Kante wird aufgehoben und das UAV soll sich zu der Zielposition 1, grüner Kreis, hinbewegen. Auf dem Weg zu dieser Zielposition verliert das UAV i das GPS-Signal. Alle anderen UAVs empfangen weiterhin das GPS-Signal. Ein in [16] konstruierter "erweiterter Kalman Filter" wird genutzt, um über die Abstandsbeziehungen Aussagen über die Position des UAVs machen zu können. Erklärungen, welche die Nutzung von Kalman Filtern im STS-Simulator betreffen, können [12] entnommen werden. Obwohl mehr als die geforderten 2 Knoten als Orientierungspunkte zur Vefügung stehen, ist das Ergebnis nicht eindeutig. Das UAV weiß nicht, ob es sich links oder rechts von den Agenten j und k befindet. Das vorliegende Problem wird als "flip ambiguity problem" bezeichnet. Es lässt sich lösen, indem die UAVs sich ihren bereits zurückgelegten Weg merken. Auch bei einer symmetrischen Formation spielt diese Doppeldeutigkeit eine Rolle, da ansonsten die Formationsvorstellung des "Leader-UAVs" sich von der tatsächlich vorliegenden Formation unterscheiden würde. Bei der Ausführung des nächsten Befehls, der beispielsweise eine weitere Knotenaddition fordert, könnte es zu einem Konflikt kommen, wenn einem UAV die Position 2, roter Kreis, als frei genannt wird, hier sich jedoch das UAV 1 befindet. Die gefärbten Kanten verdeutlichen, welche Kanten bei der Ausführung einer Knotenaddition hinzukommen. Die Möglichkeit, dass das UAV i, nachdem es die falsche Positionseinnahme bemerkt hat, den Fehler korrigieren sollte, ist zu aufwendig.

Beispiel 2: Die Formation, von der in der Abbildung 3.12 nur ein Ausschnitt gezeigt wird, ist symmetrisch. Das Ziel des UAVs i ist die Durchführung einer Knotenaddition am Rande der Formation. Das UAV i nähert sich den Agenten j und k, die im Zentrum der Formation ihre Position haben. Zu dem Zeitpunkt, bei dem das UAV an der rot markierten Stelle ist, geht das GPS-Signal verloren. Das UAV bestimmt mithilfe des erweiterten Kalman Filters seine Position. Sobald jedoch die grün markierte Stelle passiert wurde, gibt es zwei mögliche Positionen, wo sich das UAV gerade befinden könnte, wenn es die Abstandsmessdaten ausgewertet hat. Die beiden gelb markierten Kreise stellen die Doppeldeutigkeit zu einem Zeitpunkt dar und die 2 gelben Kanten verdeutlichen in der Abbildung die beiden möglichen Flugbahnen. Der Grund für die Ambiguität ist, dass die UAVs j und k kollinear sind. Es ist nicht schwierig vorauszusagen, dass das UAV seine Bewegungsrichtung nicht ändern wird, wenn es die grün markierte Stelle erreicht hat. Da das UAV jedoch ausschließlich seine Abstandsmessdaten nutzt, um die Position fortlaufend neu zu bestimmen, sind, wie im ersten Beispiel, die vorangegangenen Bewegungsaktionen, wie das Drehen, bei der Positionsbestimmung nicht berücksichtigt. Umwelteinflüsse können auch zu Bewegungsänderungen führen, deren Auswirkungen schwer vorhersehbar sind.

Um jede Mehrdeutigkeit auszuschließen, reicht es im 2-dimensionalen Raum aus, wenn mindestens 3 nichtkollineare Bezugs-UAVs zur Verfügung stehen.

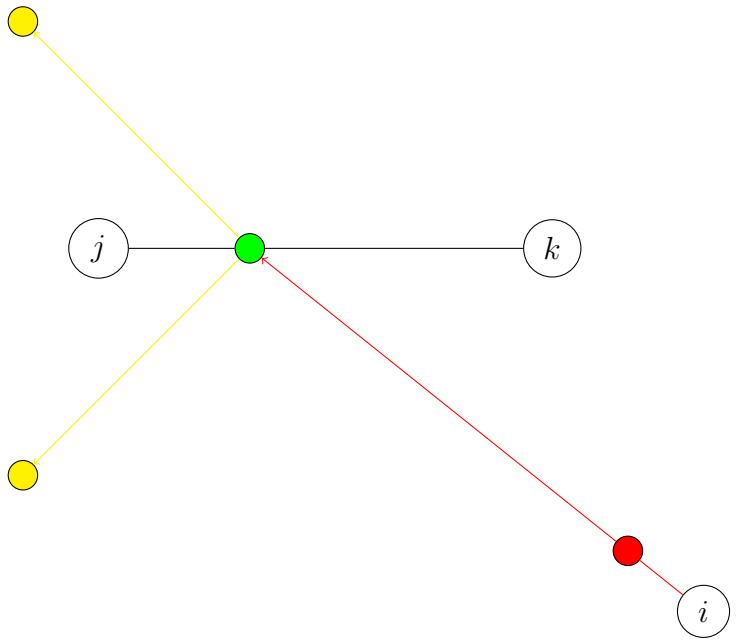


Abbildung 3.12: UAV i versucht sein Ziel zu finden, indem es sich an den anderen UAVs orientiert

Kapitel 4

Dynamische Reaktionen der Formation

In diesem Kapitel werden unterschiedliche Möglichkeiten erläutert, die den UAVs zur Verfügung stehen, um an einem Hindernis vorbeizufliegen. Hindernisse werden von Sensoren erkannt. Die Alternativen unterscheiden sich dahingehend, ob die Formation als Einheit das Hindernis umfliegen soll oder sich aufteilen sollte. Außerdem besteht die Möglichkeit, dass die UAVs sich aus ihrer starren Formation lösen, um kurzzeitig eine andere Formation anzunehmen, die ein geschickteres Ausweichen ermöglicht. Vorgestellt werden Konzepte, um die Formation aufzuteilen und später wieder unter der Bedingung herzustellen, dass minimale Starrheit nicht verloren geht.

4.1 Kollisionskontrolle

Um auf Umwelteinflüsse und auf mittels Sensoren erkannte Hindernisse reagieren zu können, ist es notwendig, dass die starre Formation aufgelöst werden kann, damit die UAVs sich den neuen Bedingungen anpassen können.

Die Sensoren der UAVs entdecken ein Hindernis, dessen Geometrie mit einer bekannten Hindernisgeometrie übereinstimmt. Unter Hinzunahme von Erfahrungswerten werden potentielle Möglichkeiten durchgespielt, wie die UAVs auf das Hindernis intelligent reagieren können. Die benötigten Informationen sind in der Datenbank eines jeden UAVs gespeichert. Da die UAVs über verschiedene Erfahrungswerte verfügen, die sie selbst in die Datenbank geschrieben haben, können

sie dank vieler, unabhängiger Meinungen sich demokratisch auf eine Ausweichstrategie einigen. Um Zeit zu sparen und die einzelnen UAVs nicht beanspruchen zu müssen, könnte auch das "Leader-UAV" die Verantwortung komplett übernehmen. Man beachte, dass die Informationslage nie vollständig sein kann.

Es könnte effizienter sein, eine bestehende Dreiecksformation aufzuheben, um eine neue Formation zu erstellen. In Kolonnen-Formation ließe sich eine enge Passage durchfliegen. Der Formationsfehler ist ein Maß für die Abweichung zwischen zwei verschiedenen Formationen [5]. Die Struktur der Formation wird nicht geändert, wenn die Agenten temporär gültige Zielkoordianten bestimmen, um das Hindernis geschlossen zu umfliegen. Diese Methode ist zeitintensiv, da die Ausweichstrecke einem Vielfachen des direkten Weges entsprechen könnte. Aus Gründen der unzureichenden Information ist dies nicht auszuschließen. Wenn eine Formation ein geschlossenes, bodennahes Hindernis erkennt, ergibt es keinen Sinn, die Formation aufzuteilen. Stattdessen sollte die Flughöhe angehoben werden. Die Formation zu teilen, damit eine Teilmenge der Agenten sich auf der einen Seite um das Hindernis herumbewegt und die andere Teilmenge auf der anderen Seite, ist ebenfalls eine Variante [5].

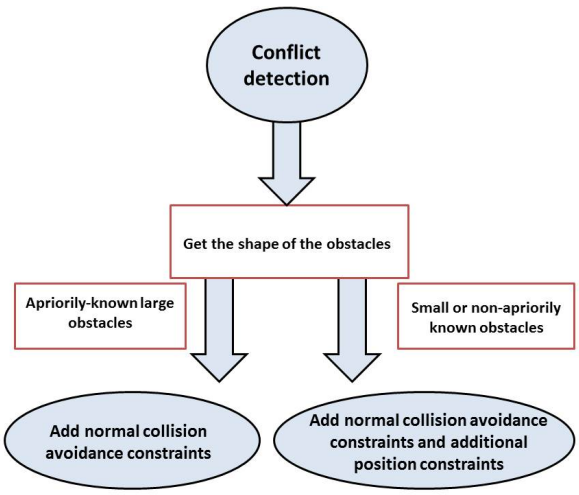


Abbildung 4.1: Einordnung des Hindernisses (aus [21])

Nachdem ein Hindernis erkannt wurde, das auf der Flugbahn von mindestens einem der UAVs liegt, laufen die folgenden Vorgänge ab, damit die optimale Ausweichstrategie ermittelt wird. Die Hindernisstruktur wird identifiziert, siehe Abbildung 4.1. Wenn es sich bei dem Hindernis um ein großes und bereits bekanntes Hindernis handelt, wird ein Standardausweichverfahren gestartet. Wenn es sich hingegen um ein kleines oder unbekanntes Hindernis handelt, wird das gewählte Standardausweichfahren noch angepasst. Beispielswiese könnte es ausreichen, die starre Formation eine Zeit lang aufzuheben, damit betroffene UAVs vor dem kleinen Hindernis ausweichen können, während die anderen UAVs ihre Flugbahn nicht verlassen. Vor größeren Hindernissen müssen logischerweise alle Formationsmitglieder ausweichen.

4.1.1 Teilung einer Formation

Einige Abstandsbedingungen werden aufgehoben, damit Kanten wegfallen können. Agenten, die zuvor inzident zu nun weggefallenen Kanten waren, behalten die aufgehobenen Formationsbedingungen im Speicher, damit zu einem späteren Zeitpunkt, die alte Formation wieder konstruiert werden kann. Dieses Konzept wird später verworfen, wenn Starrheit innerhalb der Einzelformationen gefordert wird. Ein weiterer Grund für ein Aufteilen der Formation könnte neben dem Ausweichen vor einem Hindernis sein, dass neue Unterziele oder zeitlich parallel weitere Ziele verfolgt werden sollen. Überwachungs- oder Patrouillenaufgaben sind übergeordnete Aufgaben. Während der Ausführung können sich Unterziele dynamisch entwickeln. Es können Forderungen bezüglich der Eigenschaften an die zu konstruierenden Teilformationen gestellt werden. Die Agenten beider Teilformationen sollen sich wieder in starren oder minimal starren Formationen aufhalten, damit die Teilung keinen Effizienzverlust zur Folge hat. Hierfür werden neue Kanten eingefügt, die zu neuen Abstandsbedingungen führen. Die hinzugefügten Kanten sind in erster Linie inzident zu den Knoten, deren Knotengrad nach dem Entfernen der Kanten kleiner geworden ist.

Die Abbildung 4.2 a) - c), entnommen [5], zeigt beispielhaft die bei der Teilung ablaufenden Vorgänge.

4.1.2 Vereinigung zweier Teilformationen zu einer Formation

Nach erfolgreichem Passieren des Hindernisses vereinigen sich die Teilformationen aus dem Beispiel von Abbildung 4.2. In diesem Abschnitt werden 2 Alternativen vorgestellt, mit denen 2 Teilformationen zu einer Formation wieder vereinigt werden.

Die nach der Aufteilung in den beiden Teilformationen hinzugefügten Kanten müssen nicht wieder entfernt werden. Die Kanten wurden zuvor hinzugefügt, um wieder Starrheit vorliegen zu haben. Die ursprüngliche Formation G wird nur im Hinblick auf die relative Positionierung der Knoten rekonstruiert. Neue gewichtete Kanten werden eingefügt, welche Agenten der beiden Teilmengen verbinden. In dem Beispiel in Abbildung 4.2 b) werden 5 Kanten beim Teilen entfernt, zu je einer Teilformationen kommt 1 Kante hinzu und 3 Kanten werden beim Vereinigen wieder hinzugefügt. An der Gesamtanzahl an Kanten hat sich nichts geändert, genauso wenig wie an der Starrheit. Dies ist auch die Begründung dafür, weshalb der ursprüngliche Formationsgraph nicht bezüglich der Kanten rekon-

struiert wird. Die Kanten stellen die einzuhaltenden Abstandsbedingungen dar, damit Starrheit vorliegt. Es gibt jedoch auch oftmals andere Möglichkeiten Kanten vorzugeben, damit bezüglich der Knoten die gleiche Positionierung vorliegt. Enstanden ist somit eine andere Formation, deren Mitglieder aber die gleichen Aufgaben ausführen können, wie bei der urpünglichen Formation G.

Zusätzliche Kanten nach der Formationsvereinigung im Vergleich zum Ausgangsgraph hätten zu einer längeren Rechendauer bei der Ausführung der Algorithmen geführt und auch der Steuerungsaufwand würde zunehmen. Einer Faustformel nach, entnommen [5], reicht es aus, genau 3 Kanten zwischen die Teilformationen hinzuzufügen, um eine starre Gesamtformation zu erhalten. Zu berücksichtigen ist, dass die 3 Kanten zu mindestens je 2 disjunkten Knoten aus beiden Formationen inzident sind. Dies bedeutet, dass bespielsweise die Kanten e_{16}, e_{15} und e_{17} nicht zulässig sind. Die UAVs entnehmen entweder einer Datenbank die vordefinierten Abläufe oder zufällig werden Kanten hinzugefügt und es wird von einem zuvor bestimmten "Leader-UAV" überprüft, ob die Änderung der Formation zu dem gewünschten Ergebnis geführt hat. Es wird hierzu die Starrheitsmatrix aufgestellt.

Die Tabelle fasst zusammen, welche Kanten während der Formationsänderungen in welcher Form betroffen sind.

ENTFERNTE e_{ij}	NEU HINZUGEKOMMENE e_{ij}	ZEITWEISE ENTFERNTE e_{ij}
e_{63}	e_{34}	e_{35}
e_{35}	e_{57}	e_{47}
e_{52}	e_{16}	
e_{54}		
e_{47}		

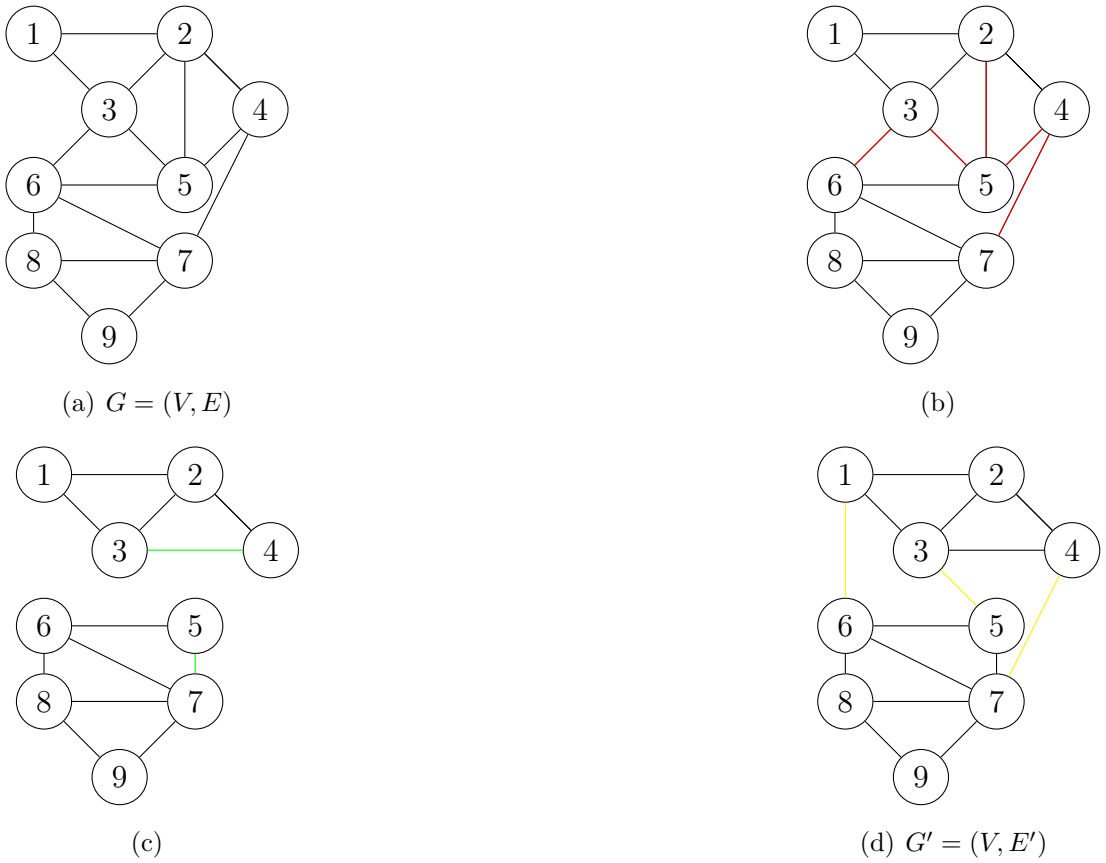
4.2 Skalierung der Abstandsvorgaben

Die UAVs müssen von ihrer vorgegebenen Fluglinie nicht abweichen, wenn es möglich ist, mittels Skalierung der Abstandsvorgaben, die Größe der Formation soweit zu verringern, dass zwischen 2 Hindernissen hindurchgeflogen werden kann. In [2] wird die Formation G als Skaleninvariant bezeichnet, wenn jede Abstandsforderung mit dem Faktor α reduziert werden kann. Es gibt eine gewünschte Abstandsmenge D', für die soll

$$D' = \alpha D \tag{4.1}$$

gelten und außerdem wird

$$\|x_i - x_j\| = \alpha d_{ij} \tag{4.2}$$



(a) $G = (V, E)$ (b) (c) (d) $G' = (V, E')$

Abbildung 4.2: a) Formation vor der Teilung. b) rot markierte Kanten werden entfernt. c) grün markierte Kanten werden neu hinzugefügt, damit beide Teilformationen starr sind. d) Teilformationen sind vereinigt. Die 3 gelb gefärbten Kanten sind so gewählt, dass Starrheit erreicht ist.

erfüllt. Der Abstand αd_{ij} zwischen Agent i und Agent j soll erreicht sein, wenn das UAV i an der Stelle x_i ist, während das UAV j gerade an der Stelle x_j im 2-dimensionalen Raum sein soll. In der Abbildung 4.3, entnommen [5], ist das UAV 1, rot gefärbt, gerade an der Position x_i und das UAV 2, grün gefärbt, an der Position x_j. Da die beiden UAVs an diesen vordefinierten Koordianten gerade die neue Abstandsbedingung αd_{ij} erfüllen, durchfliegen sie automatisch das Hindernis, das aus 2 Rechtecken besteht. Die Formation in der Abbildung erfüllt zusätzlich die Bedingung "translational invariant" [5]. Um die Strecke $\tau \in \mathbb{R}^2$ soll sich die Formation weiterbewegen, ohne dass sie im $\mathbb{R}^2$ rotiert. Diese Forderung wird erreicht, wenn jedes UAV i den Zielpunkt x_i erreicht hat, nachdem es vom Ausgangspunkt ξ_i genau die Strecke τ zurückgelegt hat. τ ist für jedes UAV gleich. Kein UAV hat eine längere Strecke aufgrund einer zusätzlichen Drehung zurückgelegt.

$$x_i = \xi_i + \tau \text{ für alle } i = 1, ..., n \tag{4.3}$$

Der Nachteil bei dieser idealen Forderung ist, dass nicht berücksichtigt wird, dass Umwelteinflüsse dazu führen, dass die einzelnen UAVs bei ungewollten Änderungen gegenlenken. Die zurückgelegte Strecke kann daher eigentlich nie genau τ sein.

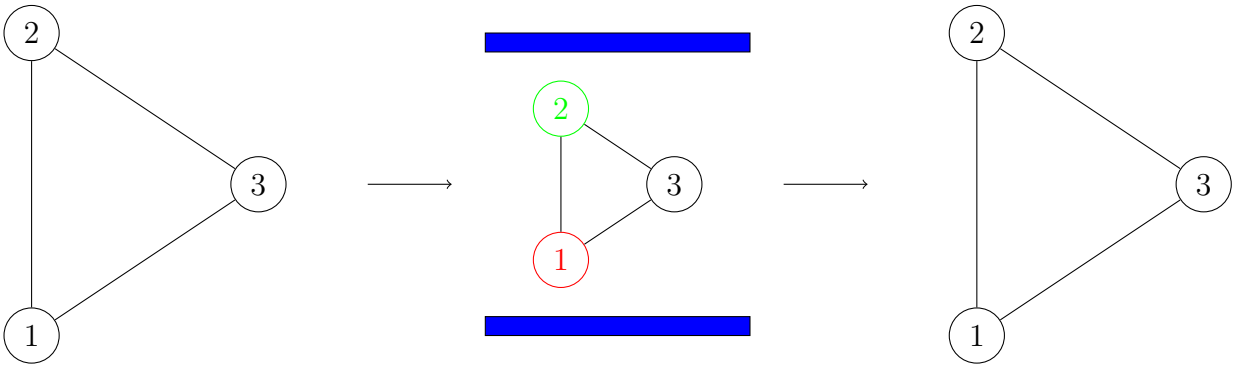


Abbildung 4.3: Durchfliegen eines Hindernisses

Skaleninvarianz ist im Grunde für jeden Formationsgraphen erfüllt. Alle Eigenschaften des Graphen sind vereinbar mit der Skalierung. Es ändern sich ausschließlich die konkreten Abstandswerte. Skalierungsinvarianz und Starrheit schließen sich gegenseitig aus.

4.3 Erstellen größerer Graphen aus minimal starren Untergraphen

Im obigen Abschnitt "Henneberg Folge: Erweiterung eines minimal starren Graphen" und in [7] werden grundlegende, auf einen Graphen anwendbare Operationen definiert. Diese Operationen werden genutzt, um die Formationsteilung und Vereinigung zu beschreiben. Nach der Ausführung einer Henneberg-Operation auf einen starren, beziehungsweise minimal starren Graph G, ist ebenfalls der nun konstruierte Graph G' starr, beziehungsweise minimal starr. Alle Kanten eines minimal starren Graphen sind stets unabhängig.

4.3.1 Vereinigung zweier Untergraphen zu einem starren Graphen

Die Vereinigung zweier Untergraphen zu einem starren Graphen ist nicht immer sofort möglich. Die beiden Subgraphen werden vereinigt mittels Hintereinanderausführung von 2 Operationsschritten [7]. Das Ziel des Verfahrens ist es, dass eine Kante des Untergraphen G_1 mit einer gleichlangen Kante des Untergraphen G_2 verschmolzen wird. Einige Abstandsbedingungen innerhalb eines Untergraphen müssen dynamisch geändert werden, wenn es keine 2 Paar Agenten gibt, deren Abstände jeweils gleich sind und die sich am Rand der Formation befinden. In der Anwendung ist der Erfolg dieses Verfahrens davon abhängig. Die "Leader-UAVs" der beiden Teilformationen tauschen die Graphentopologie über Funk oder W-LAN aus. Beide UAVs vereinbaren, an welcher Kante die Formationen verschmelzen sollen. Die "Leader-UAVs" geben den Formationsmitgliedern Bewegungsanweisungen, damit sich die Formationen so positionieren, dass ein Verschmelzen möglich wird.

Um ein Hindernis zu passieren, hat sich die Formation G auf die Teilformation $G_1 = (V_1, E_1, w_1)$ und die Teilformation $G_2 = (V_2, E_2, w_2)$ aufgeteilt. Die Knotenteilmengen V_1 und V_2 sind disjunkt.

$$V_1 \cap V_2 \tag{4.4}$$

G_1 und G_2 sind "edge-attachable", wenn es 2 Kanten gibt, welche die gleiche Länge haben. Es gibt eine Kante $e_{ij} \in E_1$ und eine Kante $e_{kl} \in E_2$, sodass $\|e_{ij}\| = \|e_{kl}\|$ gilt. Die "Kantenanfügen-Operation" lässt sich wie folgt schreiben:

$$G' = G_1|_{e_{ij}} \oplus G_2|_{e_{kl}} := G_1 \cup Ren([G_2]^-_{e_{kl}}; v_k, v_l | v_i, v_j) \tag{4.5}$$

Die Untergraphen G_1 und G_2 werden über die miteinander zu verschmelzenden Kanten e_{ij} und e_{kl} verbunden. Der neue Graph G' setzt sich zusammen aus G_1 und dem angepassten G_2. Die Kante e_{kl} wird mittels der allgemein gültigen Kantensubtraktions-Operation $G^- := (V, E \setminus \{e\})$ entfernt, da in G' sonst die Kante doppelt vorliegen würde. Innerhalb der Teilformation G_2 werden die Agenten v_k und v_l umbenannt in v_i und v_j. "Ren", wie "Rename" weist auf diese Operation hin, die ausgeführt werden muss, bevor die beiden Graphen vereinigt, $G_1 \cup G_2$, werden können. Die Knoten v_k und v_l waren zu der entfernten Kante inzident. Die neue Formation G' hat $n = (n_1+n_2)-2$ Knoten und $E = E_1+E_2-1$ Kanten. Der Aufwand ist gerechtfertigt, da der Graph starr bleibt. Wichtig ist, dass diese Operation nur genuzt wird, wenn die ursprüngliche Formation nicht rekonstruiert werden soll.

4.3.2 Übertragen auf Agenten in Simulationsumgebung

Der in [7] aufgezeigte Weg berücksichtigt nicht, dass in der Anwendung die UAVs, abstrakt in der Theorie als Knoten dargestellt, nach Umbenennung nicht miteinander verschmelzen können. Die eigene, hierfür entwickelte Erweiterung ist daher, dass die umzubenennenden Knoten an neuen Positionen mittels der Henneberg-Operation Knotenaddition oder Kantenteilung in die Formation eingebettet werden. Während der Vorgänge "Verschmelzen" und "neue Position suchen" bewegt sich das System. Die Abstandsbedingungen werden nicht die ganze Zeit eingehalten, da 2 UAVs ihre Position verlassen, damit 2 UAVs der andockenden Teilformation diese übernehmen. Die Formation ist daher kurzzeitig flexibel.

4.3.3 Beispiel für Formationsvereinigung mittels Kantenverschmelzung

Eine Formation hat sich so aufgeteilt, dass das Hindernis von zwei Dreiecksformationen, G_1 und G_2 umflogen wird. Dargestellt wird die folgende Formationsvereinigung in Abbildung 4.4.

$G_1 = (\{v_1, v_2, v_3\}, \{e_{12}, e_{23}, e_{31}\})$

$G_2 = (\{v_4, v_5, v_6\}, \{e_{45}, e_{56}, e_{64}\})$

wobei $\|e_{12}\| = \|e_{56}\|$

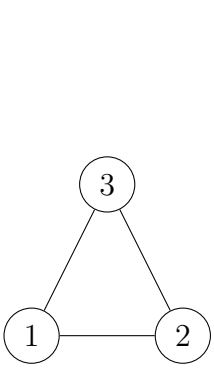
Die "Kantenanfügen-Operation" (4.5) wird auf die beiden Teilgraphen angewen-

det.

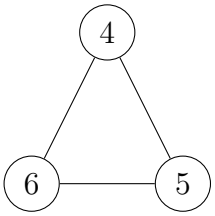
$G' = G_1|_{e12} \oplus G_2|_{e56}$

G' verfügt nun über die Knoten $V = v_1, v_2, v_3, v_4$ und die Kanten $E = \{e_{12}, e_{23}, e_{31}, e_{42}, e_{14}\}$

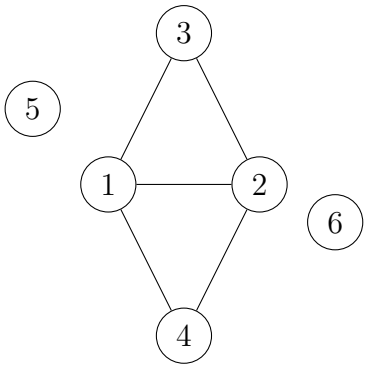
Die beiden zuvor eigentlich umzubenennenden Knoten müssen nun mittels zweimaliger Anwendung der Knotenaddition wieder in die Formation eingefügt werden.



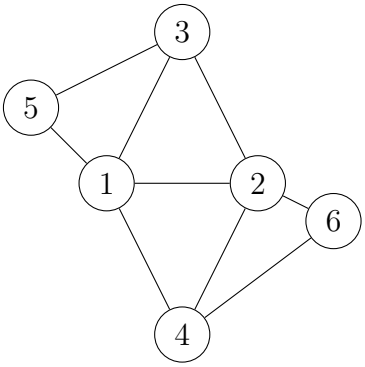
(a) G_1



(b) G_2



(c) Kanten verschmolzen



(d) 2 Knotenadditionen

Abbildung 4.4: 2 Formationen werden miteinander verschmolzen

4.3.4 Z-Link

Unter Verwendung eines Z-Links, siehe Abbildung 4.5, ist es möglich, eine Kante von G_1 mit einer Kante von G_2 verschmelzen zu lassen, ohne dass zusätzlich zwei Henneberg-Operationen durchgeführt werden müssten. Ein Z-Link [7] ist ein bipartiter Graph, ein $K_{2,2}$. In beiden Teilmengen befinden sich je zwei Knoten

und es gibt 3 Kanten. Nachdem 2 Teilgraphen, die jeder für sich bereits starr gewesen sind, mittels Z-Link verbunden wurden, ist dieser konstruierte Graph ebenfalls starr. Durch geeignetes Wählen der Kantenlängen des Z-Links entfällt die bisherige Voraussetzung, dass es 2 gleich lange Kanten geben muss.

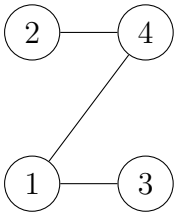


Abbildung 4.5: Z-Link: $Z = (V_Z, E_Z)$

Die Graphen $G_1 = (V_1, E_1)$ und $G_2 = (V_2, E_2)$ werden mithilfe eines Z-Links verbunden. Der Index unter dem Operator-Zeichen weist auf die Nutzung eines Z-Links hin.

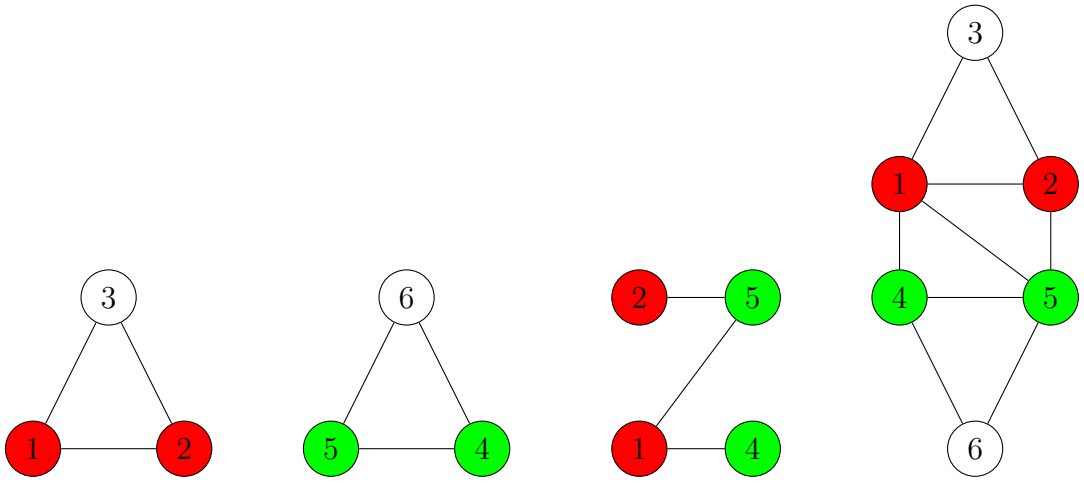
$$G' = G_1|_{e_1} \oplus_Z G_2|_{e_2} \tag{4.6}$$

Es gilt $G' = G_1 \cup Z \cup G_2$. Innerhalb der Vereinigungsmenge spielt es keine Rolle, wenn Elemente mehrfach vorkommen. Die Kanten e_1 und e_2 sind inzident zu jenen Knoten, die mit denen des Z-Links verschmelzen.

Die Knoten des Z-Links sollen die Endpunkte der in der jeweiligen Formation vorliegenden Kante e_1 bzw. e_2 werden [7]. Folgende Notation ermöglicht Rückschlüsse auf das Aussehen des benötigten Z-Links.

$$V_Z = V(e_1) \cup V(e_2) \tag{4.7}$$

G' ist starr, wenn bereits G_1 und G_2 starr waren. Das nun umgangene Problem war, dass Agenten darstellende Knoten nach der Umbenennung doppelt vorkommen würden. Da die Knoten des Z-Links verschmelzen, gehen keine Knoten der Teilformationen verloren, siehe auch folgendes Beispiel in Abbildung 4.6. Die Formation G' ist wieder minimal starr. Dies kann man ohne Rechnung vermuten, da es minimal starre Teilgraphen gibt. Siehe hierzu auch Lamans Theorem bzw. Abbildung 2.7.



(a) G_1 (b) G_2 (c) Z-Link (d) G'

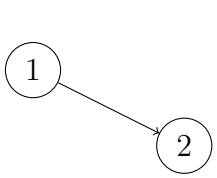
Abbildung 4.6: 2 Formationen werden über einen Z-Link zusammengeführt

4.4 "Leader-follower"-Ansatz

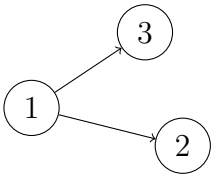
Ein UAV kann über mehr Informationen verfügen und mehr Kompetenzen zugesprochen bekommen, als andere Mitglieder der Formation. Dieses UAV wird in [1] als "Leader-UAV" bezeichnet. Der Ausgangsgrad $d_G^-(v)$ eines "Leader-UAVs" ist 0. Der Eingangsgrad $d_G^+(v_6)$ des "Leader-UAV"s beim "wheel-Graph" ist 5 und dessen Ausgangsgrad $d_G^-(v_6)$ ist 0. Ein "wheel-Graph" wird in Abblidung 4.10 a) auf Seite 79 dargestellt. Das "Leader-UAV" kann die Geschwindigkeit für alle anderen UAVs vorgeben und verfügt über die zur Erfüllung der Aufgaben notwendigen Kenntnisse. In [4] wird mittels Verwendung der Cayley-Menger Matrix ein Weg beschrieben, damit ein "Follower-UAV" die Geschwindigkeit des "Leader-UAVs" unter ausschließlicher Nutzung von Abstandsmessungen bestimmen kann. Das "Leader-UAV" kann seine Geschwindigkeit ändern, ohne dies ständig den anderen UAVs mitteilen zu müssen. Für das "Leader-UAV" wird ein sehr leistungsfähiges und robustes UAV gewählt. Es wird viel Rechenleistung benötigt, um mit mehreren UAVs gleichzeitig zu kommunizieren und ausreichend Speicher, um über alle notwendigen Formationsinformationen verfügen zu können. Im TU-Quadrokopter wird ein Intel Atom Prozessor mit 1,6 GHz verwendet. Dieser wird bislang nur zu 10% ausgelastet [25]. Falls das "Leader-UAV" während des Einsatzes aus energietechnischen Gründen zur Basis zurückkehren muss, wird rechtzeitig zuvor ein neues UAV als "Leader-UAV" ausgewählt. Ein "First-Follower-UAV" könnte gewählt werden, da der Kommunikationsweg sehr kurz ist, wenn alle wichtigen Daten von dem ehemaligen "Leader-UAV" zum neuen "Leader-UAV" kopiert werden. Wenn das "Leader-UAV" Nachrichten oder Befehle den anderen

UAVs sendet, wird der Kommunikationsgraph genutzt. "Leader-UAVs", die über sehr gute Sensoren für die Kollisionsabfrage verfügen, können unter Umständen die komplette Formation beschützen. Die anderen UAVs verfügen nun über mehr Kapazitäten, um andere Sensordaten, wie Bodenmessdaten aufzuzeichnen und auszuwerten.

In [1] wird zwischen zwei "Leader-follower"-Ansätzen, unterschieden. Wenn der Eingangsgrad eines "Leader-UAVs" 1 ist, gibt es nur ein "Follower-UAV". Dieses "Follower-UAV" versucht sowohl den Abstand d_{12} zu dem "Leader-UAV" zu halten, als auch den Winkel ψ_{12}. In diesem Ausnahmefall ist es notwendig, auch den Winkel zu berücksichtigen, da ansonsten der Freiheitsgrad des "Follower-UAVs" 1 sein würde und daher Persistenz nicht vorliegen würde. Dieser Ansatz wird auch $d - \psi$ Steuerung genannt [3]. Der Eingangsgrad des "Follower-UAV"s ist nicht beschränkt. Das zweite Muster, das in [3] erwähnt ist, wird als $l - l$ Steuerung bezeichnet, da ein UAV 2 "Leader-UAVs" folgt, während es versucht die beiden Abstände d_{13} und d_{12} zu halten. Diese Steuerung ist jedoch auch von anderen Parametern abhängig und ungeeignet für das UAV-Projekt, da es in Formationen immer nur ein "Leader-UAV" gibt.



(a) $d - \psi$ Steuerung



(b) $l - l$ Steuerung

Abbildung 4.7: "Leader-follower"-Ansätze

4.4.1 Wahl eines "Leader-UAVs"

Wenn das bisherige "Leader-UAV" nicht länger zur Verfügung steht, muss abhängig von Kriterien ein neues bestimmt werden. Möglicherweise will eine Formation, die bislang ohne "Leader-UAV" gearbeitet hat, wie die C^2-Formation, siehe Abbildung 4.10 b), nun ein "Leader-UAV" bestimmen und dafür die Formation anpassen. Um das geeigneteste UAV zu bestimmen, muss die Formation sich abstimmen. Hierzu wird zuerst das "Rendezvous-Problem" gelöst [2]. Die Agenten stellen zu jedem anderen Knoten eine Kommunikationsverbindung her. Aussagen über die UAV-Bewegung werden nur soweit gemacht, als dass sich die UAVs zueinander hinbewegen, bis der W-LAN-Empfangsbereich ausreicht. Wenn von einem Graphen gefordert wird, dass dieser keine Kantenlänge besitzen darf, die

größer als Δ ist, wird dieser Graph als " Δ-disk proximity graph" bezeichnet. Die maximale Kantenlänge Δ entspricht dem W-LAN-Empfangsradius. Der Graph ist komplett, sobald die Kardinalität von jedem Knoten genau $n-1$ ist.

$$card(v_i) = n - 1 \text{ für alle } i = 1, ..., n \tag{4.8}$$

Jedes UAV, das mit $n-1$ weiteren UAVs verbunden ist, sendet eine Broadcast-Nachricht, mit der dies mitgeteilt wird. Die Nachricht geht nur einen "hop" weit. Empfängt ein Agent diese Nachricht von allen $n-1$ Agenten, so ist ohne zentrale Steuerung bekannt, dass das Rendezvous-Problem gelöst ist. Nun kann das geeigneteste UAV bestimmt werden. Kriterien können Robustheit, technische Ausstattung und Energiereserve sein. Daraufhin wird der Formationsgraph mittels ungarischer Methode gebildet.

Wenn die Formation ein für die Ausführung der Mission potentiell wichtiges Gebiet identifiziert, das nicht auf der Flugbahn liegt, wird die Formation, abhängig von Prioritätswerten, ihre Bahn verlassen. In [15] wird ein Ansatz vorgestellt, um ein neues "Leader-UAV" zu bestimmen, das die Ausführung der temporären Mission am optimalsten leiten kann.

4.4.2 Testen auf Existenz eines "Leader-UAV"s

In [2] wird erläutert, wie ein Graph G dahingehend untersucht wird, ob dieser einen Wurzelknoten v_r besitzt, von dem aus jeder andere Knoten des Netzwerkes über gerichtete Kanten erreicht werden kann. Wenn der Graph ebenfalls azyklisch ist, spricht man von einem "rooted out-branching digraph". Hier ist das "Leader-UAV" der Wurzelknoten. Nur wenn ein "rooted out-branching digraph" vorliegt, kann ein "Leader-UAV" laut [2] zu jedem Agenten eine Nachricht senden.

Es gibt einen mathematischen Nachweis, der die Existenz eines Wurzelknotens innerhalb eines Netzwerkes überprüft [2]. Ein Graph enthält genau dann einen "rooted out-branching digraph", wenn der Rang $L(G) = n - 1$ ist. Die Laplace-Matrix $L(G)$ ist immer positiv semidefinit und wird aus der Graphentopologie bestimmt. Auf zusätzliche Starrheit einer solchen Formation wird in [2] nicht eingegangen. Es zeigt sich aber, dass das Vorhandensein eines "rooted out-branching digraph"s kein hinreichendes Kriterium für Starrheit ist.

Für den Graphen, der in [2] diskutiert wird, gibt es nur spezielle Anwendungsformen. Da Kommunikation nicht vorgesehen ist, kann das "Leader-UAV" nur einfache Anweisungen, wie "Kamera einschalten" oder "Luftpartikel analysie-

ren", den UAVs über den "gerichteten Formationsgraphen" senden. Außerdem müssten alle Pfeilrichtungen umgekehrt werden, damit der Formationsgraph auch zum Kommunikationsgraphen wird. Es wird offensichtlich, dass dieses Konzept ein Beispiel dafür ist, dass es nicht für alle Methoden Anwendungsmöglichkeiten für Formationen gibt.

4.5 Vorüberlegungen für die Entwicklung des Kommunikationsgraphen

Damit Nachrichten innerhalb der Formation auf dem kürzesten Wege zugestellt werden können, wird mittels Dijkstra-Algorithmus der kürzeste Weg zu jedem Knoten berechnet und die dafür zu durchlaufenden Zwischenknoten werden gespeichert. Zuerst wird der Fall betrachtet, bei dem der W-LAN-Senderadius unbegrenzt ist. Der Graph, auf dem der Dijkstra-Algorithmus ausgeführt wird, ist der komplette Graph des Formationsgraphen. Die kürzesten Wege werden berechnet. Logischerweise ist der direkte Weg auch der kürzeste Weg. Wenn die W-LAN Reichweite begrenzt ist, müssen die Kanten, die länger sind als der Empfangsradius, wieder entfernt werden. Um den Graphen zu erhalten, der dem kompletten Graphen am nächsten kommt, wird das modifizierte Rendezvous-Problem gelöst. Der Algorithmus ist modifiziert, da der Formationsgraph sich nicht ändern darf, jedoch trotzdem initial eigentlich ein kompletter Kommunikationsgraph erstellt werden sollte. Die Abbruchbedingung für das Rendezvous-Problem ist nun nicht mehr, dass zu jedem anderen Knoten eine Kommunikationsverbindung erreicht wurde, sondern dass ein Timer abgelaufen ist, der gestartet wird, wenn ein UAV gerade eine neue Kommunikationsverbindung herstellen konnte. Wenn jedes UAV von jedem Nachbarn zu dem eine Verbindung besteht das Timeout Signal erhalten hat, ist der initiale Kommunikationsgraph erstellt und der Dijkstra-Algorithmus kann gestartet werden. Genaueres zu Rendezvous-Problemen findet sich im Abschnitt "Wahl eines Leader-UAVs". Abbildung 4.8 zeigt eine Kolonnenformation, in der ein UAV Kontakt zu höchstens 2 von 4 potentiellen Nachbarn aufnehmen kann.

4.6 Konstruktion des Kommunikationsgraphen

Die UAVs haben ihre Position innerhalb der Formation eingenommen. Die benachbarten UAVs werden ausschließlich über Abstandsmesssensoren erkannt. Der Kommunikationsgraph ist noch unbekannt. Jedes UAV sendet eine Broadcast-

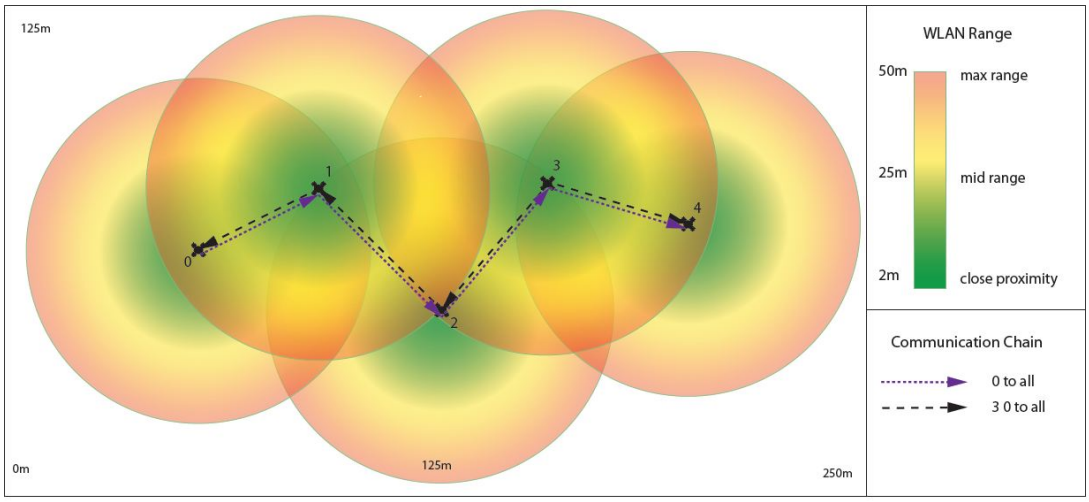


Abbildung 4.8: W-LAN-Reichweite (aus [12])

Nachricht, welche die eigene ID enthält und wartet auf Resonanz. Eine Nachricht, die in einem Rechnernetz gesandt wird, ohne Kenntnis davon, an wen die Nachricht gerichtet ist und wie viele Agenten die Nachricht empfangen werden, wird als Broadcast-Nachricht bezeichnet. Wenn ein UAV i eine Nachricht von einem unbekannten Absender erhält, überprüft es, ob es sich um eine Broadcast- oder eine Unicast-Nachricht handelt. Eine Unicast-Nachricht enthält einen klar definierten Empfänger. Auf eine Broadcast-Nachricht antwortet es mit einer Unicast-Nachricht, die die eigene ID und die ID des initialen Absenders enthält. Wenn der Empfänger einer Antwort UAV i sein soll, vermerkt das UAV i das Vorhandensein eines Nachbarn. Wenn die Nachricht hingegen an ein anderes UAV gerichtet ist, wird diese Nachricht verworfen.

Die Abbildung 4.9 a) zeigt einen Formationsgraphen. Der dazugehörige Kommunikationsgraph ist unbekannt und wird daher von den UAVs ermittelt. Das gelb markierte UAV sendet eine Broadcast-Nachricht. Die Broadcast-Nachricht kann nur von UAVs empfangen werden, die innerhalb des rot markierten Kreises liegen. Alle grün markierten UAVs haben die Nachricht empfangen und antworten, indem sie eine Unicast-Nachricht über eine grün markierte Kante zurücksenden. Dieses Vorgehen wird von jedem UAV ausgeführt. Die Abbildungen 4.9 b) bis e) zeigen die Vorgänge für die UAVs $6, 5, 4$ und 3. Aufgrund der y-Achsensymmetrie der Formation, führt die Betrachtung der UAVs 1 und 2 zu keiner weiteren Hinzunahme von Kanten. Nach Ausführung des Dijkstra-Algorithmus auf dem finalen Graphen, der wie angesprochen dem in Abbildung 4.9 e) bereits entspricht, werden die folgenden Erkenntnisse gewonnen. Die UAVs $5, 1, 4$ und 2 können direkt vom "Leader-UAV" erreicht werden. Agent 3 kann über 2 gleiche lange Distanzen erreicht werden. Als Zwischenknoten hierfür dient entweder Agent 2 oder Agent

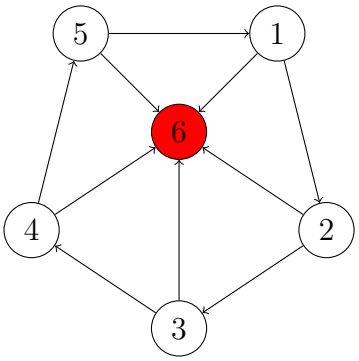
4. Daher kann entweder die Kante e_{43} oder die Kante e_{23} entfernt werden. Die Abbildung 4.9 f) zeigt den konstruierten Kommunikationsgraphen.

Der W-LAN-Kommunikationsradius ist abhängig von der verwendeten Technik an Bord des UAVs. Ein UAV mit einem großen W-LAN-Sendebereich eignet sich gut als "Leader-UAV".

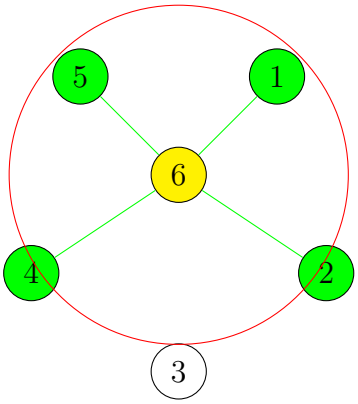
Die Schwierigkeit ist das eindeutige Identifizieren der Position des Senders innerhalb der Formation. Da das UAV nicht erkennen kann, wo die Quelle der Nachricht liegt, muss in den gesendeten Paketen auch enthalten sein, an welcher Position innerhalb der Formation sich der Sender befindet. Dies setzt voraus, dass jedes UAV den Formationsgraphen kennt. Eine Modifikation des Verfahrens nutzt die Möglichkeiten des "Leader-UAV" aus. Das "Leader-UAV" kennt im Gegensatz zur vorangegangenen Annahme nun ausschließlich die Graphentopologie. Der Kommunikationsgraph wird nun genau wie in dem beschriebenen Verfahren bestimmt. Der Unterschied besteht darin, dass das "Leader-UAV" die zuvor auf alle UAVs aufgeteilten Schritte übernimmt und sich somit quasi in die Rolle eines jeden UAVs hineinversetzt und prüft, wen es alles erreichen könnte. Damit nun die UAVs erfahren, welche Nachbarn über W-LAN erreichbar sind, sendet das "Leader-UAV" entweder direkt oder über mehrere Zwischenknoten Nachrichten an die UAVs. Das "Leader-UAV" 6 aus der Abbildung 4.9 f) würde über UAV 4 mit UAV 3 Kontakt aufnehmen.

4.7 In der Anwendung häufig verwendete Formationen

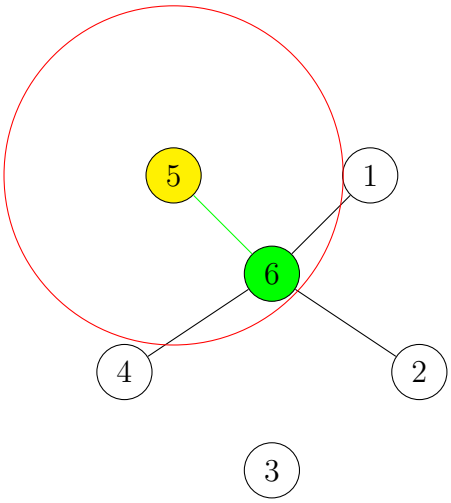
Bei den in Abbildung 4.10 dargestellten Formationsgraphen, deren Kontrollstruktur asymmetrisch ist, handelt es sich um persistente Graphen, da die Graphen starr sind und der Ausgangsgrad von keinem der beteiligten Agenten größer als 2 ist, siehe [14] für weitere Informationen. Innerhalb der wheel-Formation ist Agent 6 das "Leader-UAV", wohingegen der C^2 Graph der Gegenentwurf zu diesem Konzept ist. Alle Agenten sind innerhalb der C^2 Formation gleichwertig, da von jedem UAV Ausgangs- und Eingangsgrad übereinstimmen. Ein möglicher Einsatzbereich hierfür ist die Waldbrandaufklärung, da jedes UAV Bodenmessdaten sammeln kann. Diese können später ausgewertet werden und aufgrund des versetzten Fliegens werden sowohl redundante Daten gesammelt, als auch Daten aus unterschiedlichen Blickwinkeln aufgenommen. Die C^2 Formation entspricht einer doppelten, ineinanderverschränkten V-Formation. Messdaten, aufgenommen in einem Bereich, können zeitlich nach k Zeiteinheiten verglichen werden, da erst Agent 1 die Daten erfasst und kurze Zeit später das UAV 2. Der C^2 Formations-



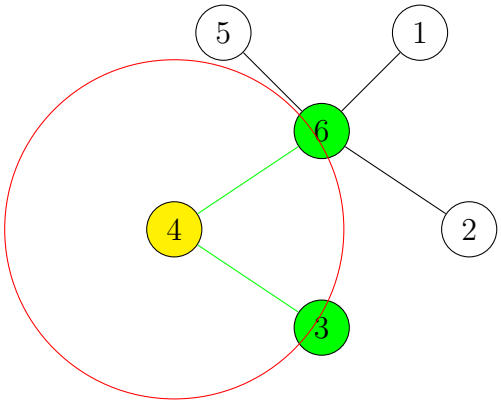
(a) Formationsgraph



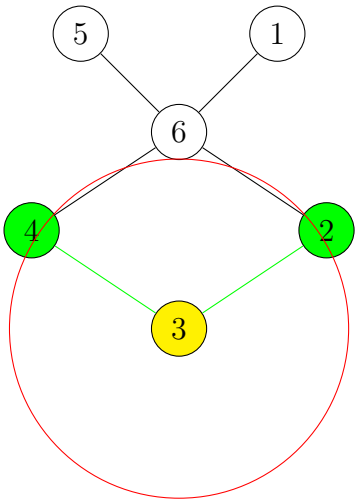
(b) UAV 6 sendet Broadcast-Nachricht



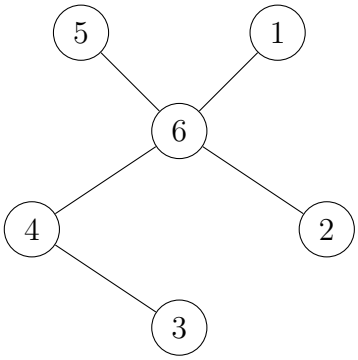
(c) UAV 5 sendet Broadcast-Nachricht



(d) UAV 4 sendet Broadcast-Nachricht

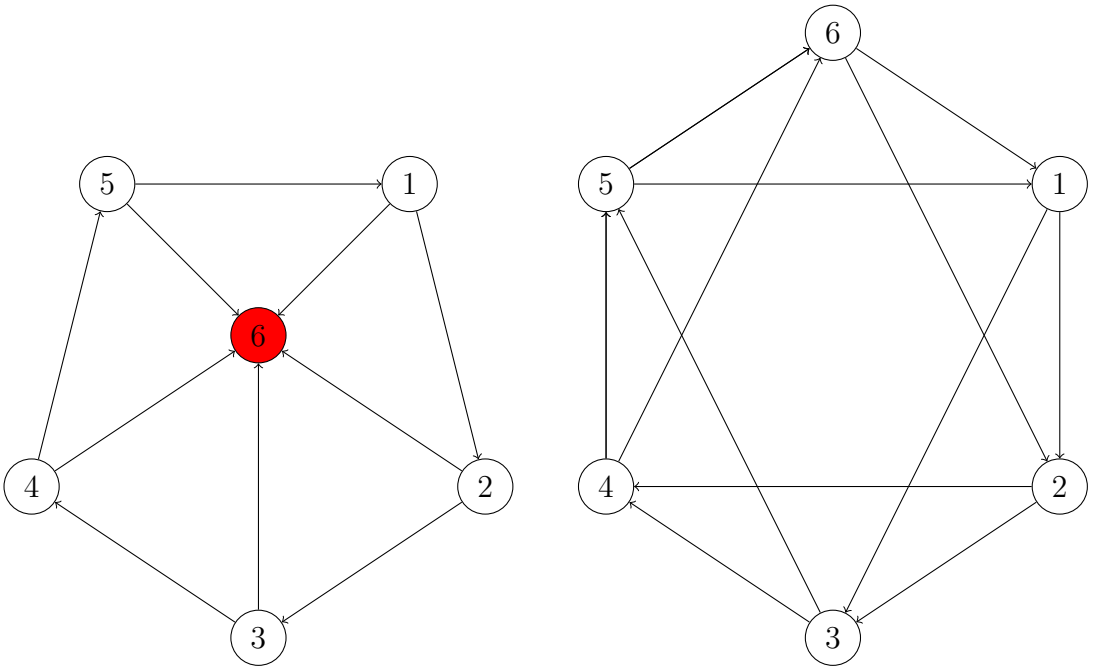


(e) UAV 3 sendet Broadcast-Nachricht



(f) Kommunikationsgraph

Abbildung 4.9: Konstruktion des Komunikationsgraphen



(a) wheel Graph, n Knoten, $2n$-2 Kanten (b) C^2 Graph, n Knoten und $2n$ Kanten

Abbildung 4.10: Beispiele für Formationen

graph erlaubt den Verlust von einer der äußeren Kanten, ohne dass die Persistenz verloren geht, wenn $V = n$ und $E = 2n$ erfüllt ist. Der C^2 Formationsgraph ist außerdem "birigid", wenn die Forderungen an V und E erfüllt sind. Im obigen Abschnitt " Redundante Starrheit" wurde definiert, dass ein starrer Graph "birigid" ist, wenn dieser auch starr bleibt, wenn ein Knoten und alle zu diesem Knoten inzidenten Kanten entfernt werden.

Bei einem wheel Graphen mit einer ungeraden Anzahl an Knoten kommt man mit 3 Farben aus, wohingegen die Knotenfärbungszahl 4 ist, wenn die Anzahl der Knoten gerade ist. Bei der Nutzung der "cyclic stop-and-go strategy" bieten sich daher wheel Graphen mit einer ungeraden Anzahl an Knoten an.

Kapitel 5

Übertragung auf Simulator und Ergebnisse

In diesem abschließenden Kapitel wird erklärt, in welcher Form vorgestellte Konzepte umgesetzt werden können und für welche Alternative sich entschieden werden sollte, wenn es vergleichbare Realisierungsmöglichkeiten gibt. Außerdem werden Features genannt, die die Agenten zusätzlich beherrschen müssen, damit sie in der Lage sind, die von ihnen geforderten formationsspezifischen Aufgaben auch auszuführen.

Bei der Ausführung eines Einsatzes für die Waldbrandaufklärung sollten die UAVs in verhältnismäßig kleinen Formationsgruppen agieren, da der Einsatzbereich nicht ungefährlich für die UAVs ist. Stichflammen können einzelne UAVs beschädigen, wenn diese die Gefahr nicht rechtzeitig erkennen und die Flughöhe anpassen. Wenn mehrere Formationen zeitgleich versuchen, in zuvor aufgeteilten benachbarten Bereichen, potentielles Waldbrandgebiet ausfindig zu machen, ist der Grad an Datenredundanz auch nicht allzu groß. Das Aufklärungsgebiet über ein verteiltes System aus unterschiedlichen Blickwinkeln zu erfassen hat nicht nur Vorteile, da auch viele überflüssige Daten gesammelt werden.

Der Algorithmus "Cyclic stop-and-go strategy" sollte im Simulator unter Nutzung der zuerst aufgestellten Regeln verwendet werden. Die Verbesserung des Algorithmus ist überzeugend, jedoch für den Anwendungsbereich ungeeignet, da für die Waldbrandaufklärung nicht allzu viele UAVs sich zeitgleich in einer Formation befinden. Die Methode, die UAVs mittels Knotenfärbung Teilmengen zuordnet, ist hingegen einfach zu implementieren und liefert zwar keine ideale, aber dennoch eine gute Lösung.

5.1 AURIS

Im folgenden wird ein Einblick in die unterschiedlichen Funktionsbereiche des Simulators gegeben, mit dem Ziel einen Eindruck von dessen bisheriger Leistungsfähigkeit zu bieten. Es werden Aspekte angesprochen, welche die Kollisionskontrolle betreffen, aber auch Verhaltensformen werden erwähnt. Es werden ebenfalls Modifikations- und Erweiterungsmöglichkeiten aufgezeigt.

5.1.1 Umsetzung der UAV-Kollisionskontrolle im Simulator

Aus den Sensordaten ermittelt das UAV, ob auf der Flugbahn sich ein Hindernis befindet. Die Simulator-Sensorreichweite $MaxRange$ gibt an, wieweit ein optoelektronischer oder akustischer Sensor im Idealfall sehen kann. In der Anwendung wird wahrscheinlich höchstens ein Sensor für ein UAV zur Verfügung stehen. Dennoch wird im Simulator bislang davon ausgegangen, dass mehrere Sensoren, die im gleichen Abstand auf einem Kreis angeordent sind, zur Verfügung stehen. Es ist vorstellbar, dass ein Sensor genutzt wird, der sich um 360° drehen kann und somit in unterschiedlichen Blickrichtungen Werte aufnehmen kann.

In eine Datenstruktur, ein Array, werden die aufgenommenen Daten geschrieben.

Messdatenarray: [4 3 2 3 4 4] , mit $n = 6$ Sensoren, $MaxRange = 4$

Das UAV erkennt, dass vor ihm ein Hindernis aufgetaucht ist. Wenn mehrere Hindernisse vor dem UAV lokalisiert werden, führt eine Gewichtung [12] der Messdaten zu der Erkenntnis, aus welcher Richtung das UAV zuerst auf ein Hindernis treffen wird. Das UAV wird der unmittelbarsten Gefahr zuerst ausweichen. Wenn beispielsweise die Sensoren auf der rechten Seite ein Hindernis ausmachen, wird sich das UAV nach links hin ausweichen.

Die Gewichtung der Daten wird genutzt, um den notwendigen Drehwinkel zu ermitteln. Die im Simulator implementierten UAVs erkennen noch nicht den Fall, bei dem es möglich ist, zwischen einem Hindernis hindurchzufliegen oder dies ermöglicht wird, nachdem die Größe der Formation skaliert wurde. Das UAV müsste daher die Messdaten mit Referenzdatensätzen vergleichen. Ein Vergleich der Summe der Einzelgewichte einer Seite mit der anderen auf Symmetrie ist schwierig, da die den Sensordaten zugewiesenen Einzelgewichte aufsummiert werden.

Wenn beispielsweise diese Messdaten auszuwerten sind

Messdatenarray: [4 3 2 4 2 3 4 4] , mit $n = 8$ Sensoren

müssten Rückschlüsse auf ein zu durchfliegendes Tor ermöglicht werden. Im Moment würde das UAV auf der linken Seite drum herum fliegen.

Erfolgversprechender könnte der Ansatz sein, die Sensormessdaten zu nutzen, um hindernisfreie Flächen zu erkennen. Geometrische Flächen, wie Dreiecke, die hindernisfreies Gebiet darstellen, lassen sich auf der Karte des UAVs einzeichnen.

5.1.2 Variable Geschwindigkeit

Die im Simulator implementierten UAVs bewegen sich alle mit der gleichen Geschwindigkeit. Eine Verbesserung dahingehend, dass sich die Geschwindigkeit an die gerade ausgeführte Aktion anpasst ist wünschenswert. Angenommen ein UAV dreht sich um 70°, um daraufhin geradeaus zu einem neuen Ziel zu fliegen. Es sollte sich hierbei schneller drehen, als wenn es sich nur um 5° drehen müsste. Hier würde eine hohe Drehgeschwindigkeit unnötig sein, da die dafür benötigte Leistung sich nicht auszahlen würde. Die Batterie sollte, wenn möglich, nicht allzu stark belastet werden. Implementiert ist hingegen bereits, dass das UAV sich maximal um einen Winkel von π dreht. Würde es sich beispielsweise um 190° im Uhrzeigersinn drehen wollen, wird es sich stattdessen um 170° gegen den Uhrzeigersinn drehen.

Die Überlegung der Geschwindigkeitsanpassung spielt auch eine Rolle, wenn nach Ausführung der Ungarischen Methode die UAVs sich zu ihren ermittelten Zielkoordinaten bewegen. Ein UAV, das eine kürzere Strecke zurücklegen muss als andere, kann sich also Zeit lassen. Denn die Zeit, die es sonst auf die anderen Gruppenmitglieder warten müsste, kann auch für einen Treibstoffsparenden Anflug genutzt werden. In [17] wird ein anderer Ansatz verfolgt. Der strategische Hintergrund ist dort, dass alle UAVs zeitgleich in einem Bereich ankommen sollen, um den Überraschungsmoment auszunutzen.

5.1.3 Informationenübergabe an UAVs mittels ID

Die Startposition aller UAVs ist momentan die gleiche. Es kommt dennoch zu keiner Kollision, da die UAVs zeitlich nacheinander gestartet werden und sich immer sofort zu ihren Zielkoordinaten bewegen. Dies bietet Spielraum für Verbesserungen. Im Simulator erhalten die UAVs ihre Spawnposition, indem sie aus einer

für alle zugänglichen Konfigurationsdatei diese herauslesen. Die ID ermöglicht es, dass jedes UAV eine andere Startposition erhält. Für das Hinzufügen eines neuen UAVs wird das Kontrollprogramm AURIS ein weiteres Mal gestartet und das UAV erhält eine fortlaufende ID. Die UnrealEngine erkennt, wieviele UAVs sich im Simulator derzeit befinden.

5.2 Möglichkeiten an Kontrollstrukturen

Die "Follower-UAVs" nutzen die asymmetrische Kontrollstruktur, um die eigene Geschwindigkeit, geschlussfolgert aus der Geschwindigkeit des "Leader-UAVs" zu bestimmen und sich an dessen Bewegungsrichtung zu orientieren. Für die Korrektur und Aufrechterhaltung der Abstandsbedingungen kann entweder eine symmetrische oder eine asymmetrische Steuerung genutzt werden. Die Wahl hierfür ist abhängig von der Zuverlässigkeit der Kommunikationswege und von der geforderten Ausführungsgeschwindigkeit der Algorithmen. Im Bereich des Möglichen ist auch eine hybride Kontrollstruktur. Nur die Kanten, die inzident zu dem "Leader-UAV" sind, sollten dann gerichtet sein. Die restlichen Kanten sind ungerichtet. Die Vorteile von beiden Steuerungsmöglichkeiten sollen ausgenutzt werden. Der Nachrichtenaustausch mit dem "Leader-UAV" geschieht immer über eine symmetrische Kontrollstruktur.

5.3 Nutzen der minimalen Starrheit

Wenn ein UAV verloren geht oder umkehrt, um die Batterie aufladen zu lassen, prüft das "Leader-UAV", ob mit der Reduktion der Anzahl an Knoten auch die Starrheit verloren geht. Wenn die Formation minmal starr gewesen ist, trifft dieser Verlust fast immer ein. Zu starren Formationen, die einen Verlust eines UAVs problemlos kompensieren, zählt der diskutierte C^2 Graph. Minimal starre Formationen sind im Hinblick auf die Anzahl benötigter Kommunikationswege und benötigtem Rechenaufwand sehr effizient. Sehr wenige Informationen müssen daher in der Datenbank hinterlegt sein. Jedoch ist die Formation nicht sehr robust [1], da Ausfälle von UAVs zu einem Verlust der Starrheit und von benötigten Kontrollwegen führen können. Die neue Formation ist nur starr, wenn zufällig ein starrer Graph mit $n-1$ Knoten entstanden ist. Die Korrektur der Formation aufgrund des Ausscheidens eines Agenten wird in [5] als „closing ranks“, "die Reihen schließen", bezeichnet. Es kommt auch zu Problemen, wenn ein Sensor ausfällt oder defekte Messwerte aufzeichnet. Es werden daher in Abhängigkeit von Einsatzbereich und Ausfallwahrscheinlichkeit teilweise redundante Wege von

vornherein bereit gestellt. Die Agenten bilden Kanten neu aus, wenn erkannt wurde, dass über einen Weg fehlerhafte Daten gesendet werden oder ein Agent sich nicht mehr meldet.

Eine nach dem Vorbild "Vogelzug" entwickelte und einfach zu steuernde Formation ist die Dreiecksformation. Sowohl im $\mathbb{R}^2$ als auch im $\mathbb{R}^3$ ist die Dreiecksformation minimal starr.

5.4 Realisierbare und nutzenmaximierende Formationen

In der internen UAV-Datenbank sind ausschließlich Formationen gespeichert, die zulässig sind. Im Englischen werden solche Formationen als "feasible" bezeichnet. Dies bedeutet, dass die geforderten Abstandsbedingungen widerspruchsfrei sind. Die Formation ist widerspruchsfrei, wenn der Wert der Lyapunov Funktion gegen 0 gehen kann. Die lernfähigen Agenten sollten in der Lage sein, sich eigenständig Formationen zu überlegen, wenn sie erkennen, dass sie aus den bekannten nicht den größten Nutzen für die derzeit aktiven Ziele ziehen können. Die Agenten erkennen ein unzufrieden stellendes Resultat nach Durchführung der ungarischen Methode. Die Agenten versuchen unter einer Obergrenze von einer maximal erlaubten Kostensumme zu bleiben. Es sind maximal 3 Versuche zulässig, ansonsten wird sich für die beste Formations-Alternative der 3 Versuche entschieden. Mittels Henneberg Operationen lassen sich zu der geplanten Formation Alternativen erstellen, indem die k Knoten mittels inversen Operationen entfernt werden, die schlechte Werte verursacht haben. Es werden k neue Knoten unter Verwendung der Henneberg Operationen hinzugenommen. Entscheidungen hierüber trifft ebenfalls das "Leader-UAV". Die Effizienzsteigerung wird vor dem Hintergrund gesehen, dass die UAVs sich langsamer bewegen, als die Rechenzeit dauert. Daher lohnt sich dieses beschriebene Suchen nach Verbesserungen. Diese Verbesserung lässt sich nicht nutzen, wenn Forderungen bezüglich des Aussehens der Formation für den Anwendungszweck gestellt werden.

5.5 UAV-Design

Um zu verdeutlichen, wie die Architektur und Arbeitsweise der UAVs im Hinblick auf die Implementierung im Simulator aussehen könnte, werden im folgenden Ideen für 2 Steuerungs-Hierarchien vorgestellt. Die 2. Steuerungshierarchie, entnom-

men [20], ist eigentlich auf keine Formation ausgelegt, sondern für ein einzelnes UAV, das eine Aufgabe ausführen soll, modelliert. Daher wird abschließend noch auf die notwendige Anpassung für die Nutzung von Formationen eingegangen.

5.5.1 Steuerungs-Hierarchie für Formationen

Die Funktionen der UAVs sind in einem Schichtenmodell strukturiert [18]. In Softwaresystemen wird als Architektur häufig ein Schichtenmodell genutzt. Das OSI-Schichtenmodell wird in verteilten Systemen von Kommunikationsprotokollen, wie TCP oder UDP, genutzt. Ein Schichtenmodell fördert die Übersichtlichkeit und ermöglicht ein einfaches Austauschen von bestimmten Schichten. Die UAV-Steuerung beruht auf den folgenden Überlegungen. Die oberen Schichten erhalten Befehle, sowie konkrete und eindeutige Signale. Die unteren Schichten hingegen empfangen kontinuierlich und sich ständig ändernde Anweisungen. Man bezeichnet dies als hybride Arbeitsweise. Jede Schicht ist selbstständig und unabhängig in ihrer Arbeitsweise. Kontrollanweisungen können nur an die nächsttiefere Schicht weitergereicht werden, wohingegen Informationen auch nach oben gegeben werden dürfen. In den oberen Schichten werden globale Informationen verarbeitet, wohingegen in den unteren Schichten schnell arbeitende Algorithmen lokal Entscheidungen treffen und Werte bestimmen. Das Ziel ist es, die Kohäsion des Moduls so weit wie möglich zu stärken. Das Schichtenmodell setzt auf sequentielle Kohäsion.

Das Schichtenmodell, welches das Verhalten und Agieren eines UAVs beschreibt, ist in Abbildung 5.1 dargestellt.

In der obersten Schicht, "Mission Control Interface", sind die Eingabeparamter der UAV-Status und die über Sensoren ermittelten Daten von der Umwelt. Abhängig von den Zielen werden Flugmanöver ausgearbeitet. In der obersten Schicht laufen ebenfalls Programme ab, welche die kooperative Arbeitsweise mehrerer Formationen koordinieren. Hierfür kommunizieren die Formationen miteinander über Funk.

Die zweite Schicht, "Structural Commands", trifft Entscheidungen, die das Aufteilen von Formationen und das Vereinigen von Teilformationen umfassen. Die Befehle zur Durchführung werden an die darunterliegende Schicht weitergegeben.

Innerhalb der "Formation Management"-Schicht werden Voraussetzungen für die Durchführung der Algorithmen geschaffen. Der Bewegungs-Ablaufplan wird für die "Cyclic stop-and-go strategy" bestimmt. Das Ziel ist es, dass die Formation gehalten wird. Für UAVs, die ihren Formationsplatz verlassen sollen, um zu einem

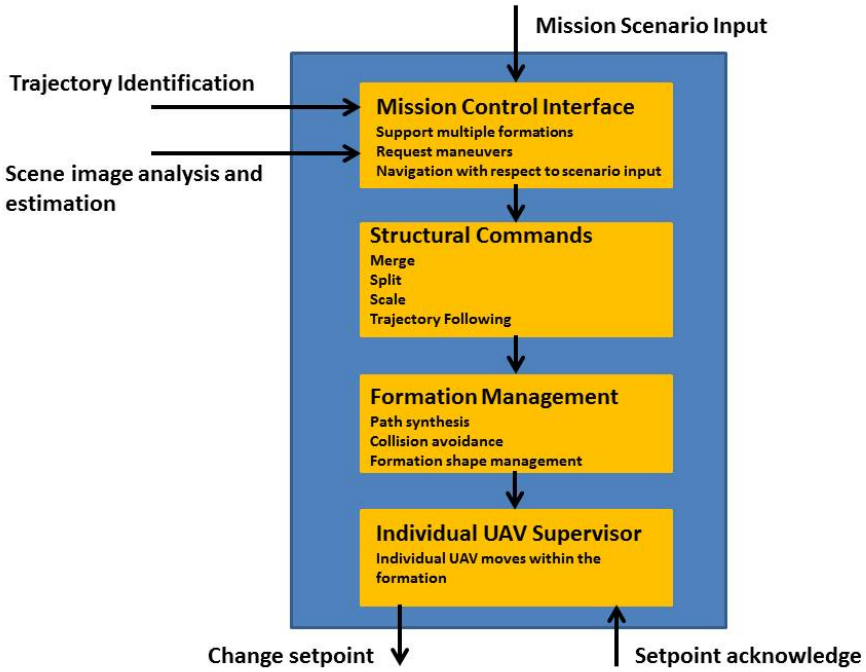


Abbildung 5.1: Steuerungs-Hierarchie, die in erster Linie die Formation überwacht (aus [18])

Neuen zu wechseln, wird die optimale und kollisionsfreie Flugbahn bestimmt. Wenn das "Leader-UAV" nicht länger verfügbar ist, wird innerhalb dieser Schicht der Algorithmus zur Bestimmung des geeignetsten Nachfolgers ausgeführt.

Die unterste Schicht "Individual UAV Supervisor" bestimmt Anweisungen, die nur für jeweils ein UAV gelten. Innerhalb dieser Schicht sind beispielsweise die durchzuführenden Bewegungen bekannt, die zu einer Verbesserung des Lyapunov-Funktionswertes führen sollen. Anweisungen werden an die "Regulation Schicht" weitergegeben, damit in dieser die Flugbahn verfolgt werden kann.

5.5.2 Steuerungs-Hierarchie für die Flugbahnermittlung

Die in [20] vorgestellte Steuerungs-Hierarchie umfasst, wie in der Abbildung 5.2 erkennbar ist, schwerpunktsmäßig die Ermittlung der Flugbahn mittels Interpretation der Missionsziele. Allgemein unterscheidet sich diese Architektur, beruhend auf dem Schichtenmodell, wenig von der in [18] vorgestellten. Die Grundidee für die Architektur ist die gleiche und auch bezüglich der Vorteile gibt es keine Un-

terschiede.

In der obersten Schicht wird das an das UAV übergebene Missionziel identifiziert. Alle zur Erfüllung notwendigen Teilschritte werden ermittelt und in einer geeigneten Datenstruktur abgelegt. Eine nach dem First In - First Out-Prinzip arbeitende Warteschlange ist am geeignetsten. Wenn ein UAV bespielsweise die optimale Position innerhalb einer Formation bestimmen möchte, nutzt es die Ungarische Methode. Diese ist wie in Abschnitt 3.1 beschrieben nur eine Teilaufgabe des Prozesses, der vom Identifizieren der optimalen Position bis zum tatsächlich dorthin fliegen reicht. Sämtliche Prozessschritte können ermittelt werden, bevor die erste "Task Info" an die darunterliegende Schicht weitergegeben wird. Es ist jedoch auch möglich, Teilaufgaben bereits ausführen zu lassen, obwohl noch nicht alle Teilaufgaben ermittelt wurden. Dies spart Zeit und beansprucht den Prozessor wenig. Der Ablaufplan für das Erreichen des Ziels muss dynamisch geändert werden, wenn zufällige Ereignisse eintreten, wie das Auftauchen von Hindernissen oder äußere Umwelteinflüsse, die die Ausführung der Aufgaben erschweren. Eine mögliche Änderung der Teilaufgabenausführung muss nicht berücksichtigt werden, wenn Informationen über die Umgebung bereits durch Aufklärungsflüge beschafft wurden oder es sich bei der Aufgabe um eine sich zyklisch wiederholende handelt, wie es beim mehrfachen Transportieren von Gütern zu einem Zielpunkt hin der Fall ist.

Die "Task Info", die an die darunterliegende Schicht gesandt wird, enthält Angaben über die folgenden 6 Punkte.

1. Zeit, die für das Erfüllen der Aufgabe zur Verfügung steht
2. Zielposition im Raum
3. geforderte Ausrichtung des UAVs, die berücksichtigt wird, wenn UAV Zielposition erreicht hat
4. Information, ob es sich bei Zielangabe um relative oder absolute Werte handelt
5. zulässige Höchstgeschwindigkeit
6. zulässige stärkste Beschleunigung

Die Zeit spielt beispielsweise bei der Ausführung des Positionsfindungs-Prozesses eine wichtige Rolle, da in der "Mid Level"-Schicht abhängig von der zur Verfügung stehenden Zeit die UAV-Geschwindigkeit optimal bestimmt werden kann. Absolute, GPS gestützte Zielkoordinaten werden übergeben, wenn es um die Zielposition im Raum geht und relative Vektorangaben, wenn beispielsweise die UAVs den Abstand zueinander skalieren sollen.

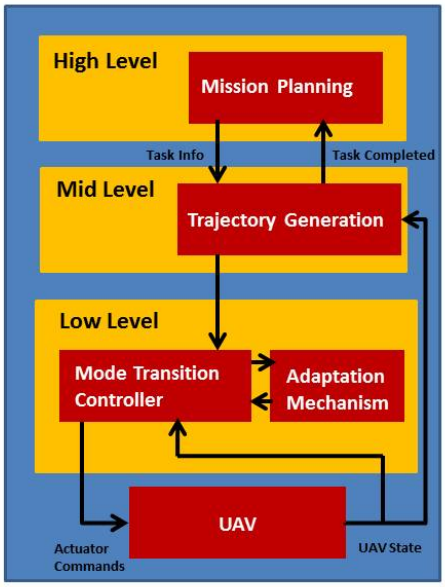


Abbildung 5.2: Steuerungs-Hierarchie mit dem Schwerpunkt Arbeitsabläufe zu ermitteln (aus [20])

Bei der Suche nach Waldbränden ist eine höhere Maximalgeschwindigkeit sinnvoll, da diese Aufgabe zeitkritisch ist, wohingegen bei der Aufnahme von Kamerabildern eine langsamere Fluggeschwindigkeit zu einer Verbesserung der Aufnahmen führt. Da jede Aufgabe in zahlreiche Teilaufgaben unterteilt ist, kann die Fluggeschwindigkeit zum Startpunkt der Aufnahmen um einiges höher sein.

Wenn eine Teilaufgabe erfüllt wurde, sendet die "Mid Level" Schicht eine "Task Completed" Nachricht an die oberste Schicht, die daraufhin eine weitere Teilaufgabe an die "Trajectory Generation"-Komponente in der "Mid Level"-Schicht senden kann oder auf weitere Rückmeldungen wartet.

Die "Trajectory Generation"-Komponente der "Mid Level"-Schicht errechnet Vorgaben, wie Wegpunkte der Flugbahn und Flughöhe für die "Low-level controllers", damit das UAV zielführend gesteuert wird, siehe auch Abschnitt "Modellierung der Flugbahn".

Die unterste Schicht, "Low Level", erhält die Wegpunkte. Innerhalb dieser Schicht wird das UAV stabilisiert, die Höhe kontrolliert und die Rotoren werden gesteuert. Eine ausführliche Beschreibung der Steuerung eines UAVs ist in [12] zu finden.

Das UAV bewegt sich schließlich entlang der Wegpunkte zu dem Zielpunkt.

5.5.3 Erweiterung der Steuerungshierarchieen

In [20] wurde davon ausgegangen, dass ein UAV alle im Schichtenmodell enthaltenen Schichten selbst ausfüllt und in [18] fehlt die Ausführung, wie das vorgeschlagene Schichtenmodell für den Anwendungsfall zu nutzen ist. Da innerhalb von Formationen in der Regel mehrere UAVs die gleichen Aufgaben ausführen, kann die Planungsphase auch einem UAV übertragen werden, nämlich dem "Leader-UAV". Bei dem Modell 1 sollte das "Leader-UAV" über die Schichten "Mission Control Interface" ,"Structural Commands" und "Formation Management" verfügen. Beim 2. Modell sollte das "Leader-UAV" über die Schichten "Mission Planning" und "Trajectory Generation" verfügen. Die Gründe dafür, dass so viele Schichten an das "Leader-UAV" abgetreten werden können, sind vielfältig. Das "Leader-UAV" muss nur für sich selbst die Flugbahn bestimmen, damit alle anderen UAVs innerhalb der starren Formation der Flugbahn ebenfalls folgen. Da das "Leader-UAV" den Überblick über die Topologie hat, kann es Entscheidungen treffen, die die Struktur der Formation betrefffen.

In Modell 1 dienen die Daten, die ein UAV aufnimmt, wenn es ein Hindernis erkennt, als Übergabeparameter an die oberste Schicht. Die Nachrichten werden über die Kommunikationswege gesandt. Das Schichtenmodell wird jedoch ebenfalls komplett in jedem UAV implementiert, denn beispielsweise bei der Ausführung der Ungarischen Methode ist das UAV noch nicht Teil einer Formation und muss somit alle Aufgaben durchführen können. Mit dem Ziel, das "Leader-UAV" zu entlasten, werden Berechnungen, die UAV-interne Daten, wie den Batteriestatus benötigen, von den UAVs selbst durchgeführt.

Für die Umsetzung im STS-Simulator ist eine Mischung aus beiden Steuerungshierarchien effektiv, da sich deren Aufbau nicht gegenseitig ausschließt, sondern vielmehr ergänzt.

5.6 Ausblick

In dieser Arbeit wurde erläutert, wie Formationen beschrieben werden können und welche Voraussetzungen für deren Konstruktion erfüllt sein müssen. Außerdem wurden Vorschläge gemacht, wie der Informationsaustausch zwischen den Agenten innerhalb der Formation stattfinden sollte und welche Protokolle zu nutzen sind. Die problemlose Skalierbarkeit des Systems wurde verdeutlicht. Of-

fensichtlich ist, dass große Unterschiede zwischen Theorie und Anwendung bestehen.

Der Anwendungsbereich von Formationen kann noch weiter ausgebaut werden. Formationen können auch genutzt werden, um Güter zu transportieren, die ein einzelnes UAV nicht alleine tragen könnte. Bei der Formationskonstruktion werden Form, Gewicht und Schwerpunkt der Last berücksichtigt. Die Agenten sind einfach anpassbar und erweiterbar für neue mechanische, sowie elektronische Geräte und Sensoren. Für die im STS-Simulator verwendeten UAVs ist mit dieser Arbeit eine Grundlage für die Implementierung von Formationen geschaffen. Im Simulationsprogramm wird sich zeigen, wie effizient in der Anwendung die Überlegungen und Konzepte sind. Die Absprache zwischen Formationen für das koordinierte Vorgehen ist ein wichtiger Teilaspekt, der weiterverfolgt werden könnte. Eine Gruppe UAVs warnt andere Formationen vor Gefahren, oder fordert diese zur Unterstützung an.

Literaturverzeichnis

[1] Brian D.O. Anderson, Baris Fidan, Changbin Yu, and Dirk van der Walle: *UAV Formation Control: Theory and Application*

[2] Mehran Mesbahi and Magnus Egerstedt: *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press 2010.

[3] YangQuan Chen and Zhongmin Wang: *Formation Control: A Review and A New Consideration*. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005

[4] Ming Cao, Changbin Yu, Brian D.O. Anderson: *Formation control using range-only measurements*. In: *Automatica 47*. 2011, 776-781

[5] Brian D. O. Anderson, Changbin Yu, Baris Fidan, Julien M. Hendrickx: *Rigid Graph Control Architectures for Autonomous Formations*. In: *IEEE Control Systems Magazine*. December 2008, 48-59

[6] Hua Wang, Yi Guo, and Zhaoyang Dong: *Graph Rigidity Control of Mobile Robot Networks*. In: *8th IEEE International Conference on Control and Automation*. Xiamen, China June 2010, 2218-2220

[7] Reza Olfati-Saber and Richard M.Murray: *Graph Rigidity and Distributed Formation Stabilization of Multi-Vehicle Systems*. In: *Proceedings of the 41st IEEE Conference on Dicision and Control*. Las Vegas, Nevada USA December 2002, 2965-2971

[8] Marko Walther: *Henneberg-Aufbau in O^n Schritten*. FU-Berlin

[9] Jie Gao and Amitabh Basu: *cse590 Rigidity Theory*. In: *Information Processing in Sensor Networks* . Lecture 3

[10] Andis Mehlmann: *Die ungarische Methode*. `http://www.eos.tuwien.ac.at/OR/Mehlmann/Andis/publ/methoden/mopvorl4.pdf`

[11] A. Nixon: *Combinatorial Rigidity and graph constructions*

[12] Gerry Siegemund: *AURIS Autonomous Robot Interaction Simulation*. Hamburg 2011

[13] Bernd Schulze: *Symmetric Laman theorems for the groups C_2 and C_S*. In: *The electronic journal of combinatorics*. Berlin June 2010

[14] Julien M. Hendrickx, Brian D.O. Anderson and Vincent D. Blondel: *Rigidity and Persistence of Directed Graphs*

[15] X.C. Ding, M. Powers, M. Egerstedt, and R. Young: *An Optimal Timing Approach to Controlling Multiple UAVs*

[16] Guoqiang Mao, Sam Drake, and Brian D. O. Anderson: *Design of a Extended Kalman Filter for UAV Localization*

[17] Timothy W. McLain, Phillip R. Chandler, Steven Rasmussen and Meir Pachter: *Cooperative Control of UAV Rendezvous*. In: *Proceedings of the American Control Conference*. Arlington, VA June 2001, 2309-2310

[18] Raman K. Mehra, Jovan D. Boskovic, Sai-Ming Li: *Autonomous Formation Flying of Multiple UCAVs under Communication Failure*. 2000, 371-374

[19] Changbin Yu, Brian D.O. Anderson: *Agent and Link Redundancy for Autonomous Formations*. In: *Proceedings of the 17th World Congress "The International Federation of Automatic Control"*. Seoul, Korea, July 2008, 6584-6589

[20] Luis B. Gutiérrez, George Vachtsevanos, and Bonnie Heck: *A Hierarchical/Intelligent Control Architecture for Unmanned Aerial Vehicles*

[21] Zhao Weihua and Tiauw Hiong Go: *Robust Decentralized Formation Flight Control*. In: *International Journal of Aerospace Engineering*. `http://www.hindawi.com/journals/ijae/2011/157590/` , 2011

[22] Stuart Russell and Peter Norvig: *Artificial Intelligence: A Modern Approach*. Prentice Hall, December 2002

[23] Fachgebiet Nachrichtentechnische Systeme der Universität Duisburg-Essen. `http://nts.uni-duisburg-essen.de/research/research_be/iWBB.shtml` Stand 8.1.2009

[24] MikroKopter - Überblick. http://www.mikrokopter.de/de/Kopter.php

[25] TUHH Quadrokopter Projekt. http://quadro.fst.tu-harburg.de/index.php/quadrocopter

[26] M.F. Thorpe and P.M. Duxbury: *Rigidity Theory and Applications*. Kluwer Academic Publishers 2002.

[27] Färben der Knoten von Graphen - Speziell: 4-Farbenproblem. http://fuzzy.cs.uni-magdeburg.de

Abbildungsverzeichnis