TUHH
Technische Universität Hamburg-Harburg

Institut für Regelungstechnik

Diplomarbeit

Hannes Rose

Design and Implementation of a Simulation Environment for Multi-Agent System Formation Control

Advisor: MSc. A. Popov
1. Examiner: Prof. Dr. U. Weltin
2. Examiner: Prof. Dr. H. Werner

December 2009 Report 2009.24

Hereby I declare that I produced the present work myself only with the help of the indicated aids and sources.

Hiermit erkläre ich, die vorliegende Arbeit selbstständig durchgeführt und keine weiteren Hilfsmittel und Quellen als die angegebenen genutzt zu haben.

Hamburg, December 18, 2009 Hannes Rose

Abstract

Autonomous vehicles like unmanned aerial vehicles, mobile robots, or satellites have various applications today. Formations of these autonomous agents have shown superior behavior in cooperative work over single, complex systems. However, coordination of such formations requires elaborate control strategies and poses new challenges to controller design. Simulations provide a low-cost method to test those controllers and to gain insight into several fields of formation control.

This project proposes a simulation environment for formation control with an object-oriented software architecture. The simulator is developed in `Matlab` and supports linear and non-linear two- or three-dimensional systems. Inter-agent communication is defined using a graph-theoretic approach and the behavior of formations of agents with equal or different dynamics and controllers can be simulated and analyzed. Formation properties, like the number of agents, communication topology, external reference input, and formation shape can be changed dynamically during a simulation run. Influences of communication disturbances on the formation performance due to obstacles and a limited transmission range, as well as transmission time delays, can be analyzed. Simulation results are visualized in 2D or 3D and stored for further analysis.

Scenarios for formations of quadrocopters are designed and used to validate the simulation environment and to analyze formations. Influences of controller design, communication topology, formation reference input, and communication time delays on formation stability and performance are investigated. Additionally, two different strategies to define formations are compared. To analyze performance, several measures are introduced and evaluated with respect to their significance.

Contents

List of Figures

List of Abbreviations and Symbols

Abbreviations

AHS	Automated Highway System
AUV	Autonomous Underwater Vehicle
DoF	Degree of Freedom
GUI	Graphical User Interface
MAS	Multi-Agent System
OOP	Object-Oriented Programming
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language

Symbols

$\otimes$	Kronecker product

Matrices

$\boldsymbol{A}$	Adjacency matrix
$\boldsymbol{A}_K$, $\boldsymbol{B}_K$, $\boldsymbol{C}_K$, $\boldsymbol{D}_K$	System matrices of the controller
$\boldsymbol{A}_P$, $\boldsymbol{B}_P$, $\boldsymbol{C}_P$, $\boldsymbol{D}_P$	System matrices of the plant
$\boldsymbol{\Omega}_P$, $\boldsymbol{\Gamma}_P$	Discrete-time system matrices of the plant
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{L}$	Laplacian matrix

Vectors

$\boldsymbol{c}_{2D}$	Two-dimensional coordinate vector
$\boldsymbol{c}_{3D}$	Three-dimensional coordinate vector
$\boldsymbol{e}$	Formation error for the whole formation
$\boldsymbol{e}_i$	Formation error for one agent, controller input vector
$\boldsymbol{\gamma}_i$	Plant output vector for internal feedback of an agent
$\boldsymbol{r}$	External reference input vector for the whole formation
$\boldsymbol{r}_{ij}$	Reference vector between agents i and j
$\boldsymbol{u_i}$	Controller output vector/plant input vector for one agent
$\boldsymbol{v}_i$	Controller state vector for one agent
$\boldsymbol{x}_i$	Plant state vector for one agent
$\boldsymbol{y}$	Agents output vector for the whole formation
$\boldsymbol{y}_i$	Agent output vector for one agent, transmitted to the other agents
$\boldsymbol{y}_{i,ref,k}$, $\boldsymbol{y}_{j,ref,k}$	Agent reference output vector at timestep k
$\boldsymbol{y}_{i,k}$, $\boldsymbol{y}_{j,k}$	Agent output vector at timestep k

Scalars

α_a	Critical-agents ratio
α_l	Critical-links ratio
$\mathcal{A}$	Set of arcs of a graph $\mathcal{G}$
$d_{\mathcal{G}}(v)$	Degree of a vertex v
$d_{\mathcal{G},in}(v)$, $d_{\mathcal{G},out}(v)$	In-/out-degree of a vertex v
d_{ij}	Reference distance between agents i and j
$\overline{\epsilon}_{f1}$	Formation Error 1
$\overline{\epsilon}_{f2}$	Formation Error 2
$\overline{\epsilon}_{t1}$	Tracking Error 1
$\overline{\epsilon}_{t2}$	Tracking Error 2
$\mathcal{G}$	Graph
$\mathcal{J}_i$	Set of vehicles in a formation that agent i can receive from
N_a	Number of agents in a formation
N_k	Number of timesteps of a simulation
N_l	Number of communication links in a formation

$\mathcal{V}$	Set of vertices of a graph $\mathcal{G}$
v_i	Vertices of a graph $\mathcal{G}$
x, y, z	Spatial coordinates
θ, η, ϕ	Rotational coordinates

Indices

i, j	Agents
k	Timestep

Chapter 1

Introduction

Mobile autonomous agents have been a field of extensive research during the last decades. Technological advances in this area have increased the interest in formations of vehicles in multi-agent systems (MAS). Those vehicles can be all kinds of mobile, independent agents that autonomously interact with the environment and other agents. A few applications of formations are given in the following section. Section 1.2 introduces the control challenges posed by MAS and Section1.3 summarizes available simulation tools. An outline of the thesis is given in Section 1.4.

1.1 Agent Formations

A formation can be any constellation of agents, consisting of at least two, and up to a large number of vehicles. Vehicles in the sense of this thesis include all kinds of autonomous systems, in the air, under water, or on ground. Formations with very large numbers of members are also named *flocks* or *swarms* in literature [OS06]. A formation consists of a number of autonomous agents with same dynamical behavior, which have the ability to communicate with each other. They are dynamically decoupled, that is, the motion of one agent does not directly affect the other agents.

The interest in vehicle formations is fueled by some major advantages that groups of agents might have, compared to individual vehicles. Groups of vehicles have the potential to perform tasks that go beyond the abilities of individual agents [FM04]. Distributed control of formations can also increase robustness and reliability, since a mission goal does not solely rely on a centralized control unit and single agents can be replaced without compromising the overall mission goal. Furthermore, the number of agents can be varied

according to the specific task, and generally less communication capabilities are required compared to centralized systems [PPW09]. In the following, a few examples of vehicle formations are given to illustrate those advantages.

Unmanned Aerial Vehicles (UAVs) Advances in autonomous flight control and navigation have enabled various applications for UAV formations. An early and major field of research are military and security systems [Mur07, CPR01]. Fields of interest are autonomous formation flights of aircraft, cooperative spatial reconnaissance and surveillance, or rendezvous maneuvers.

But also civil scientific research shows an increased interest in formation control of UAVs [PPW09]. Applications might be civil reconnaissance or security surveillance. For consistent surveillance (without dead spots), for example, a formation of UAVs is likely to outperform individual, uncoordinated vehicles.

Satellite Missions Next to terrestrial vehicle formations, also spacecraft formations are subject to current research. An US Airforce mission has investigated the potential of microsatellite clusters for high-resolution imaging. The equivalent antenna realized by a formation of those microsatellites is much larger than an instrument deployed on a single spacecraft [Fax02]. Microsatellites have the potential to reduce cost and simultaneously increase functionality and reliability compared to large-scale single satellites [Kri07].

Another example for a currently planned satellite formation flight is the joined NASA/ESA science mission "*LISA*" (Laser Interferometer Space Antenna). It consists of three spacecraft in an equilateral triangular setup to measure gravitational waves [LIS09]. Since the separation of the three satellites constitutes interferometer arms of 5 million km length, a comparable setup is impossible to integrate into a single satellite.

Automated Highway Systems (AHS) With an increasing number of advanced driver assistance systems being integrated into modern automobiles, the technology of AHS is brought closer to reality. The *Program on Advanced Technology for the Highway (PATH)* at the University of California, Berkeley, has proven the capabilities of automated vehicle control [PAT09]. Cooperation of vehicles increases safety, since information sharing between individual vehicles can support collision avoidance. Beyond that, automated vehicle platoons driving at close space allow for higher traffic density in populous areas.

Autonomous Underwater Vehicles (AUVs) AUVs are used for scientific, commercial, and military applications. To solve complex tasks, formations of AUVs have recently been investigated by researchers. Facing the same challenges as satellite- or UAV-formations, the underwater environment poses some further difficulties for AUV formations. Ocean currents can heavily impact vehicle dynamics. Also, underwater communication is rather limited compared to above-ground or satellite communication [Fax02].

1.2 Formation Control

The general goal of formation control is to establish and maintain the position of several agents relative to each other or an external reference. [Mur07] classifies formation control as one subtask of cooperative control of MAS. This control task can be faced with centralized or decentralized approaches. Since superiority of formations over single vehicles largely founds on autonomous agents with decentral control units, this thesis focuses on decentralized formation control.

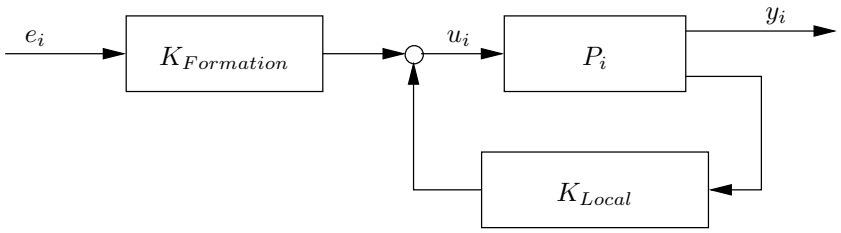

Figure 1.1: Vehicle with two-level controller structure

A decentralized control task, for example the formation control of autonomous quad-rotor helicopters in [PPW09], can be divided into two levels, as illustrated in Figure 1.1. First, each single unit's dynamics (P_i) have to be stabilized internally by a local controller K_{Local}. On a second level, the relative position of the dynamically decoupled units to each other are controlled by $K_{Formation}$. While K_{Local} uses the internal states of a vehicle, $K_{Formation}$ needs information acquired from other agents and from external references. Inter-agent communication is a central point of interest in decentralized formation control. The formation and communication framework used in this thesis is explained in Chapter 2.

1.3 Available Simulation Tools

Several software tools are available to analyze interaction between agents in one or another way. This section briefly investigates some of those tools, which have been found in literature or on the Internet, towards their applicability to the simulation of formations.

With an increased interest in agent-based simulations in various fields of research during the last years, several software libraries have been established. Often, they are available as free open-source software, developed privately or by university research teams. `MASON` [Mas09], `Swarm` [Swa09], `JADE` [BPR99], and `NetLogo` [Net09] are four well-known software frameworks, which are freely accessible and usable. Due to their very general architecture, they allow for various applications, reaching from the simulation of ant behavior [LCRPS04] to insights into the renewable energy market [Wei08]. However, this general architecture requires extensive and costly adaptions to adjust to the requirements for a formation simulation framework, since standard control syntax is generally not supported. Also, special requirements to the simulation environment, like a switching communication topology or communication time delays would not be straightforward to implement.

Next to those freely accessible general software libraries, commercial agent-based simulation platforms are available. `MATSim` [Mat09] and `SIMWALK` [Sim09] are tools to simulate traffic or pedestrian flows. With limitations to those areas of application, they are not suitable to analyze formation control problems. `Webots` is a commercial software package for the simulation of mobile robots [Cyb09]. It allows to model robot interaction and communication via sensors, and to program controllers in standard programming languages. However, `Webots` focuses on prototyping of robots and their components. To get insights into formation behavior, the agents in a formation would have to be physically developed before, which is beyond the scope of this thesis. Thus, `Webots` might be a promising platform for future work on developing real vehicles, but is too restricted to physical representations of agents for the requirements of the simulation environment designed in this thesis.

As further explained in Section 3.3, `Matlab`-based simulation environments provide convenient solutions for control analysis. There are several `Matlab`-based simulation testbeds, like a spacecraft formation simulation developed by the NASA Jet Propulsion Laboratory [Jet09] or software evolving from Master theses [Hac09, Cle06]. These simulators are either not freely accessible, or are restricted to simulating very particular problems. No `Matlab`-based software has been found that could conveniently be adapted to the problems discussed in this thesis.

Despite the variety of agent-based simulation software available freely or commercially, no tools could be found that can easily and cost-efficiently be adapted to the formation control framework described in Chapter 2 and, thus, serve as a basis for a formation simulation environment.

1.4 Contributions and Thesis Outline

Formation control is a broad field of research, with various concepts and control strategies being developed. While specialized simulation tools provide possibilities to analyze particular problems, they are often hard to adapt to changing demands. To overcome this problem, this thesis introduces an object-oriented simulation environment that is flexible and easily scalable and extendable.

In detail, this includes:

- Development of an object-oriented simulation environment for formation control that
  - has a modular structure and is easily extendable,
  - implements standard `Matlab` control syntax,
  - supports dynamic changes to the number of agents, communication topology, and formation shape during a simulation,
  - allows for simulations of agents with different dynamics and controllers within one formation,
  - includes transmission time delays, a limited communication range, and obstacles disturbing the communication,
  - implements two different ways to define a formation by its reference input,
  - visualizes simulation results on-the-fly with various animation features and allows to generate plots and save simulation data,
  - is equipped with a graphical user interface (GUI).
- To support future collaborative extensions to the program code by several contributers, a version control system is used.
- For validation, test simulations are carried out, which investigate the influence of controller design, communication topology, and time delays on formation stability and performance.

- Several performance measures are proposed to evaluate different aspects of formation performance and robustness.

Chapter 1 gives a brief introduction to applications of vehicle formations and the concept of formation control. Available simulation tools for MAS are reviewed. The control framework implemented in the simulator, as well as basics of Graph Theory, which is used to model the communication structure, are described in Chapter 2. Chapter 3 depicts the modeling approach that has been followed, and provides detailed explanation of the simulator structure and functionality. The simulation environment is validated with various test simulations in Chapter 4. Additionally, performance measures to support these analyses are defined and evaluated with respect to their significance.

Chapter 2

Formation Control Framework

Various fields of research subsume under the term *formation control*, and terminology is partly ambiguous. [Mur07] categorizes formation control as one subtask of cooperative control in MAS. This chapter depicts the formation control framework, on which the proposed simulation environment founds. Section 2.1 explains the methodology to define the inter-agent communication, Sections 2.2 and 2.3 describe control strategy and related reference input used for the simulations in Chapter 4.

2.1 Communication Topology and Graph Theory

A property which is important to all areas of formation control is the underlying communication topology, where communication entitles all kind of information exchange between agents. Without information about other vehicles in the formation, an agent is not able to maintain its relative position. Thus, information sharing is a necessary prerequisite for control of formations. From an individual agent's point of view, information about other agents can be obtained in two ways. First, the agent can sense the positions of neighboring vehicles via on-board sensors, the other vehicle has a passive role and the information flow is uni-directional [DFK$^+$02, BA98]. Second, agents can transmit information about their own states to other agents. In this case, vehicles need to be capable of sending and receiving, the information flow can be uni- or bi-directional. Either way of information exchange represents a form of communication. To define this communication, a graph-theoretic concept is deployed.

Graph Theory has proven to be a convenient way to mathematically express a formation's communication topology. [Fax02] and [FM04] are early and frequently quoted works,

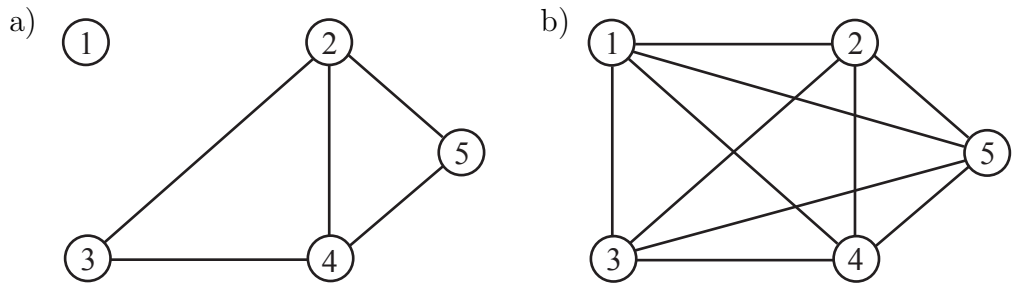

Figure 2.1: Undirected graph, a) $\mathcal{G}_{u1}$ with disconnected vertex, b) $\mathcal{G}_{u2}$, complete graph

which systematically apply graph Theory to formation control and link it to control-theoretic concepts. Some relevant basics are explained below, following the definitions in [Die05]. Advanced mathematical background and detailed derivations and explanations can also be found in [GR04]. [GY06] and [Fou09] give numerous applications.

2.1.1 Basic Definitions

A graph $\mathcal{G}$ consists of a set of vertices $\mathcal{V}$, also called nodes, and a set of arcs (or edges, lines) $\mathcal{A}$. An arc a connects two vertices v_i, v_j, where $a = (v_i, v_j) \in \mathcal{A}$ and $v_i, v_j \in \mathcal{V}$. It is assumed that there are no self-loops in the graph, which means for any $(v_i, v_j) \in \mathcal{A}$, $v_i \neq v_j$. Hence, an arc starting from one vertex never leads back to the same vertex. As long as no direction is specified for the connection between two vertices, that is, for each arc $(v_i, v_j) \in \mathcal{A}$ also $(v_j, v_i) \in \mathcal{A}$, the graph is called *undirected*. A graphical representation of a simple undirected graph is shown in Figure 2.1 (a).

The number of arcs connected to each vertex is given by the degree $d_{\mathcal{G}}(v)$ of a vertex v. A vertex with $d_{\mathcal{G}}(v) = 0$ is disconnected from the graph. If all vertices of $\mathcal{G}$ have maximum degree, meaning every possible arc in the graph exists, $\mathcal{G}$ is said to be *complete* (see Figure 2.1 (b).

2.1.2 Directed Graphs and Connectivity

To utilize Graph Theory for the representation of communication topologies of a group of agents, undirected graphs need to be extended to directed graphs. That is, arcs between vertices not only symbolize a connection of those vertices, but also its direction. A directed graph can be illustrated simply by adding arrow-heads to the arcs, see Figure 2.2. An arrow from v_i to v_j means that v_i has *access* to v_j. Thus, if the vertices represent agents

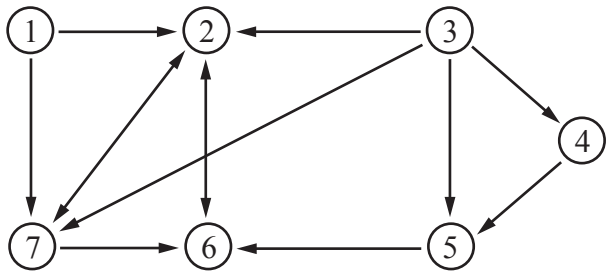

Figure 2.2: Directed graph $\mathcal{G}_d$

in a formation, agent v_3 can sense/receive information from v_2, while v_2 can not receive from v_3. Note that this definition might make the arrow point in the opposite direction of the actual information flow, depending on the kind of information transmission.

For directed graphs, the degree of a vertex $d_{\mathcal{G}}(v)$ as defined in the previous section has to be extended analogously. The *in*-degree $d_{\mathcal{G},in}(v)$ of a vertex v is the number of arcs pointing to v, the *out*-degree $d_{\mathcal{G},out}(v)$ determines the number of arcs starting from v and pointing to another vertex. If in-degree and out-degree of $\mathcal{G}$ are equal for every vertex, $\mathcal{G}$ is undirected.

A graph $\mathcal{G}$, in which every vertex has access to any other vertex (also by concatenating several arcs) is termed *strongly connected* [Fax02]. If there are subsets of vertices in $\mathcal{G}$, which do not have access to other subsets in the graph, $\mathcal{G}$ is divided into several *components*. Since the subset $\{v_3, v_4, v_5\}$ in Figure 2.2 can access the remaining graph, but can not be accessed by the other vertices in $\mathcal{G}$, it is said to be one component of $\mathcal{G}$. Likewise, the subset $\{v_2, v_6, v_7\}$ and the single vertex $\{v_1\}$ are components of $\mathcal{G}$. All three components are depicted in Figure 2.3.

2.1.3 Algebraic Representation

Graphical representation of graphs is convenient to support visualization and understanding of a problem. Algebraic graph Theory links the structure of a graph to different matrix representations of the same graph, which are beneficial for computational processing. The algebraic definition of a graph, which is the most relevant in the context of this thesis, is the *adjacency matrix* $\boldsymbol{A}$. The adjacency matrix of a graph $\mathcal{G}$ is a square matrix of size $n \times n$, where n is the number of vertices v_i in $\mathcal{G}$. It is defined as

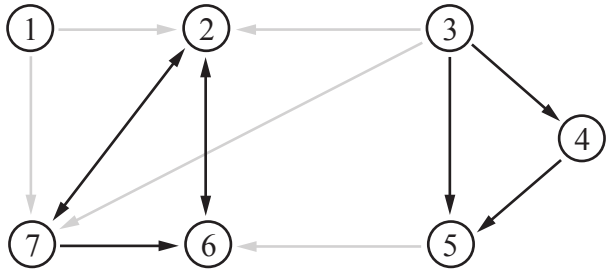

Figure 2.3: Components of $\mathcal{G}_d$

$$a_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in \mathcal{A}(\mathcal{G}) \\ 0, & \text{otherwise} \end{cases} \tag{2.1}$$

The i-th row of $\boldsymbol{A}$ determines, which vertices the vertex v_i can receive from, the i-th column specifies the vertices that can receive from v_i. The adjacency matrix of the graph in Figure 2.2, for example, is

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.2}$$

A second important matrix-representation of a graph is the normalized *Laplacian* matrix $\boldsymbol{L}$ [Fax02], which is defined as

$$\boldsymbol{L} = \boldsymbol{D}^{-1} (\boldsymbol{D} - \boldsymbol{A}) \tag{2.3}$$

where $\boldsymbol{D}$ is a matrix with the out-degree $d_{\mathcal{G},out}(v_i)$ of each vertex on the diagonal. The Laplacian corresponding to the adjacency matrix in (2.2) and the graph in Figure 2.2 is

$$\boldsymbol{L} = \begin{pmatrix} 1 & -\frac{1}{2} & 0 & 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 1 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{1}{4} & 0 & -\frac{1}{4} \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & -\frac{1}{2} & 0 & 0 & 0 & -\frac{1}{2} & 1 \end{pmatrix} \tag{2.4}$$

Note that each row sums to one and the outgoing arcs of each vertex are weighted equally.

While the adjacency matrix uniquely defines the structure of a graph, the Laplacian holds additional information. Particularly the eigenvalues of $\boldsymbol{L}$ are frequently analyzed by researchers to gain insight into graph-theoretic properties and formation-stability issues [FM04].

2.2 Control Strategy

A control strategy in formation control can focus on several areas, some of which are formation synthesis, formation maintaining, inter-agent collision avoidance, obstacle collision avoidance, and tracking.

[FM04] distinguishes between two main categories in formation control, the *leader-follower* approach and the *virtual leader* approach. In the leader-follower approach, a reference trajectory is defined by the leading vehicle and other vehicles try to maintain a predefined relative distance to the leader and among each other. The virtual leader approach states that the vehicles in the formation synthesize a common single reference point, which acts as a virtual leader for the formation. This task is often achieved by *consensus algorithms* [OSFM07, XA05, PRS07], which allow the agents to agree on common states and, thus, synthesize a common virtual leading vehicle. The simulation environment developed as the subject of this thesis implements the leader-follower approach and is further described in Chapter 3. However, it is aimed at keeping the design as flexible as possible to allow for future extensions to other concepts.

The control strategy, which will be used to validate the simulator and for analysis in Chapter 4, is described in [PPW09], which founds on [FM04]. Each agent in the formation consists of a model P, representing the agent's dynamics, and a controller K. K

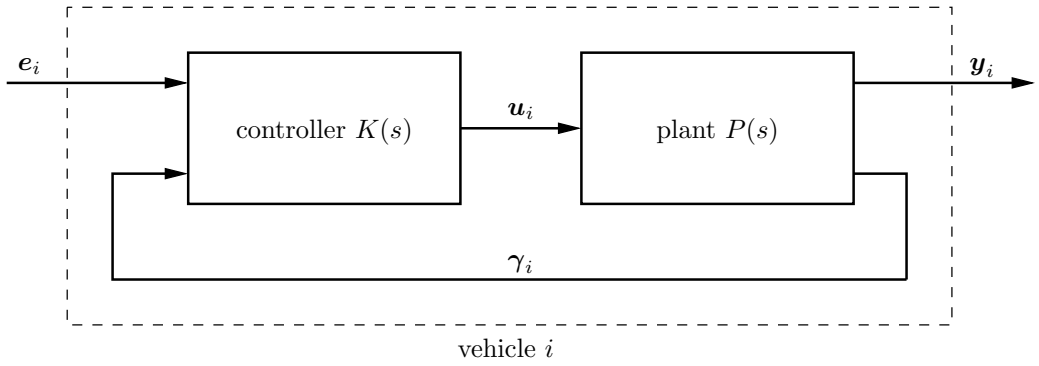

Figure 2.4: Block diagram of a single agent

simultaneously stabilizes the vehicle and compensates for the position/orientation error within the formation. A block diagram of one single agent is depicted in Figure 2.4. The dynamics of agent i with internal states $\boldsymbol{x}_i$ in state space form are

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= \boldsymbol{A}_P \boldsymbol{x}_i + \boldsymbol{B}_P \boldsymbol{u}_i \\ \boldsymbol{y}_i &= \boldsymbol{C}_{P,1} \boldsymbol{x}_i + \boldsymbol{D}_P \boldsymbol{u}_i \end{aligned} \tag{2.5}$$

where $\boldsymbol{u}_i$ is the control input signal and $\boldsymbol{y}_i$ the sensible output, that is, coordinates and spatial orientation of the agent. [PPW09] neglects $\boldsymbol{D}_P$ for real-world quadrocopter agents. Since physical signals are never delay-free, there is no direct feed-through of the input signal $\boldsymbol{u}_i$.

As Figure 2.4 illustrates, a signal $\boldsymbol{\gamma}_i$ closes the internal feedback loop of the agent, which might have other dimensions than $\boldsymbol{y}_i$. The complete state space model for one agent is

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= \boldsymbol{A}_P \boldsymbol{x}_i + \boldsymbol{B}_P \boldsymbol{u}_i \\ \boldsymbol{y}_i &= \boldsymbol{C}_{P,1} \boldsymbol{x}_i \\ \boldsymbol{\gamma}_i &= \boldsymbol{C}_{P,2} \boldsymbol{x}_i \end{aligned} \tag{2.6}$$

The control law for each agent writes

$$\begin{aligned} \dot{\boldsymbol{v}}_i &= \boldsymbol{A}_K \boldsymbol{v}_i + \boldsymbol{B}_{K,1} \boldsymbol{e}_i + \boldsymbol{B}_{K,2} \boldsymbol{\gamma}_i \\ \boldsymbol{u}_i &= \boldsymbol{C}_K \boldsymbol{v}_i + \boldsymbol{D}_{K,1} \boldsymbol{e}_i + \boldsymbol{D}_{K,2} \boldsymbol{\gamma}_i \end{aligned} \tag{2.7}$$

with internal states $\boldsymbol{v}_i$ and input $\boldsymbol{e}_i$. The inner loop to stabilize the agent's dynamics is closed by the feedback signal $\boldsymbol{\gamma}_i$.

For control of the vehicle's relative position within the formation, a second loop needs to be closed to feed back states of other agents (see Figure 2.5). This second loop represents

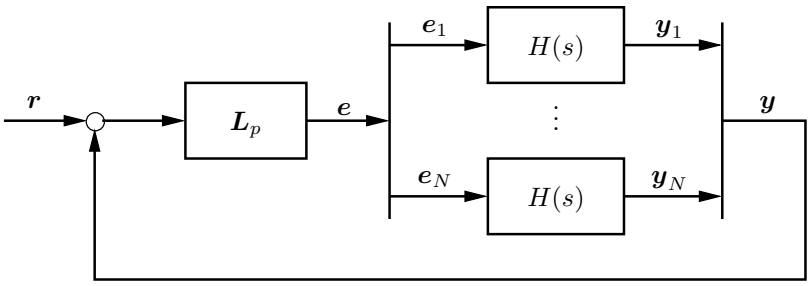

Figure 2.5: Closed loop system of a formation

the communication between agents. From the fed-back outputs $\boldsymbol{y}$ and the reference signal $\boldsymbol{r}$, an error signal $\boldsymbol{e}$ has to be derived for each agent. [FM04] defines $\boldsymbol{e}_i$ as the equally weighted sum of errors relative to all neighbors agent i can receive information from,

$$\boldsymbol{e}_i = \frac{1}{|\mathcal{J}_i|} \sum_{j \in \mathcal{J}_i} \boldsymbol{e}_{ij} \tag{2.8}$$

with $\mathcal{J}_i$ being the set of vehicles agent i can receive from and $|\mathcal{J}_i|$ the number of those vehicles. $\boldsymbol{e}_{ij}$ is the error between the i-th and the j-th vehicle,

$$\boldsymbol{e}_{ij} = \boldsymbol{r}_{ij} - (\boldsymbol{y}_i - \boldsymbol{y}_j) \tag{2.9}$$

$\boldsymbol{r}_{ij}$ is the relative reference input, for example the desired distance between agents i and j.

The dimensions $\boldsymbol{e}$, $\boldsymbol{r}$, and $\boldsymbol{y}$ in Figure 2.5 hold error signals, absolute reference signals, and outputs for all vehicles in the formation in one single column vector, $\boldsymbol{e} = \left[\boldsymbol{e}_1^T, ..., \boldsymbol{e}_N^T\right]^T$, $\boldsymbol{r} = \left[\boldsymbol{r}_1^T, ..., \boldsymbol{r}_N^T\right]^T$ and $\boldsymbol{y} = \left[\boldsymbol{y}_1^T, ..., \boldsymbol{y}_N^T\right]^T$. The error $\boldsymbol{e}$ can then be computed with matrix calculus as

$$\boldsymbol{e} = \boldsymbol{L}_{(p)} (\boldsymbol{r} - \boldsymbol{v}) \tag{2.10}$$

with $\boldsymbol{L}_{(p)} = \boldsymbol{L} \otimes \boldsymbol{I}_p$. $\boldsymbol{I}_p$ is the unity matrix of rank p, where p is the number of outputs of a single agent.

2.3 Reference Input

Next to the communication topology, which defines *who* can exchange information, it is also important *what* kind of information is transmitted. Depending on the controller design, this question is directly related to the type of reference input that is available to the agents, since the reference input is compared to the transmitted outputs of other

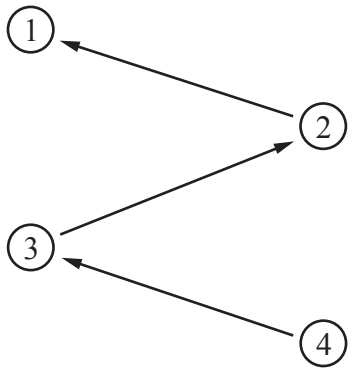

Figure 2.6: Formation defined by directed vectors

agents, in order to generate a control signal. The reference input holds the information, which distances or positions the agents should keep relative to each other. This thesis focuses on two types of reference signals, which are explained in the following sections.

2.3.1 Directed Vectors

The controller design approach followed in [PPW09] uses directed vectors to specify the reference input. If the outputs $\boldsymbol{y}_i$ in Figure 2.5 hold the three spatial coordinates $\boldsymbol{y}_i = [x_i, y_i, z_i]^T$, for example, then the $\boldsymbol{r}_{ij}$ in (2.9) represent a directed vector between agents i and j,

$$\boldsymbol{r}_{ij} = \begin{pmatrix} r_{x,i} - r_{x,j} \\ r_{y,i} - r_{y,j} \\ r_{z,i} - r_{z,j} \end{pmatrix} \tag{2.11}$$

Thus, a formation in which each agent can only receive information from one neighbor, as in Figure 2.6, is uniquely defined. With the robust controller from [PPW09], this formation will also be stable. Note that the relative vectors in (2.11) are not limited to the three spatial coordinates, but could be extended to include agent rotations as well.

2.3.2 Distances

A second approach to define a formation are relative distances. In contrast to directed vectors, only the undirected distance between two agents is specified. In the case of

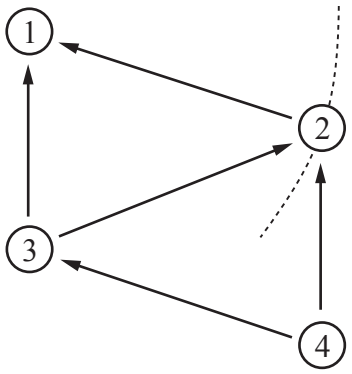

Figure 2.7: Formation defined by undirected distances

three-dimensional spatial coordinates, the relative distance between two agents writes

$$d_{ij} = \left\| \begin{pmatrix} r_{x,i} - r_{x,j} \\ r_{y,i} - r_{y,j} \\ r_{z,i} - r_{z,j} \end{pmatrix} \right\| \tag{2.12}$$

With specifying distances as reference input, the formation in Figure 2.6 is not rigid anymore. An agent, which only receives from one neighbor, can be anywhere on a circle around the other agent to drive its error signal to zero. Thus, further requirements to the underlying communication topology have to be established. [YADF09] developed a control law for two-dimensional formations, assuming a *minimally persistent* communication topology. A minimally persistent formation is defined by a graph, which is minimally rigid and each vertex (agent) has an out-degree of not more than two. Applying this definition to the formation in the previous section, the communication topology is extended as illustrated in Figure 2.7. Now, the formation is rigid again. However, there is one additional degree of freedom (DoF), compared to a formation defined by directed vectors. Agent 2, introduced as the *first follower* by [YADF09], only receives from the leader. In the orthogonal direction, on a circle around the leader, the first follower can rotate the formation. Thus, in a two-dimensional formation defined by undirected distances, a leader can only define the position, while it requires a co-leader to specify the orientation.

Chapter 3

Simulator Architecture

The design of the proposed simulation environment is influenced by various requirements, such as dynamic changes of the communication topology or the number of agents, easy future extendability, and a modular structure. This chapter outlines the modeling approach in the following section, briefly describes the object-oriented design in the `Matlab` environment in Sections 3.2 and 3.3, and details the simulator structure in Section 3.4.

3.1 Modeling Approach

The first step in developing an environment to simulate the real world is to decide on a modeling approach. This requires to reduce the complex real world to significant attributes and find suitable models for those attributes. In the sense of this thesis, the real world is represented by the formation control framework as described in Chapter 2. Relevant attributes in this framework are the *agents* in a formation, the *space*, in which a formation is located, the *time*, along which it propagates in space and possibly changes its shape, communication topology, or number of members, and the *communication* between agents.

3.1.1 Agents Modeling

In the simulation environment, agents, for example quad-rotor helicopters as in [PPW09], are defined by their dynamic model. In case of a state space representation as in (2.6), the linearized model simplifies vehicle dynamics. However, the simulator also allows to consider nonlinear models.

Agents do not have any spatial extent. That is, they can be located at coincidental points and not collide. However, depending on the dynamical model, they have a spatial orientation and can rotate around the given DoF.

3.1.2 Space Modeling

Many of the applications of formation control in Section 1.1, like UAVs, AUVs, or satellite missions include three-dimensional constellations of vehicles. Thus, the simulation environment allows to implement and visualize two- and three-dimensional systems. Space is implemented as a Cartesian coordinate system free of physical obstacles. An agent's coordinate vector has the form

$$\boldsymbol{c}_{3D} = [x \;\; y \;\; z \;\; \theta \;\; \eta \;\; \phi]^T \tag{3.1}$$

$$\boldsymbol{c}_{2D} = [x \;\; y \;\; \phi]^T \tag{3.2}$$

where x, y, z are the position of an agent and θ, η, ϕ its orientation. The measurement dimension is not specified and depends on the units used in the model. The three spatial dimensions are not physically limited. However, their size is restricted by the `Matlab` data type *double*.

3.1.3 Time Modeling

Requirements, such as dynamic changes to the formation during a simulation, as well as inter-agent communication at specific points in time, suggest a discrete-time environment. In the real world, continuous time poses challenges for the implementation of, for example, a clocked communication. A discrete time simulation environment simplifies a timed communication and execution of tasks by providing a global clock with constant timesteps k of length t_s for the system.

Agent dynamics given in continuous time, as in (2.6), can easily be transformed to a discrete-time representation [Wer09]

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Omega}_P \boldsymbol{x}_k + \boldsymbol{\Gamma}_P \boldsymbol{u}_k \tag{3.3}$$

$$\boldsymbol{y}_k = \boldsymbol{C}_P \boldsymbol{x}_k + \boldsymbol{D}_P \boldsymbol{u}_k \tag{3.4}$$

where $\boldsymbol{\Omega}_P$ and $\boldsymbol{\Gamma}_P$ are system matrix and input matrix of the discrete-time system. The control law in (2.7) can be transformed analogously. Since physical signals are never delay-free, real-world agents need at least one timestep to compute the next outputs. Thus,

there is no direct feed-through from the input u_k to the output y_k, and in the following

$$\boldsymbol{D}_P = 0 \tag{3.5}$$

3.1.4 Communication Modeling

Since the different agents in a formation are dynamically decoupled, communication is the only way of interaction between agents. As outlined in Section 2.1, inter-agent communication is modeled using Graph Theory. The adjacency matrix determines if for two given agents i and j a communication link exists. This communication channel is defined as a directed link, as depicted in Figure 3.1.

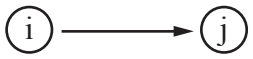

Figure 3.1: Directed communication link, i receives from j

The direction of a communication link follows the definition in [FM04], thus, an arrow from i to j means that i has access to information from j, which does not necessarily represent the physical direction of information flow. This convention goes in line with the UML definition for *getter*-methods [BRJ05], i *gets* information from j.

3.2 Object-Oriented Design

Some key demands in the motivation to develop this formation control simulation environment are to provide a flexible platform that

- is easily scalable to the number of agents,
- allows for simulations of formations with dynamically changing communication topology,
- supports on-the-fly adding and removing of agents,
- is easily extendable,
- is intuitively understandable and usable.

While classic procedural programming approaches often provide a quick and cheap solution to a particular control problem, they have proven to struggle with the requirements mentioned above. Object-oriented programming (OOP) offers a good way to meet those requirements. In the following, some key concepts of OOP are briefly reviewed.

3.2.1 Abstraction

In order to solve problems, programming languages provide abstractions of the real world [Eck06]. OOP follows a very intuitive way of abstraction by modeling problem with several *objects*. Objects in the program code can then be very similar to real-world objects; they have a name, properties, and methods/actions one can perform on them. Objects are instances of classes, which define their properties and methods. Figure 3.2 shows an UML class diagram[1] of a class `cup`. An object of this class has properties like its *content* or *fillingLevel*. Those properties can be changed by the user by invoking the object's methods $fill()$ or $drink()$.

Likewise, the object-oriented design of the simulation environment aims at providing an intuitively understandable and easy usable platform. Every agent, for example, is an instance of the class *agent* with properties and methods similar to a real-world agent in a formation. This approach supports intuitive usability and understanding, as well as dynamic changes during program execution, since an additional agent will just be created by another instance of its corresponding class.

Figure 3.2: Class `cup`

3.2.2 Reusability and Extendability

One of the major advantages of OOP is the reusability and extendability of code [Eck06]. There are two main concepts to achieve this.

First, objects of existing classes can be used within new classes, the functionality of those existing classes can be reused. This concept is called *composition*. The `cup` class in Figure

[1] All unified modeling language (UML) diagrams in this thesis follow the definitions in [BRJ05]

3.2, for example, could be used in another class `café`. For the class `café`, the functionality of the cups in it would not have to be defined again.

As a second approach, existing classes can be extended to new classes. That is, properties and methods of an existing class are extended or changed within a new class, the new class is derived from an existing class by *inheritance*. To create cups with a label of a café printed on them, for example, a new class `labeledCup` could be derived from the existing class `cup`, which inherits all properties and methods of the parent class and just adds an extra property *label*.

The simulation environment makes use of composition and inheritance, as explained in Section 3.4, to allow for extension of the program and reusability of all or parts of the code in future projects.

3.2.3 Object Interaction Paradigms

In OOP, two main programming paradigms which describe the interaction between objects have to be distinguished [dCLF93]. In the well-known case of synchronized communication, objects interact by calling each others methods. Although one could think of this procedure as an uni-directional communication, the calling object waits for a reply and, thus, the communication is synchronized and bidirectional.

The second approach is *event-driven* programming. That is, objects can trigger an event, and other objects, which listen to this event, execute a specified behavior. Since the object that triggers the event does not know about the listeners to this event, this way of communication is unidirectional and unsynchronized.

For the simulation environment in this thesis, a hybrid approach is followed. To ensure a stable and robust simulation run, the main simulation follows the concept of synchronized communication, which leads to a sequential program execution. For example, agents in the formation should not transmit their new states to each other before every agent has finished computing them. The sequential simulation loop is explained in Section 3.4. For user-friendly operation, a (GUI) is developed, see Section 3.5. A GUI is a typical event-driven application, since it reacts to user prompts. The event-driven GUI triggers processes of the sequentially executed simulation environment.

3.3 Matlab Environment

The proposed simulation environment is developed in `Matlab`, a widely used software for control systems engineering. This way, systems and controllers defined in `Matlab` can be conveniently tested in the simulator, and output data can easily be processed with existing tools. Providing the simulation environment as a `Matlab` toolbox also supports future extensions by researchers, who might not be familiar with "traditional" OOP languages, like `JAVA`.

With the `2008a` release, `Matlab` supports OOP features similar to those of other object-oriented languages like `JAVA` or `C++`. OOP has also been possible in versions prior to the `2008a` release [Reg07], syntax was rather inconvenient however. Since `2008a`, it is possible to define properties and methods in class definition files. Hence the proposed simulation environment requires a `Matlab` release of `2008a` or higher.

3.4 Structure

The proposed simulation environment follows a hierarchical structure. Objects in the program represent either physical components in the real world, like the agents in a formation, or features of the simulator. The UML class diagram of the simulation environment in Figure 3.3 illustrates the hierarchical composition. The class `SimulationManager` provides the overall organizational structure and centralizes all functionalities of the simulator. As the head of the hierarchy, it contains all other objects, directly or as components of other objects.

The classes directly contained in `SimulationManager` are `RefInputGenerator`, `Formation`, and `Animator3D` or `Animator2D` respectively, depending on the dimensions chosen by the user. Those two classes inherit from the parent class *`Animator`*. `Formation` holds one object of `Memory` and `VirtualLeader`, and several objects `Agent`, depending on the size of the formation. `VirtualLeader` and `Agent` inherit from their parent class *`GeneralizedAgent`*. Each `Agent` object contains a model, which is either a `LinearModel` object or a `QuadrocopterModel` object, both derived from the parent class *`Model`*. The `QuadrocopterModel` is an example for a nonlinear model implementation, further nonlinear models can easily be configured as child classes of *`Model`*.

The classes *`Animator`*, *`GeneralizedAgent`*, and *`Model`* are abstract classes. This means, they serve as parent classes for other classes, but do not have any direct instances. A

closer look at the functionalities of each class is given in the following section. Section 3.4.2 depicts the temporal process of a simulation run.

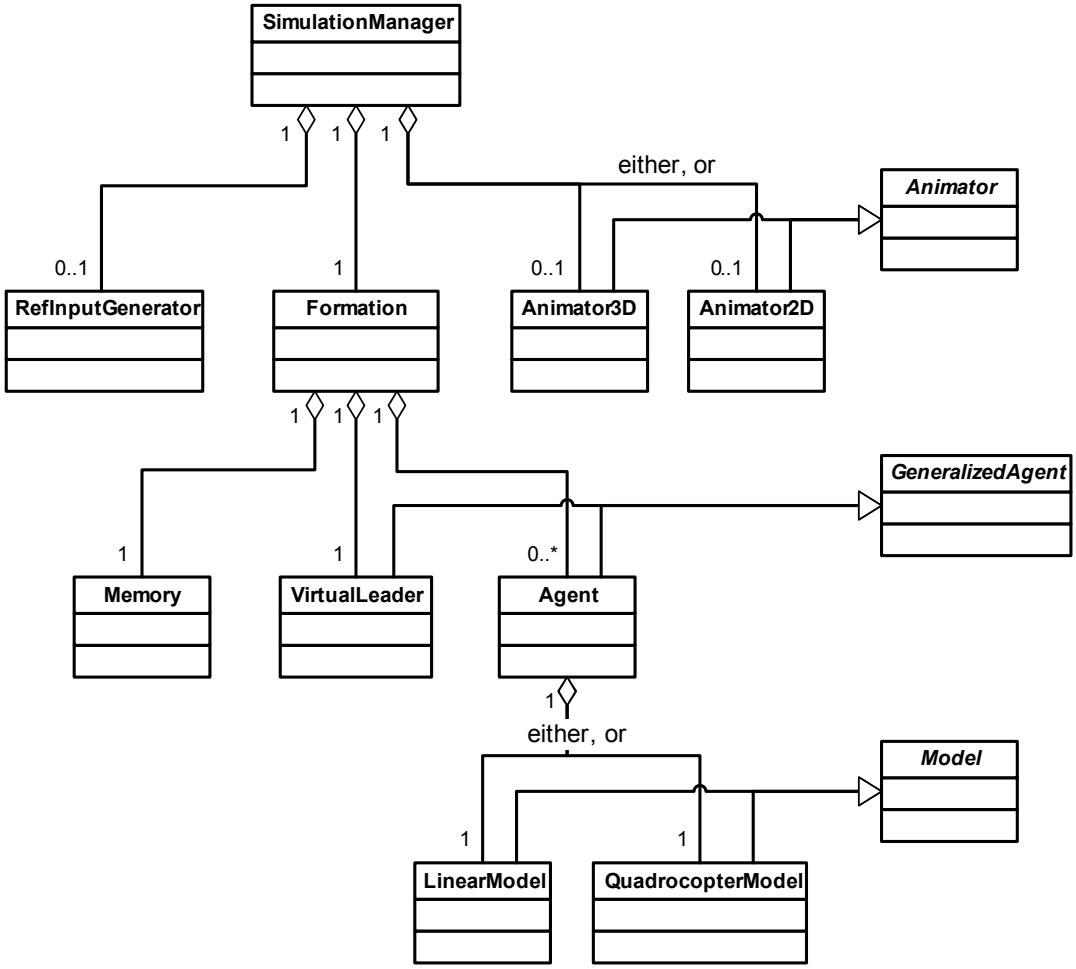

Figure 3.3: UML class diagram of the simulation environment

3.4.1 Classes

For the purpose of easy reference and future extendability, the individual classes of the simulator are presented in this section. Each class consists of a class name, properties, and methods. For abstract classes, the class name is printed in italic characters. This also holds for abstract methods, which are implemented in any of the corresponding child classes, but do not have an implementation in the parent class.

Properties and methods are provided with different access identifiers. *Public* properties or methods are indicated with a preceding "+", they can be accessed by any other object

in the program. Properties and methods, which can only be accessed by their own object, are called *private* and marked with a "−". Private methods that can also be accessed by objects of inherited classes are *protected* and identified with a "#".

For clarity and simplicity, all getter- and setter-methods are omitted in the diagrams in this section. The only purpose of those methods is to retrieve or set the value of a method's property. The most important properties and methods of each class are explained in the following.

Simulation Manager The class `SimulationManager` controls the simulation flow and is depicted in Figure 3.4. It is responsible for the initialization of the formation and simulates every timestep according to its global sampling time t_s. During each simulation loop, the formation or its reference input might be changed with *changeFormationRefInput()* and *changeFormation()*. Depending on the dimension of the considered system, each timestep can be visualized with an `Animator2D` or `Animator3D` object. All settings for the formation setup and the animation are stored in *settings*. To assign an unique ID to each agent (since the number of agents might change during a simulation run), the *idCount* variable stores the highest used agent ID. To further process simulation data, the methods *plot()* and *saveData()* allow to plot and save agent positions and the underlying communication topology.

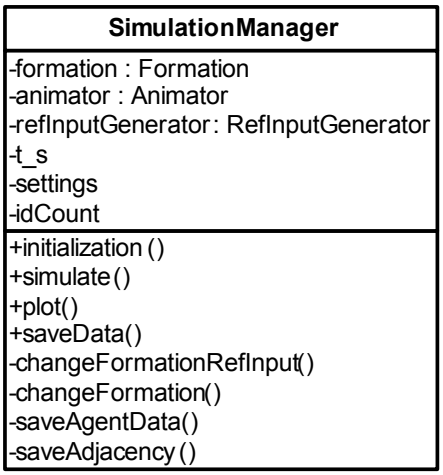

Figure 3.4: Class `SimulationManager`

Formation As the point of interest for the simulation environment, the `Formation` class in Figure 3.5 subsumes all objects that are physically involved in a formation of

agents. It holds a cell array *agents* containing the virtual leader and all agents being present in the formation. The `formation` class organizes the interaction between agents (*agentsSend()*, *agentsReceiveFromOthers()*) and can add and remove agents (*addAgent(), addVirtualLeader(), removeAgent()*). It keeps track of the relevant formation properties (*agentIds, formationRefInput, relRefInput, outputs, coordinates, agentDistances*) and the communication topology (*adjacency*), as well as changes to it (*modifyAdjacency()*). An object *memory* is used to store agent coordinates at each timestep (*saveAgentData()*) and their respective communication topology (*saveAdjacency()*).

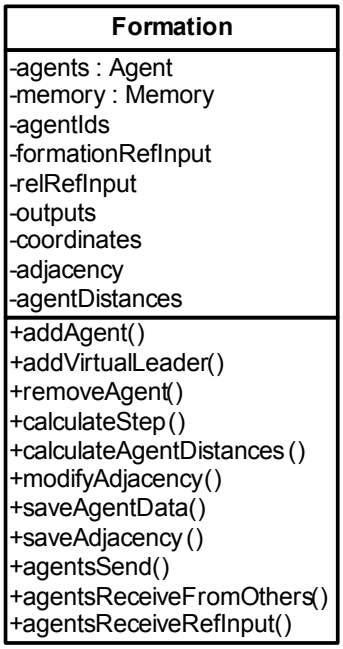

Figure 3.5: Class `Formation`

Agents The agents in the formation are represented by instances of the class `Agent`, which inherits from the class `GeneralizedAgent`, see Figure 3.6. Also the virtual leader, which acts as reference input for the formation, inherits from `GeneralizedAgent`, since many actions have to be performed on both the agents and the virtual leader. Common for both classes are a unique *id*, input and output vector *refInput* and *output* and spatial coordinates *coordinates* derived from the output vector. All methods that are common to both agents and virtual leader are defined in the `GeneralizedAgent` class.

Each agent contains an object `Model`, which holds models of the controller and agent dynamics. The method *calculateStep()*, which calls *calculateE(), calculateNewState(), and calculateOutput()*, computes the formation error and the new states and output of

the agent. In order to do this, the agent needs the outputs (*outputOthers*) of other agents, which it can receive from, and, in case of time delays, its own backdated output (*ownSavedOutput*). Since the virtual leader only represents a reference input, it does not require any other properties. The methods *calculateStep()* and *calculateOutput()* update the reference input and make it available to the other agents in the formation, which can receive the virtual leader's output.

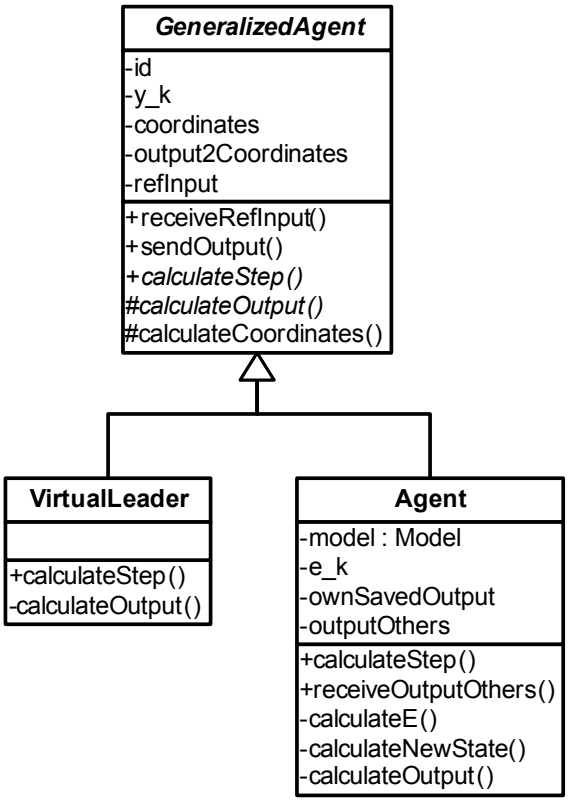

Figure 3.6: Abstract class `GeneralizedAgent` with child classes `VirtualLeader` and `Agent`

Model As an abstract class, `Model` in Figure 3.7 does not have any direct instances. For the standard case, the class `LinearModel` is derived as a child. To allow for implementation of nonlinear models or controllers as well, further child classes can be created. As an example for the implementation of a nonlinear dynamic model, the class `QuadrocopterModel` has been built.

Common to all models are controller and model states u_k, v_kplus1, y_k, x_kplus1, and an offset value v_0, representing the initial position of the agent. The abstract method

calculateStep() has different implementations in the child classes, depending on the kind of model or algorithm used.

The `LinearModel` class holds a *controllerModel* and a *plantModel* as standard `Matlab` state space representations. The quadrocopter model has the same linear controller *controllerModel* and needs the additional variables x and t for the nonlinear model algorithm implemented in *calculateStep()*.

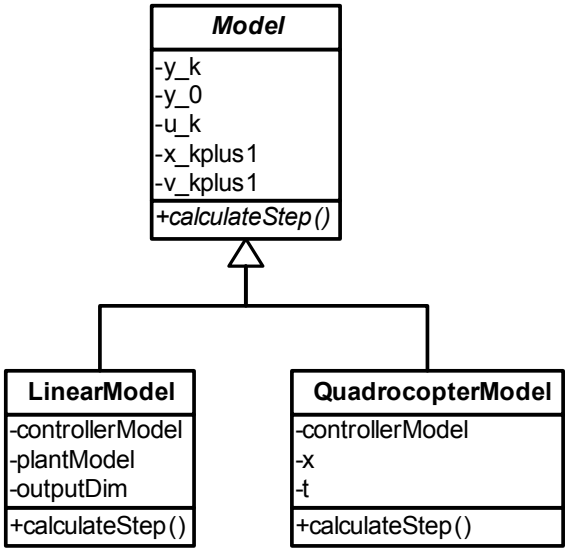

Figure 3.7: Abstract class *`Model`* with child classes `LinearModel` and `QuadrocopterModel`

Animator For visualization of results during a simulation, the simulation manager contains an object inherited from the abstract *`Animator`* class. At creation time of the `SimulationManager` object, the dimension specified in the initialization file determines if an `Animator2D` or `Animator3D` object is created. Both classes have the same methods, which only differ slightly due to different `Matlab` commands for two or three dimensions. The various settings variables of *`Animator`* determine graphical properties of the animation. The public method *animate()* plots the agents and calls, depending on the specified plotting settings, the private methods to plot arrows, trails, agent IDs, and obstacles, and to resize the current axis dimensions, if necessary. A matrix in the user input file determines, which entries of the agents' output vectors are visualized (i.e., which entries correspond to the physical dimensions x, y, z).

Memory An instance of the `Memory` class in Figure 3.9 is contained in the `Formation` object. The methods *saveOutputs()* and *saveAdjacency()* store agent data and the ad-

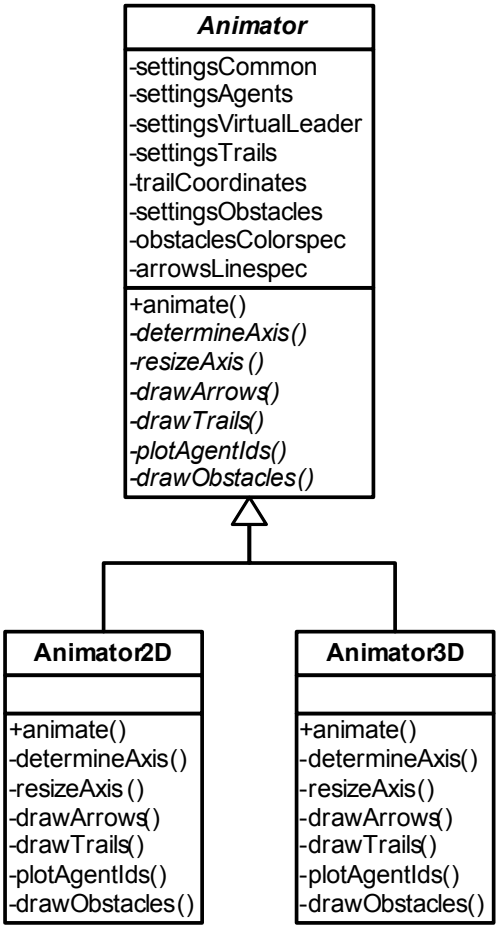

Figure 3.8: Abstract class *Animator* with child classes Animator2D and Animator3D

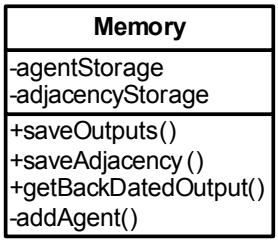

Figure 3.9: Class Memory

jacency matrix of the formation in the corresponding variables *agentStorage* and *adjacencyStorage*. This data can be retrieved after a simulation run for plotting or further analyses. An additional function of the Memory object is to provide agent outputs from preceding timesteps in case of time delays. The method *getBackDatedOutput()* allows to

access backdated data for each agent in the formation, which can be used to represent signal transmission time delays.

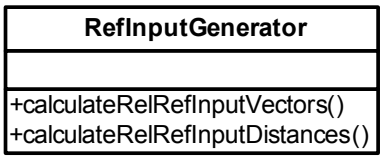

Figure 3.10: Class `RefInputGenerator`

Reference Input Generator The `RefInputGenerator` class (see Figure 3.10) contained in the simulation manager provides the functionality to compute the relative distance from each agent to all other agents from the absolute agent positions specified in the initialization file. Depending on the way the reference input is defined (as directed vectors or distances, as explained in Section 2.3), the methods *calculateRelRefInputVectors()* or *calculateRelRefInputDistances()* compute the corresponding values.

3.4.2 Simulation Flow

The run of a simulation is determined by user input and internal processes. Those processes are controlled by the simulation manager, which in turn is triggered by the GUI. The main simulation loop, divided into those two control instances, is illustrated in an UML activity diagram in Figure 3.11. Some extended functionalities for expert users are provided in Section 3.6.

A new simulation is started on user command by the GUI. The GUI reads initialization data from an input file and passes this data, as well as further simulation and animation settings specified by the user, to the simulation manager. After the new formation is initialized in the simulation manager, and each time the simulation is interrupted, settings can be changed in the GUI.

The simulation loop controlled by the simulation manager is triggered by the GUI. To ensure secure program execution, the simulation manager only reacts to commands of the GUI after every full executed loop. The user can choose to interrupt the simulation, for example to change settings or plot data, or to finish the simulation run. As long as no user input to the GUI is registered, a new loop is executed.

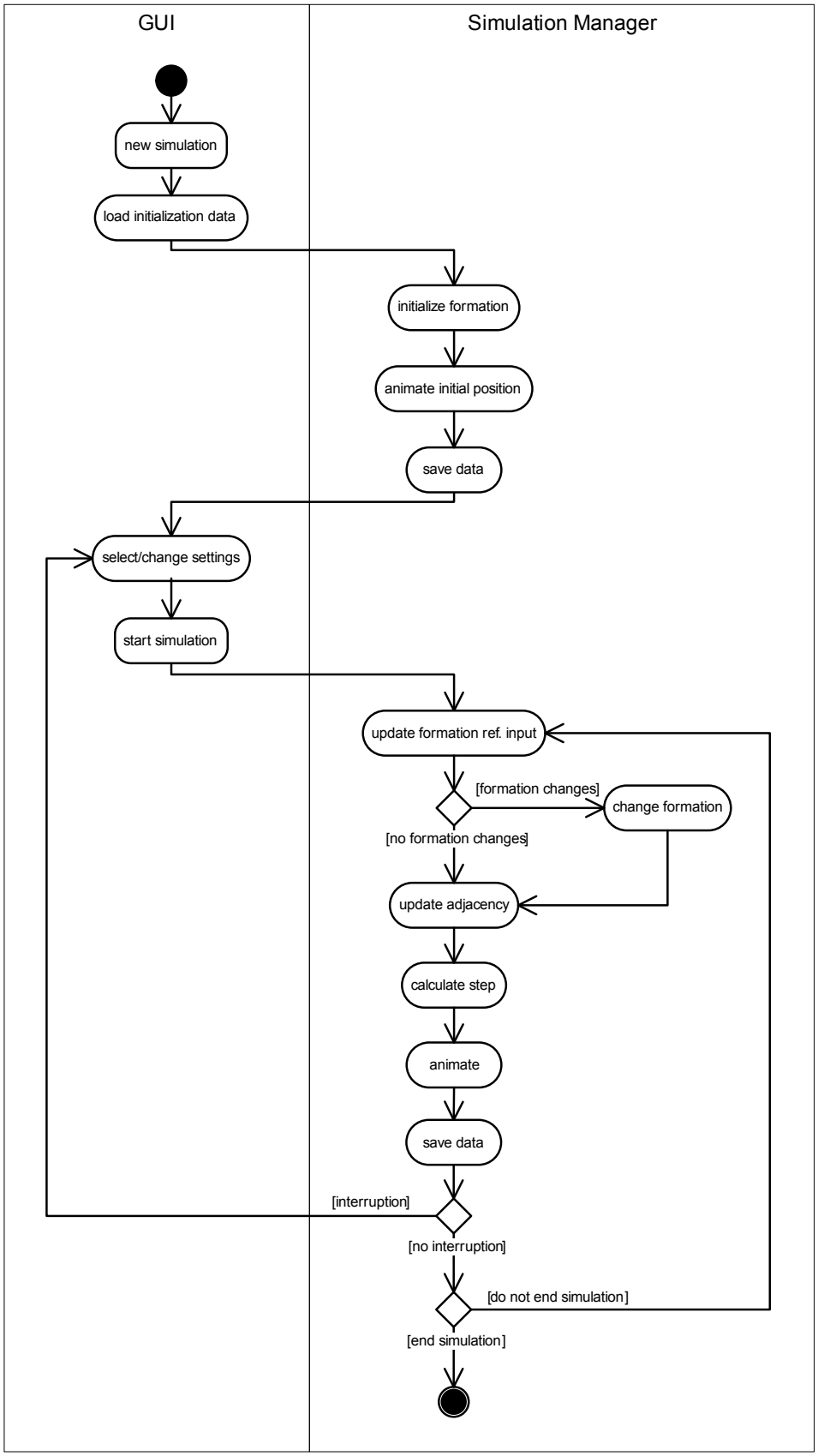

Figure 3.11: UML activity diagram of the top-level simulation process

3.5 Graphical User Interface

The simulation environment contains a GUI, which provides the interface between user and program. The GUI is opened by executing the `Matlab`-file *FAST.m* (FAST stands for Formation Analysis Simulation Toolbox). A screenshot of the GUI is shown in Figure 3.12. Its two main functions are to display the simulation results and to start and control simulations by user input. The GUI is divided into four main functional regions, which are the central visualization frame, a group of general control buttons in the top right corner, control elements for formation properties at the right side of the visualization frame, and control elements for animation settings below the animation frame.

For during-the-simulation visualization, the central part of the GUI shows a two-dimensional or three-dimensional coordinate space, in which, depending on the animation settings, all or part of the formation is plotted. The user can rotate the coordinate frame with a toggle button in the top left toolbar, or switch to plane- or 3D-view with the corresponding buttons. The elapsed simulation time is printed above the coordinate frame.

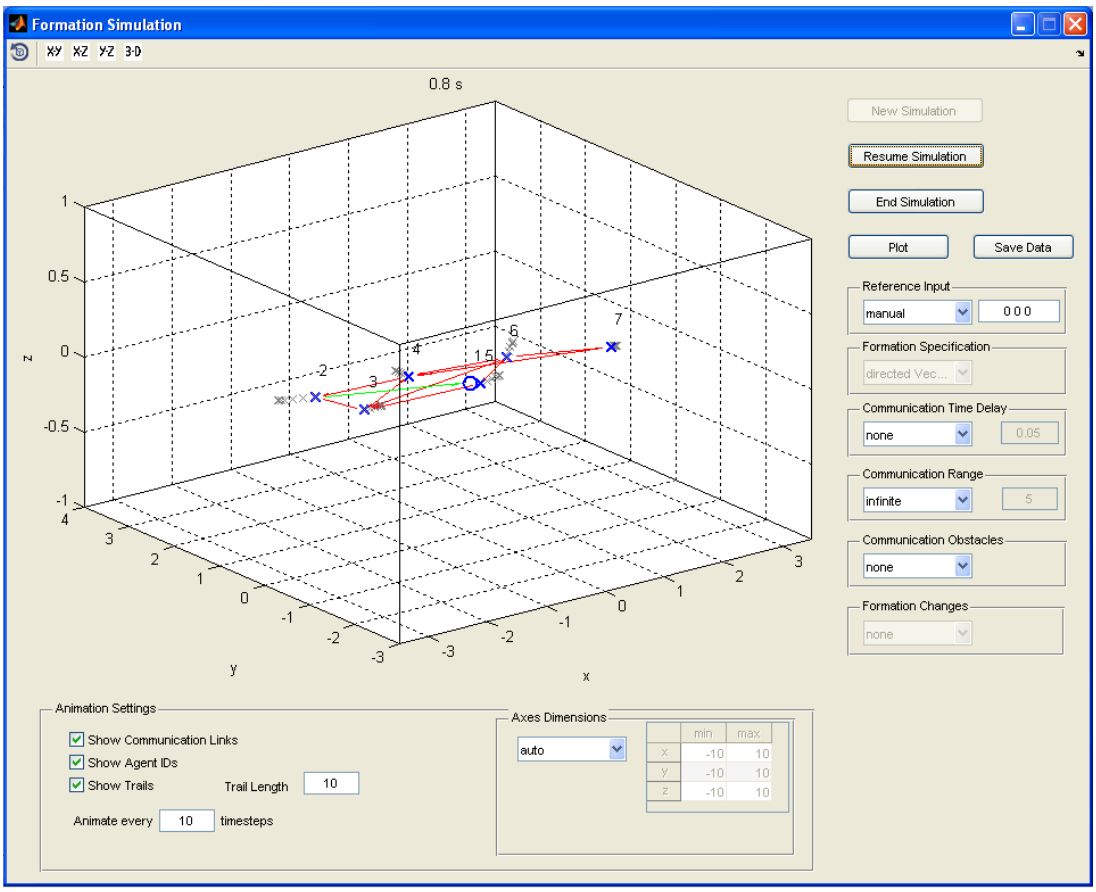

Figure 3.12: Graphical user interface of the simulator

To set up a new simulation and to start, interrupt, or end it, the main control buttons in the top right corner of the GUI are used. While a simulation is interrupted, or after it is finished, the user can request to generate plots or save simulation data to a file.

Formation settings, like reference input, time delays, or a limited communication range, can be set by the drop-down menus on the right side. The user can also define communication obstacles or changes of the formation, which will then be read from a specified file. Formation settings can be changed before a simulation, or during an interruption.

For during-the-simulation animation of the simulation results, the user can select to plot communication links, agent ids, or trails showing the previous positions of each agent. To speed up a simulation, the number of timesteps to be animated can be reduced. By default, the axes of the coordinate frame are resized to show all agents in the formation as close as possible. However, axes can also be manually adjusted by the user.

During initialization of a simulation, the GUI collects all required settings and passes them to an object of the `SimulationManager` class. To initialize formations with requirements that go beyond the capabilities of the GUI, a `SimulationManager` object could also be created and provided with these settings manually, as explained in the following section.

3.6 Advanced Functionality

Settings for most simulations can be controlled by the GUI, as explained in the previous section. Next to an initialization file, which has to be specified prior to a simulation, no further input is required for a variety of applications. This section depicts some extended simulation functionalities provided to the user.

Advanced Initialization Next to formations with equal agent controllers and dynamics, as examined in Chapter 4, the initialization file allows to specify different controllers and dynamic models for each agent in the formation. Since the models for controller and dynamics of each agent are stored in a vector, each vector entry might be defined separately to provide those different models.

Initial positions of agents can be specified by offset values, added to the agents' output vector. If required, also initial states for each agent can be defined in the initialization file.

N-Dimensional Simulations The simulation environment allows to simulate systems of arbitrary dimensions. Since only two or three dimensions can be visualized, a matrix *output2Coordinates* determines, which of the dimensions of the agents' output vectors should be animated. This way, two or three dimensions of n-dimensional systems can be visualized. For 1-dimensional configurations, x- and y-dimension coincide.

Reference Input The user can define the formation reference input in the GUI. Alternatively, a reference trajectory can be specified in a file. For each timestep in a simulation, the next entry of the reference input vector is evaluated. If the number of timesteps exceeds the length of the vector, the last entry is treated as a constant value until the end of the simulation. During a simulation, the user can switch between input from a file and manual input.

Communication Obstacles Communication obstacles can be defined in another input file. They can be configured as balloons or two-dimensional surfaces with arbitrary location. If a communication link between two agents intersects with an obstacle, the link becomes inactive and no communication between the corresponding agents is possible as long as the intersection continues. The user can enable and disable communication obstacles during a simulation.

Formation Changes Various changes of formation properties can be specified in an input file prior to a simulation. Each change is linked to a point in time, at which it occurs. This way, repeatable simulations with changing properties can be realized. The simulation environment implements

- changes in the communication topology,
- changes in the formation shape,
- removal of one or several agents,
- adding of one or several agents.

Several changes can be scheduled for the same time instant and will be executed after each other.

Simulation without GUI The GUI provides a convenient way to initialize, start, control, and end simulations. However, it is also possible to create a `SimulationManager` object manually. At creation time, the user has to provide all required arguments, like initialization data or animation settings, to construct the object. All methods to control a simulation are public and can be invoked manually, once the `SimulationManager` is created.

Chapter 4

Simulation and Analysis

The proposed simulation environment aims at providing a general, extendable controller testbed, to gain insight into several fields of formation analysis and for various types of agents. However, to validate the simulation environment and to carry out analyses, particular scenarios have to be chosen. All analyses in this chapter, except Section 4.2.3, implement a linearized quadrocopter model as derived in [PPW09]. The measurement dimension is meter. The simulator's capabilities are demonstrated by analyses conducted with those quadrocopter models.

For control of quadrocopter formations, three different controllers are used in this chapter:

1. K_{robust}, a robust controller derived in [PPW09] with a H_∞ design approach. This controller claims to be robustly stable for any given communication topology and number of agents.

2. $K_{nominal}$, a controller which stabilizes one single quadrocopter [Pop09]. For formations of quadrocopters, this controller does not guarantee stability.

3. $K_{complete}$, a controller developed to stabilize a formation of ten quadrocopters with a complete communication topology [Pop09]. Thus, it stabilizes any formation of ten copters, as long as there exists a communication link between any two copters in the formation.

Comparing formation performance of different simulations in this chapter raises the question of how to measure this performance. While for stability issues various "hard" criteria exist and a formation can be either stable or unstable, comparing the tracking performance of a formation is ambiguous and requires suitable measures. Thus, this chapter has a dual

focus: Several simulations are analyzed with respect to stability and performance in Sections 4.2, 4.3, and 4.4. To support those analyses, performance measures are proposed in Section 4.1 and evaluated in Section 4.5.

4.1 Definition of Performance Measures

To allow for quantitative formation performance analyses, suitable criteria have to be defined. Four aspects of tracking performance are identified in this section, which describe

- how close a formation can follow a reference trajectory,
- how well the formation shape is maintained while following this trajectory,
- how robust the formation is against communication link failures,
- how robust the formation is against agent failures.

Corresponding measures to deploy those criteria are defined in the following.

Tracking Error To quantify differences in how well a formation can follow a specific reference trajectory, the outputs of the agents in a formation have to be compared to the predefined reference signal for each timestep. Two approaches are presented to compare those values.

First, the output error of each agent, compared to its corresponding reference value, is computed as shown in Figure 4.1 (a). To obtain the overall error for a period of time, those output errors are summed up for all agents in the formation and over all timesteps:

$$\bar{\epsilon}_{t1} = \frac{1}{N_k} \sum_{k=1}^{N_k} \left(\frac{1}{N_a} \sum_{i=1}^{N_a} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| \right) = \frac{1}{N_a N_k} \sum_{k=1}^{N_k} \sum_{i=1}^{N_a} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| \tag{4.1}$$

with N_k being the number of timesteps, N_a the number of agents i in the formation, $\boldsymbol{y}_{i,ref,k}$ the reference signal (for example the spatial coordinates) for agent i at timestep k, and $\boldsymbol{y}_{i,k}$ the current output of agent i at the corresponding timestep. Thus, $\bar{\epsilon}_{t1}$ is the average distance over all agents and timesteps that an agent in the formation deviates from its reference position.

Another approach is to compare the formation's common position error, rather than each agent's error separately. This requires to define one single point as the common position of a formation, see Figure 4.1 (b). To weigh all agents equally, the geometric center of all

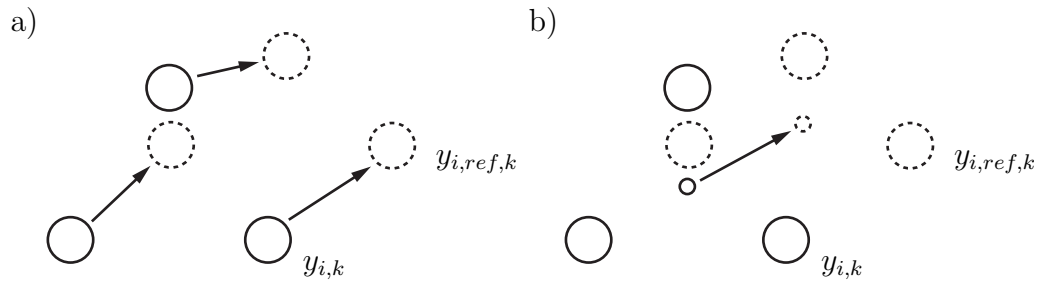

Figure 4.1: Distances considered for the tracking errors, a) $\overline{\epsilon}_{t1}$, b) $\overline{\epsilon}_{t2}$

agents' positions is chosen. This way, one point in space is computed as the position of a formation and compared to the point computed from the reference signal in the same way:

$$\overline{\epsilon}_{t2} = \frac{1}{N_k}\sum_{k=1}^{N_k}\left\|\frac{1}{N_a}\sum_{i=1}^{N_a}\boldsymbol{y}_{i,ref,k} - \frac{1}{N_a}\sum_{i=1}^{N_a}\boldsymbol{y}_{i,k}\right\| = \frac{1}{N_a N_k}\sum_{k=1}^{N_k}\left\|\sum_{i=1}^{N_a}\left(\boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k}\right)\right\| \tag{4.2}$$

$\overline{\epsilon}_{t2}$ is the average distance between the geometric center of the formation and the geometric center of the predefined reference signal over all timesteps.

Formation Error Next to the precision at which a formation can follow a reference trajectory, it is also important how well the formation maintains its shape. A formation shape is determined by the agents' positions relative to each other. Thus, the inter-agent distances are identified as relevant dimensions to be considered.

The first proposed formation error compares the directed vector between two agents in the formation to the vector derived from their reference positions, as illustrated in Figure 4.2. Subtracting those two vectors gives the error-vector (grey vector arrow in the figure). The average length of all error vectors between each two agents over all timesteps is

$$\begin{aligned}\overline{\epsilon}_{f1} &= \frac{1}{N_k}\sum_{k=1}^{N_k}\left(\frac{1}{\frac{N_a(N_a-1)}{2}}\sum_{i=1}^{N_a}\sum_{j=i+1}^{N_a}\left\|(\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k})\right\|\right) \\ &= \frac{2}{N_a(N_a-1)N_k}\sum_{k=1}^{N_k}\sum_{i=1}^{N_a}\sum_{j=i+1}^{N_a}\left\|(\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k})\right\|\end{aligned} \tag{4.3}$$

with N_k, N_a, $\boldsymbol{y}_{i,ref,k}$, and $\boldsymbol{y}_{i,k}$ as specified for (4.1), and $\boldsymbol{y}_{j,ref,k}$ and $\boldsymbol{y}_{j,k}$ analogously for agent j.

A second, broader defined error does not consider directed vectors between the agents, but only undirected inter-agent distances. Hence, the distance between each two agents is compared to the reference distance and summed up for all agents in the formation and

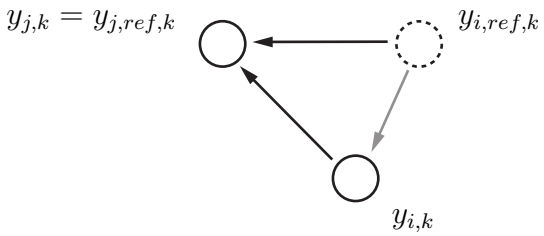

Figure 4.2: Distances considered for the formation errors

over all timesteps,

$$\begin{aligned}\overline{\epsilon}_{f2} &= \frac{1}{N_k}\sum_{k=1}^{N_k}\left(\frac{1}{\frac{N_a(N_a-1)}{2}}\sum_{i=1}^{N_a}\sum_{j=i+1}^{N_a}\left|\left\|\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k}\right\|-\left\|\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k}\right\|\right|\right)\\ &= \frac{2}{N_a(N_a-1)N_k}\sum_{k=1}^{N_k}\sum_{i=1}^{N_a}\sum_{j=i+1}^{N_a}\left|\left\|\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k}\right\|-\left\|\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k}\right\|\right| \qquad (4.4)\end{aligned}$$

where the single bars in the above expression represent the absolute value or modulus. Other than for $\overline{\epsilon}_{f1}$, the lengths of the two black vector arrows in Figure 4.2 are compared directly. Thus, if both vectors have the same length, but different direction, the error would be zero.

Critical-Links Ratio To measure robustness of a formation's performance, two more criteria are defined. Formations might track reference signals very well, but fail fatally as soon as one communication link breaks. The critical-links ratio α_l reveals, how many communication links, relative to the overall number of links, are critical to the formation's existence.

$$\alpha_l = \frac{N_{l,critical}}{N_l} \qquad (4.5)$$

where $N_{l,critical}$ is the number of links critical to the formation, and N_l the total number of communication links. A critical-links ratio of $\alpha_l = 0.1$, for example, means that out of ten existing communication links in a formation, the removal of one link would destroy the formation, while the removal of each of the nine other links does not harm the formation structure. Performance might decrease, however, for any removed link, which is not considered in this ratio.

Critical-Agents Ratio Analogously to the critical-links ratio, a critical-agents ratio is defined. This quotient specifies, how many agents are critical to the formation's existence. An agent is called critical, if its failure would destroy all or part of the remaining formation.

The critical-agents ratio is defined as

$$\alpha_a = \frac{N_{a,critical}}{N_a} \tag{4.6}$$

with $N_{a,critical}$ being the number of these critical agents, and N_a being the total number of agents in the formation.

4.2 Stability

Stability of a formation is a precondition for applications and their safe operation. This section investigates the effect of two different controllers on the overall formation stability, as well as the influence of time delays and changes of the communication topology.

4.2.1 Controller

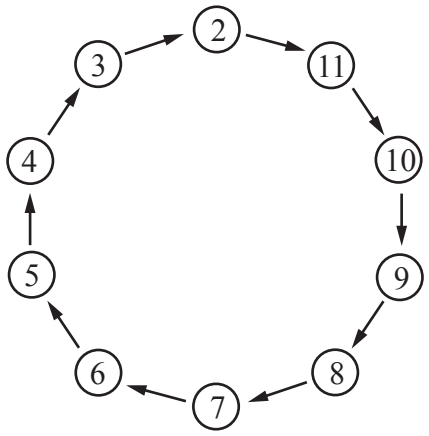

Figure 4.3: Circular communication topology with ten agents

A communication topology, which is challenging for formation stability, is shown in Figure 4.3. In this circular communication structure, each agent only receives from one other agent, its neighbor to one side. The formation does not have access to any external reference input. Thus, errors can easily be amplified in the circular closed loop.

As a model formation in this section, a circular shape is chosen. Thus, Figure 4.3 not only depicts the communication topology, but also the desired formation shape. All agents start from the same initial position, which will lead to a large position error and, thus, to a high acceleration of each agent in the beginning of the simulation. Note that, for

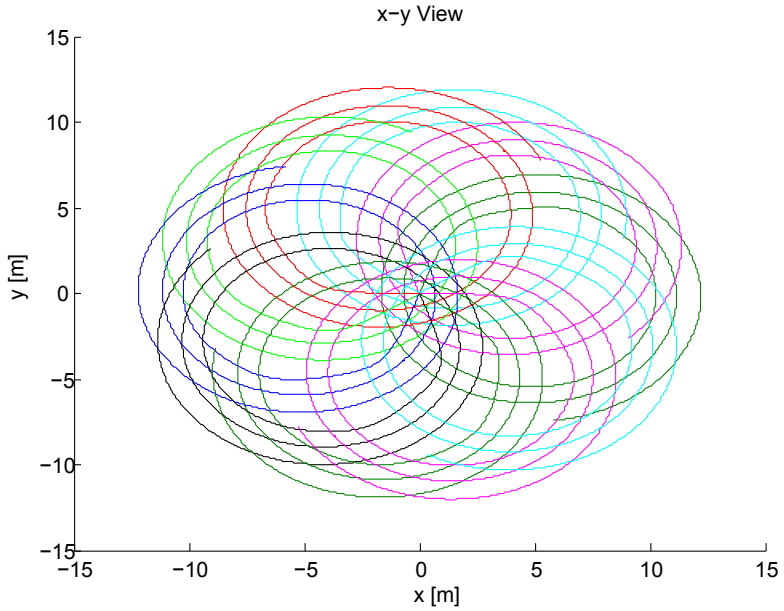

Figure 4.4: Trajectories of the circular formation in the x-y-plane with $K_{nominal}$ after 30 s

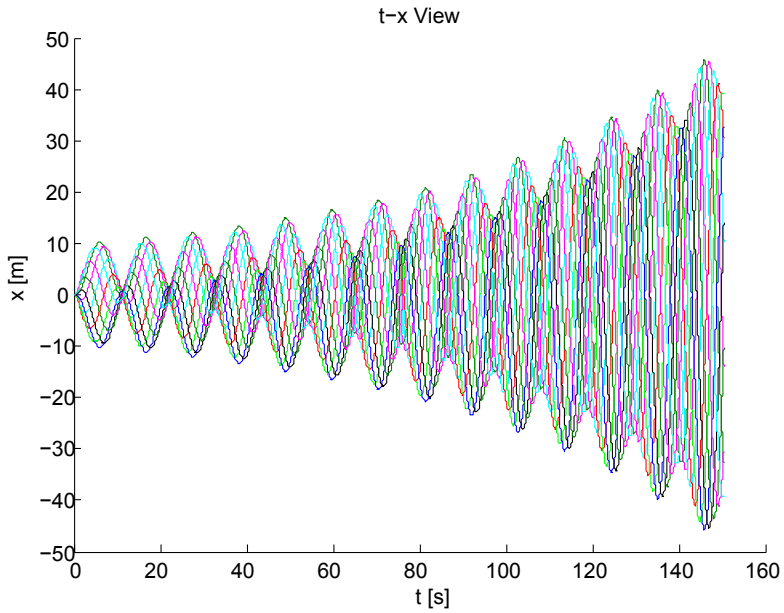

Figure 4.5: x-positions over time of the circular formation with $K_{nominal}$ after 150 s

conformity with the simulator conventions, the ten agents are labeled by 2 to 11. Number 1 always holds the formation reference input, which is not needed for this simulation.

Two different controllers are compared in this section. One is the robust controller K_{robust}

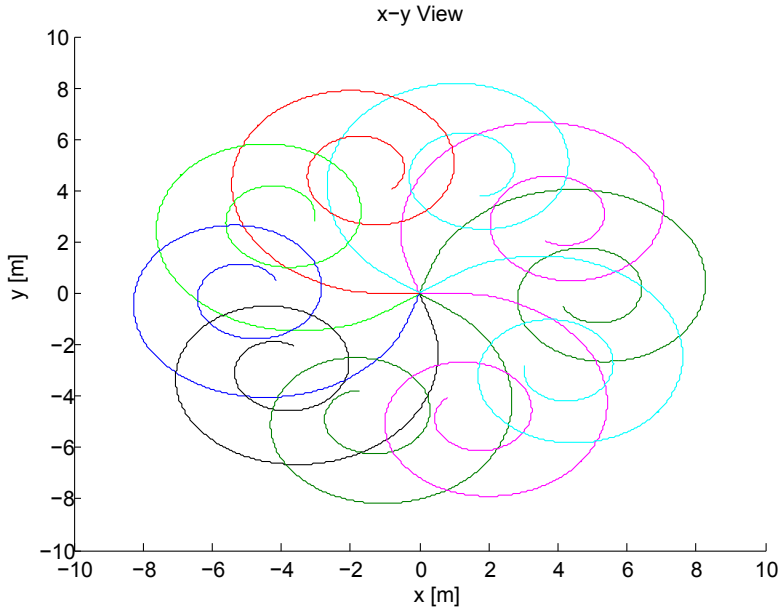

Figure 4.6: Trajectories of the circular formation in the x-y-plane with K_{robust} after 30 s

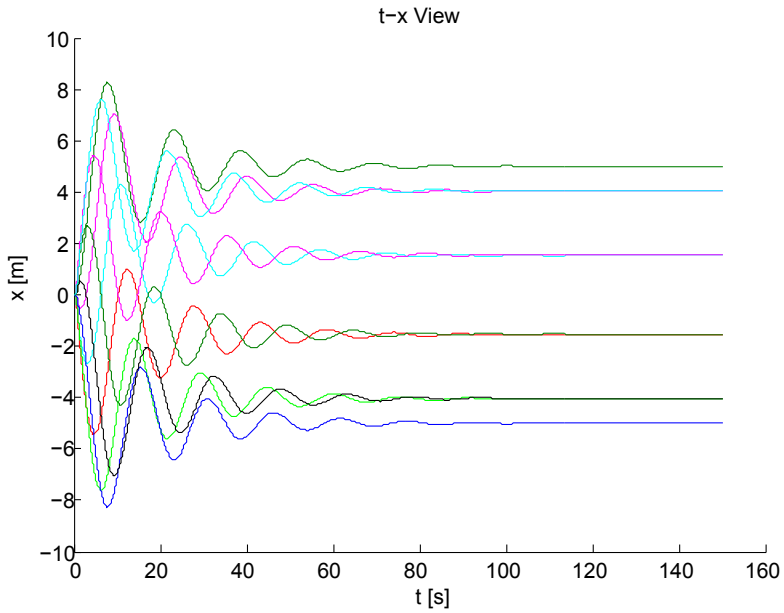

Figure 4.7: x-positions over time of the circular formation with K_{robust} after 150 s

explained in the beginning of this chapter, designed to stabilize any formation of quadrocopters with arbitrary communication topology. The second controller $K_{nominal}$ was de-

signed to stabilize a single quadrocopter. For the simulation, all ten agents start from coincident initial positions and receive reference input to form a circle with radius 5 m.

The trajectories of all agents for the formation with controllers $K_{nominal}$ are drawn in Figure 4.4. Depicted are the first 30 s of the simulation. All agents are moving on spiral paths with increasing diameter, which clearly shows formation instability. For a longer time period of 150 s, the x-positions of all agents are shown in Figure 4.5. Amplitudes of the spiral oscillation grow exponentially and the formation becomes instable. Thus, the controller $K_{nominal}$, which can stabilize single quadrocopters, is not able to stabilize the formation with the underlying circular communication topology.

The second considered controller K_{robust} leads to the trajectories depicted in Figure 4.6. Compared to $K_{nominal}$, the copters move on similar spiral trajectories, but with notably decreasing diameter. This indicates that the quadrocopters converge to distinct final positions within the spiral, the formation is stable. Also Figure 4.7, showing x-positions over time, supports this observation. Amplitudes decrease clearly, and after about 100 s, the quadrocopters have reached their final positions.

Comparison of the two controllers $K_{nominal}$ and K_{robust} shows that formations pose special challenges to controller design. Controllers, which might efficiently stabilize single agents, are not necessarily suitable for formation control in the given framework. They have to achieve two complementary goals, stabilization of a single vehicle and of the formation as a whole.

4.2.2 Time Delay

If it comes to implementations of formation control to real vehicles, an important issue to be considered is transmission time delay. No physical signal can be transmitted delay-free. Especially for formations of flying agents, which might be separated by large distances, time delays have to be considered.

Figure 4.8 shows the trajectories for the formation with K_{robust} and the same circular communication topology in Figure 4.3, but with a communication time delay of 0.4 s. The controller K_{robust}, which is robust for the circular, delay-free communication topology, can not stabilize the formation for the given time delay. Figure 4.9 clearly shows increasing amplitudes, the formation becomes instable. Thus, time delays pose an additional challenge to formation control. The robust controller K_{robust} is not able to stabilize formations with transmission time delays. Strategies have to be found, how to account for communication time delays in formation control.

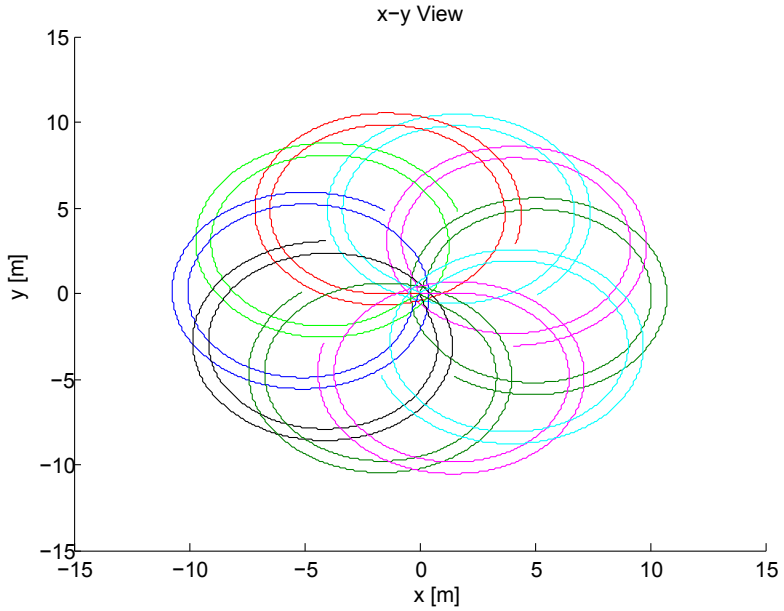

Figure 4.8: Trajectories of the circular formation in the x-y-plane with K_{robust} and a time delay of 0.4 s after 30 s

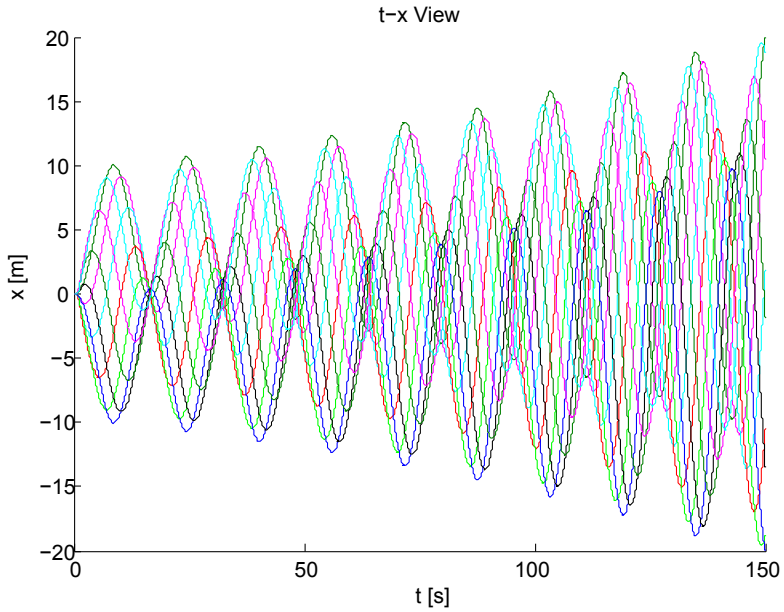

Figure 4.9: x-positions over time of the circular formation with K_{robust} and a time delay of 0.4 s after 150 s

4.2.3 Changing Communication Topology

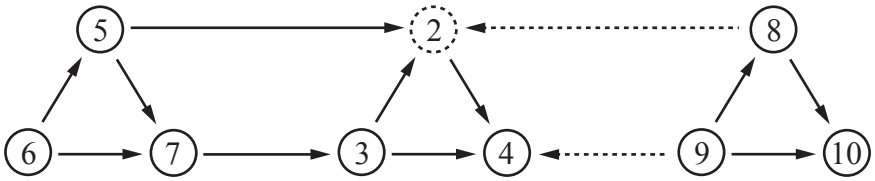

Figure 4.10: Formation of nine agents with three triangles

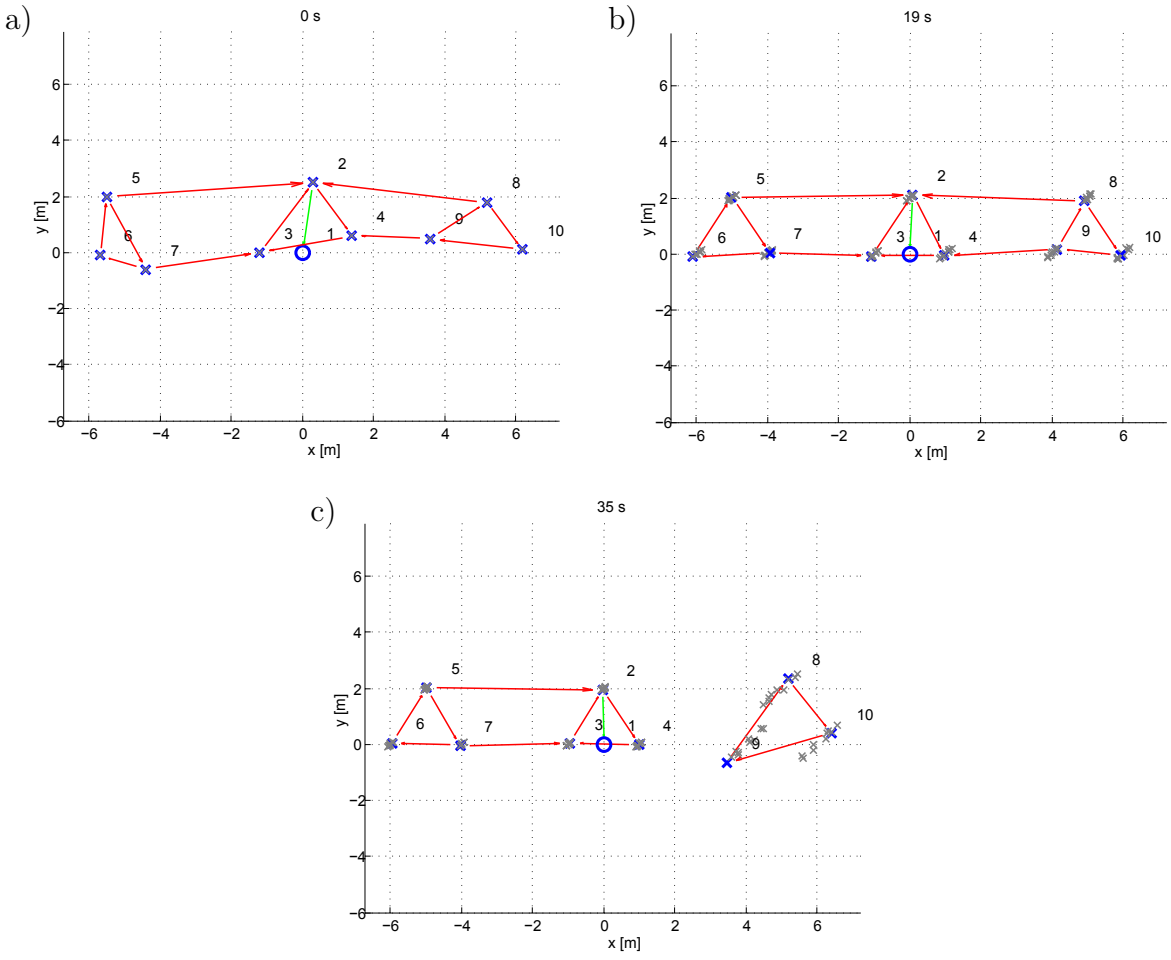

Figure 4.11: Triangle-formation in the x-y-plane, a) initial positions, b) after 19 s, c) after 35 s

In the real world, changes to the communication structure of a formation could be caused by obstacles or perturbation of the signal transmission. Thus, it is important to gain insight into the influence of those changes to formation stability. For the simulation in this section, the formation in Figure 4.10 is chosen. It consists of three triangles, connected

in a line. For demonstration purposes, a non-linear quadrocopter model is implemented for this simulation. $K_{nominal}$, which does not guarantee formation stability, is implemented as a controller.

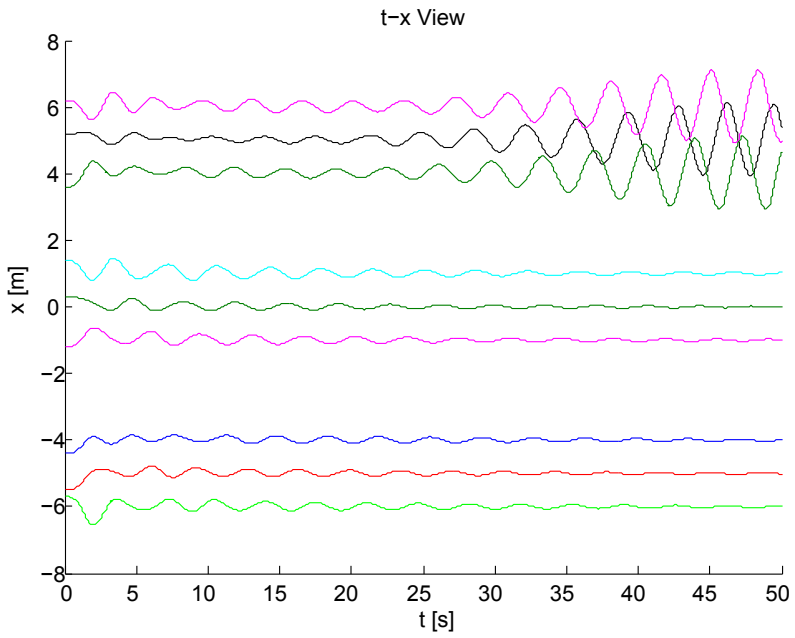

Figure 4.12: x-positions over time of the triangle-formation with $K_{nominal}$ after 50 s

At the beginning of the simulation, all agents are assigned initial positions that differ from their reference position to generate position errors, see Figure 4.11 (a). The controller is able to stabilize the formation with the given communication topology, the agents converge to their specified positions (Figure 4.11 (b)). After 20 s, the communication links between agents 2 and 8 and between 4 and 9 are deleted. As a consequence, the right triangle, which is now disconnected from the remaining formation, starts to oscillate with increasing amplitude (Figure 4.11 (c)) while the remaining formation stays stable. The agent's x-positions over time in Figure 4.12 clearly illustrate that, after a stabilization of the formation during the first 20 s, the changed communication topology leads to instability of part of the formation. This phenomenon emphasizes the necessity for the design of robust controllers as in [PPW09], that stabilize formations independently of changes to the communication topology.

4.3 Tracking

One important property of formations is their tracking capability. Since applications for vehicle formations mostly involve movements of the formation over time, it is significant how well the formation can follow a reference trajectory. This section investigates the influence of controller design (Section 4.3.1), reference input availability (Section 4.3.2), and inter-agent communication (Sections 4.3.3 and 4.3.4) on the tracking performance. The performance measures defined in Section 4.1 are used.

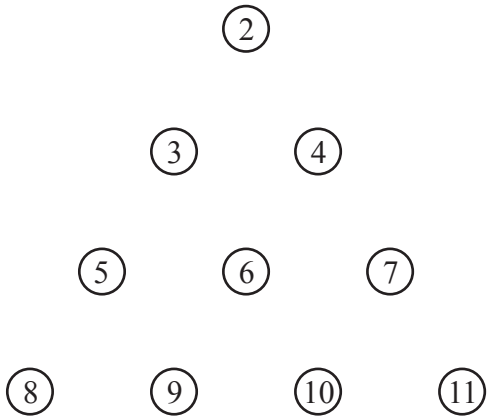

Figure 4.13: Delta-formation with ten agents

As a model formation, a delta-constellation with ten agents is chosen, see Figure 4.13. Again, for conformity with the simulator conventions, the ten agents are labeled by 2 to 11. The number 1 always holds the formation reference input, which is not shown in this figure.

For the reference trajectory, which serves as a dynamic reference signal for the formation, a sine signal in two dimensions,

$$y = 5 \sin(0.25x) \tag{4.7}$$

is used. For simplicity, the z-coordinate is kept constant.

4.3.1 Controller

This section compares two different controllers in their influence on the tracking performance. Both analyses implement the delta-formation in Figure 4.13 with a full communication topology. That is, each agent can receive from each other agent, and each agent can also receive the formation reference input.

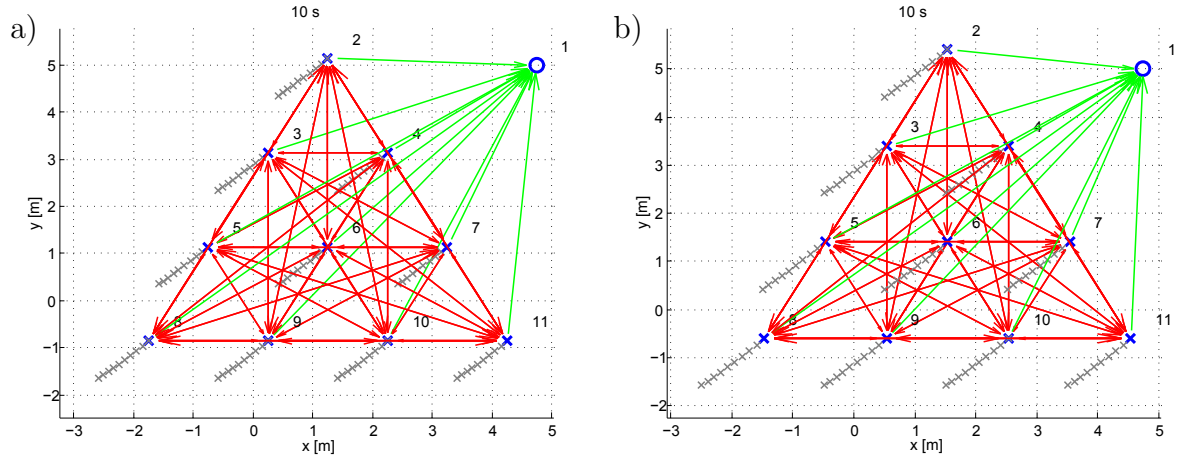

Figure 4.14: Delta-formation in the x-y-plane after 10 s, full communication topology, all agents receive the reference input, a) with K_{robust}, b) with $K_{complete}$

The first controller is the robust controller K_{robust}. It claims to be robustly stable for any given communication topology and number of agents. A snapshot of the formation at 10 s, following the sine trajectory in (4.7), is given in Figure 4.14 (a). The formation shape is perfectly maintained, while it can only follow the reference signal at a notable distance delay (agent 6 should coincide with the reference signal, represented by the blue circle). Following (4.1) and (4.2), the tracking errors for one full sine trajectory are

$$\bar{\epsilon}_{t1} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| = 6.6056 \tag{4.8}$$

$$\bar{\epsilon}_{t2} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \left\| \sum_{i=1}^{10} \left(\boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right) \right\| = 6.6056 \tag{4.9}$$

The formation errors from (4.3) and (4.4) give

$$\begin{aligned} \bar{\epsilon}_{f1} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left\| (\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k}) \right\| \\ &= 6.3772 \times 10^{-15} \end{aligned} \tag{4.10}$$

$$\begin{aligned} \bar{\epsilon}_{f2} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left| \left\| \boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k} \right\| - \left\| \boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k} \right\| \right| \\ &= 4.1210 \times 10^{-15} \end{aligned} \tag{4.11}$$

which are negligible for the given formation.

For the full communication topology, the critical-links ratio is

$$\alpha_l = \frac{0}{55} = 0 \tag{4.12}$$

since no failure of a single link is critical to the formation shape. Likewise

$$\alpha_a = \frac{0}{10} = 0 \tag{4.13}$$

No agent failure endangers the existence of the formation.

The same analysis is carried out with the controller $K_{complete}$, especially developed for a formation of ten agents and full communication topology. Figure 4.14 (b) shows a similar situation compared to the previous simulation for a formation implementing this controller. The formation shape is maintained perfectly, while the reference signal can be followed at a slightly shorter distance. Tracking and formation errors compute as

$$\overline{\epsilon}_{t1} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| = 5.6870 \tag{4.14}$$

$$\overline{\epsilon}_{t2} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \left\| \sum_{i=1}^{10} \left(\boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right) \right\| = 5.6870 \tag{4.15}$$

$$\begin{aligned} \overline{\epsilon}_{f1} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left\| (\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k}) \right\| \\ &= 2.4028 \times 10^{-14} \end{aligned} \tag{4.16}$$

$$\begin{aligned} \overline{\epsilon}_{f2} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left| \left\| \boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k} \right\| - \left\| \boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k} \right\| \right| \\ &= 1.5377 \times 10^{-15} \end{aligned} \tag{4.17}$$

Since the communication topology did not change, α_l and α_a are zero again.

Comparing the tracking errors for both controllers, it concludes that $K_{complete}$ is superior to K_{robust} for the given communication topology. Not surprisingly, different controllers influence the tracking performance of a formation for the same underlying communication topology.

4.3.2 Reference Input Accessibility

In the previous section, all agents had access to the formation reference signal. To analyze the effect of the reference input accessibility on the tracking performance, the same

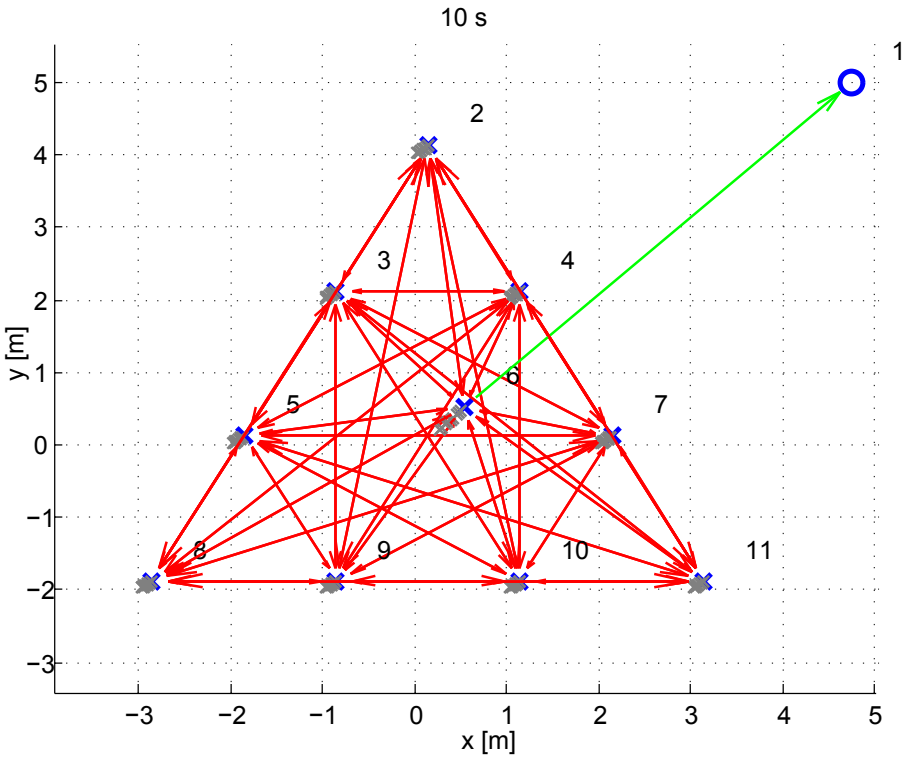

Figure 4.15: Delta-formation in the x-y-plane after 10 s, full communication topology, one agent receives the reference input

simulation with controller $K_{complete}$ has been carried out for only one agent receiving this signal. The snapshot after 10 s in Figure 4.15 demonstrates that the formation's tracking capability is reduced drastically. While the reference signal has proceeded from $[0\ 0]^T$ to $[4.75\ 5]^T$, the formation, initially centered with agent 6 at $[0\ 0]^T$, has only moved slightly. The tracking errors for the whole sine trajectory,

$$\bar{\epsilon}_{t1} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| = 11.8457 \tag{4.18}$$

$$\bar{\epsilon}_{t2} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \left\| \sum_{i=1}^{10} \left(\boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right) \right\| = 11.8457 \tag{4.19}$$

are more than two times larger as for the corresponding formation in Figure 4.14 (b).

The formation errors of

$$\bar{\epsilon}_{f1} = \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left\| (\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k}) \right\|$$
$$= 0.2063 \tag{4.20}$$

$$\bar{\epsilon}_{f2} = \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left| \left\| \boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k} \right\| - \left\| \boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k} \right\| \right|$$
$$= 0.1400 \tag{4.21}$$

show that also the formation shape is maintained less accurately. This is due to a displacement of agent 6 compared to the other agents, see Figure 4.15.

Critical-links and critical-agents ratio give

$$\alpha_l = \frac{1}{46} = 0.0217 \tag{4.22}$$

$$\alpha_a = \frac{1}{10} = 0.1 \tag{4.23}$$

This is due to agent 6, whose failure would be fatal, since it is the only agent receiving the reference signal.

It appears that, with an increasing number of agents being able to receive the reference signal, the ability to maintain the formation shape increases, as well as robustness of the communication topology.

4.3.3 Communication Topology

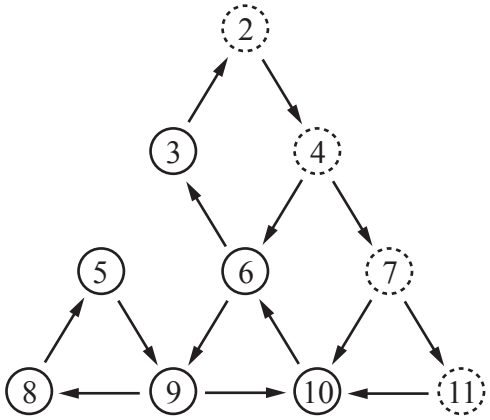

Figure 4.16: Delta-formation with 14 communication links (choice 1), four agents receive the reference input

As a major property of formations, their underlying inter-agent communication topology is expected to influence tracking performance. Thus, this section focuses on two delta-formations with ten agents as in Figure 4.13. Both formations have the same controllers

$K_{complete}$, the same number of communication links, and four agents each can receive the reference input signal.

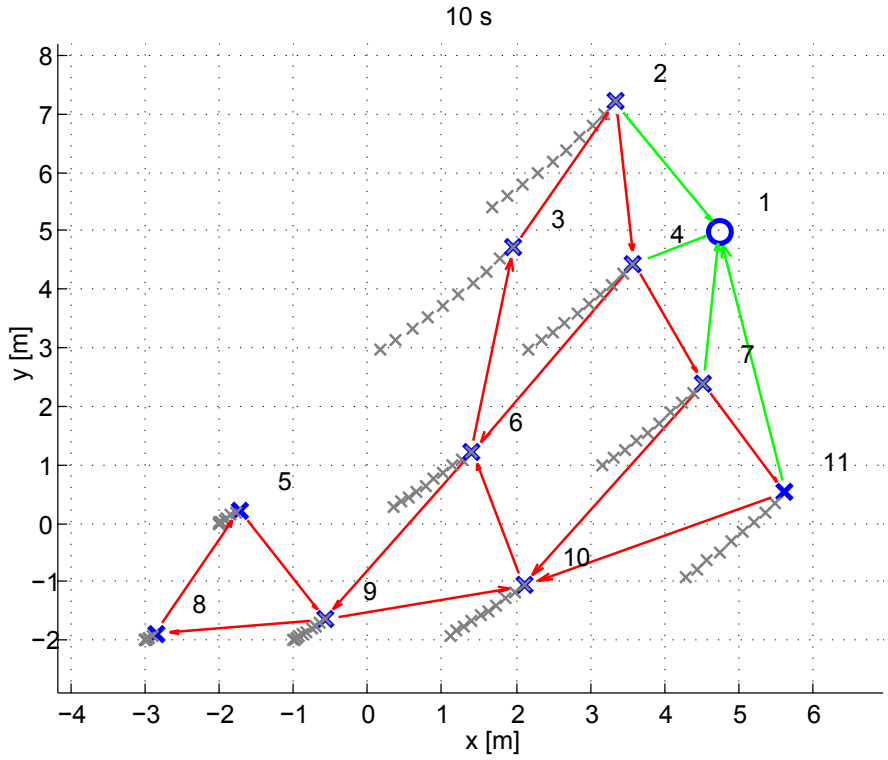

Figure 4.17: Delta-formation in the x-y-plane after 10 s, 14 communication links (choice 1), four agents receive the reference input

The first proposed communication topology is depicted in Figure 4.16. 14 inter-agent communication links are distributed among the agents. Agents 2, 4, 7, and 11 can access the reference signal and are drawn with a dashed line. Figure 4.17 illustrates that, after 10 s, the formation is stretched compared to its specified shape. Agents 5, 8, and 9 move much slower than the other agents, illustrated by the grey trails following each agent. For one full sine trajectory, the tracking and formation errors are

$$\bar{\epsilon}_{t1} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| = 5.6512 \tag{4.24}$$

$$\bar{\epsilon}_{t2} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \left\| \sum_{i=1}^{10} \left(\boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right) \right\| = 5.6360 \tag{4.25}$$

$$\bar{\epsilon}_{f1} = \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left\| (\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k}) \right\|$$
$$= 2.8456 \tag{4.26}$$

$$\bar{\epsilon}_{f2} = \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left| \left\| \boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k} \right\| - \left\| \boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k} \right\| \right|$$
$$= 2.0533 \tag{4.27}$$

and the ratios compute as

$$\alpha_l = \frac{6}{18} = 0.33 \tag{4.28}$$

$$\alpha_a = \frac{5}{10} = 0.5 \tag{4.29}$$

This means, 1/3 of all communication links and 1/2 of all agents are critical to maintain the formation shape.

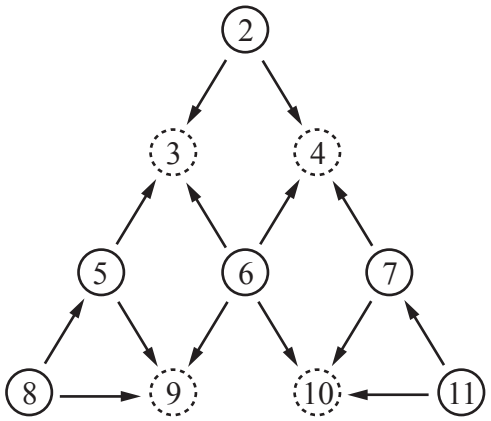

Figure 4.18: Delta-formation with 14 communication links (choice 2), four agents receive the reference input

The same formation with another communication topology is depicted in Figure 4.18. Again, four agents can receive the reference input and 14 communication links exist. Figure 4.19 shows that, after 10 s, the formation can maintain its shape much better than the one in Figure 4.17. The tracking errors give

$$\bar{\epsilon}_{t1} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| = 1.3021 \tag{4.30}$$

$$\overline{\epsilon}_{t2} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \left\| \sum_{i=1}^{10} \left(\boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right) \right\| = 1.3016 \tag{4.31}$$

and the formation errors are

$$\begin{aligned} \overline{\epsilon}_{f1} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left\| (\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k}) \right\| \\ &= 0.5323 \end{aligned} \tag{4.32}$$

$$\begin{aligned} \overline{\epsilon}_{f2} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left| \left\| \boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k} \right\| - \left\| \boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k} \right\| \right| \\ &= 0.3374 \end{aligned} \tag{4.33}$$

The critical-links and -agents ratios give

$$\alpha_l = \frac{4}{18} = 0.22 \tag{4.34}$$

for the four reference signal links of agents 3, 4, 9, 10, and

$$\alpha_a = \frac{0}{10} = 0 \tag{4.35}$$

$\overline{\epsilon}_{t1}$ and $\overline{\epsilon}_{t2}$ are about four times smaller compared to the first formation, $\overline{\epsilon}_{f1}$ and $\overline{\epsilon}_{f2}$ as much as six times smaller. Additionally, α_l and α_a show that the second formation is more robust to communication link and agent failure.

As a conclusion, the communication structure has a large influence on a formation's tracking performance. Even for an equal number of communication links, performance varies significantly.

Furthermore, also the number of inter-agent communication links heavily influences tracking performance. While the formation in Section 4.3.1 with full communication topology maintains its shape perfectly, the communication topology in Figure 4.18 is superior in keeping a small position error. This poses a trade-off between formation shape and tracking performance.

As a special case, the communication topology in Figure 4.20 is analyzed. Each agent only receives from agent 6, which in turn is the only agent that can access the external reference input. This gives agent 6 the role of a single leader in the formation.

The formation flight after 10 s is shown in Figure 4.21. The formation shape is very well maintained, only agent 6 is off-centered compared to its neighbors. In addition, the

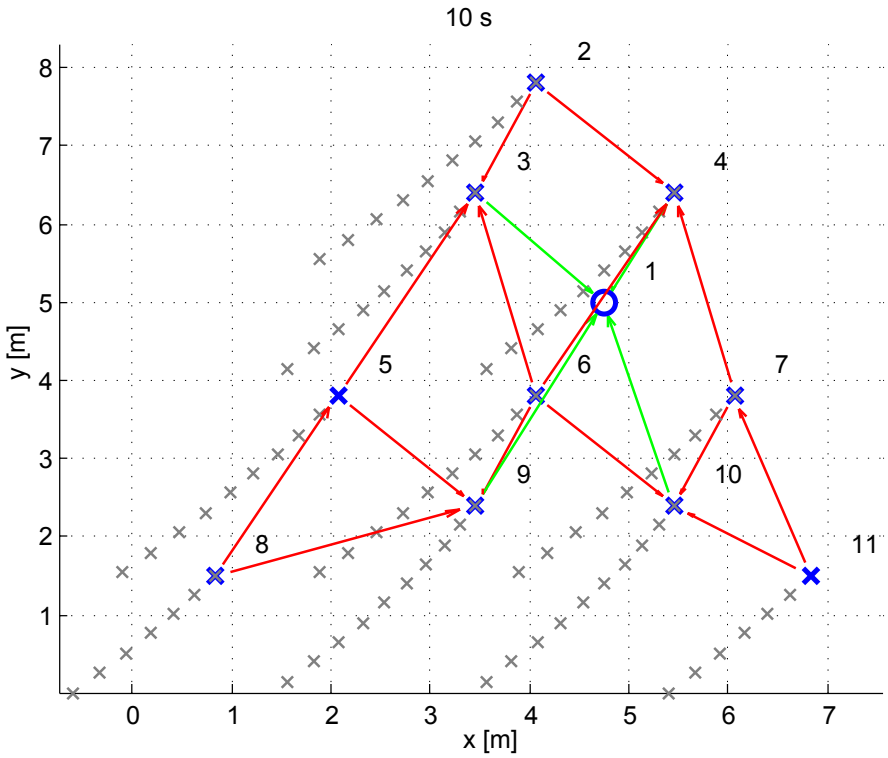

Figure 4.19: Delta-formation in the x-y-plane after 10 s, 14 communication links (choice 2), four agents receive the reference input

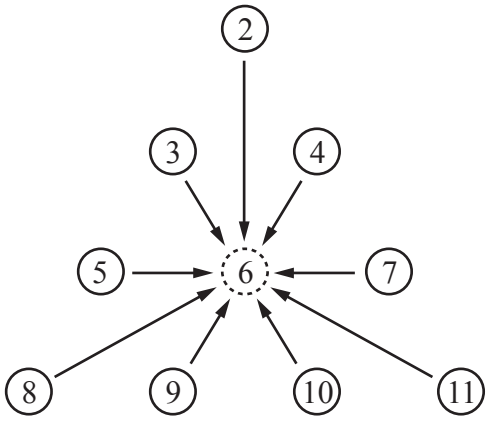

Figure 4.20: Delta-formation with star-communication topology, one agent receives the reference input

formation follows the reference trajectory with very small delay. The errors give

$$\bar{\epsilon}_{t1} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \left\| \boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right\| = 2.1489 \tag{4.36}$$

$$\bar{\epsilon}_{t2} = \frac{1}{10 \cdot 5026} \sum_{k=1}^{5026} \left\| \sum_{i=1}^{10} \left(\boldsymbol{y}_{i,ref,k} - \boldsymbol{y}_{i,k} \right) \right\| = 2.1489 \tag{4.37}$$

$$\begin{aligned} \bar{\epsilon}_{f1} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left\| (\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k}) \right\| \\ &= 0.1452 \end{aligned} \tag{4.38}$$

$$\begin{aligned} \bar{\epsilon}_{f2} &= \frac{2}{10(10-1)5026} \sum_{k=1}^{5026} \sum_{i=1}^{10} \sum_{j=i+1}^{10} \left| \left\| \boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k} \right\| - \left\| \boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k} \right\| \right| \\ &= 0.0919 \end{aligned} \tag{4.39}$$

$$\alpha_l = \frac{10}{10} = 1 \tag{4.40}$$

$$\alpha_a = \frac{0}{10} = 0.1 \tag{4.41}$$

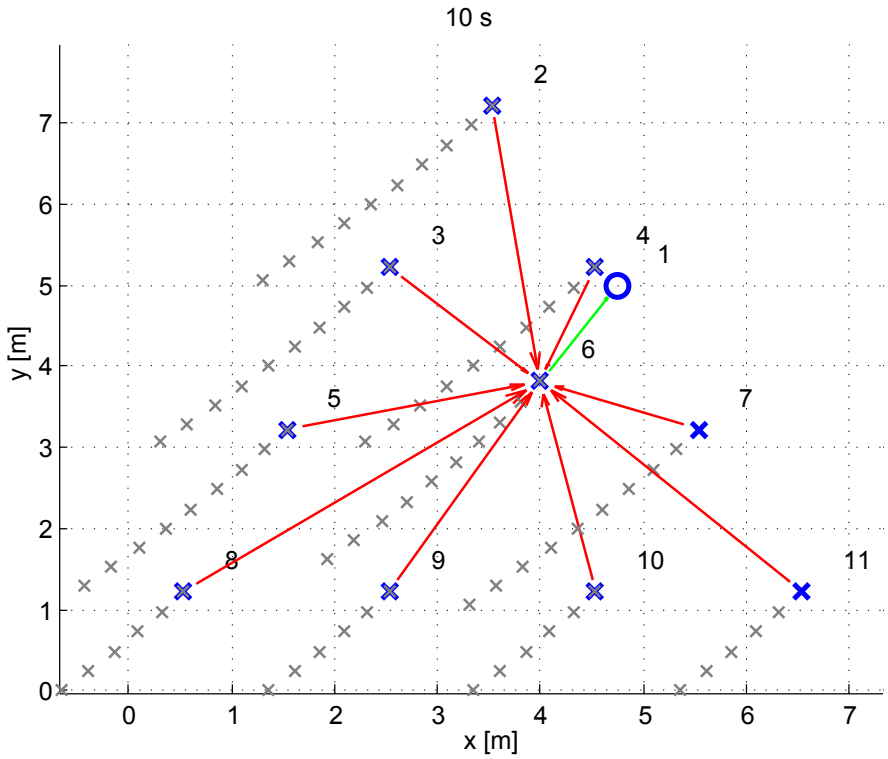

Figure 4.21: Delta-formation in the x-y-plane after 10 s, star communication topology, one agent receives the reference input

It is obvious that this communication topology combines very good tracking behavior with very good shape-maintaining. However, a failure of agent 6 would be fatal and

leave the formation uncontrolled. Also, every link failure in the formation would lead to a disconnected agent, see (4.40). A failure of the reference signal link to agent 6 would affect the whole formation. Thus, system robustness is reduced drastically with this reduced single-leader communication topology.

4.3.4 Limited Communication Range

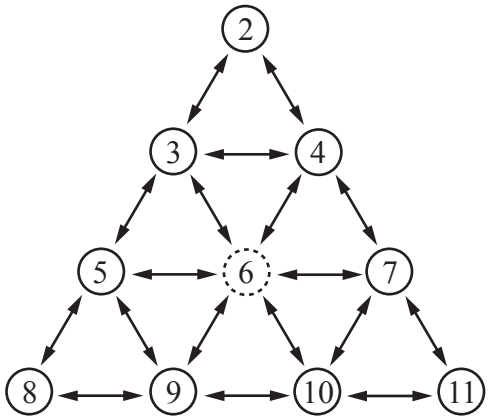

Figure 4.22: Delta-formation with limited communication range, one agent receives the reference input

Inspired by technical limitations in the real world, one exemplary simulation with limited communication range is presented in this section. Depending on the implementation of communication links between real agents, restrictions to the communication range could be a thread to formation performance. Under water, for example, signal transmission is challenging compared to radio frequency transmission in air. Also realizations making use of optical sensors have a limited transmission range.

For this simulation, a full communication topology is defined, that is, each agent can receive from any other agent. However, by setting a very restrictive transmission range of 3 m, the initial communication topology is reduced to the one depicted in Figure 4.22. In the undeformed formation, each agent can receive from their closest neighbors.

Since only agent 6 can receive the reference signal, the formation shows poor performance in tracking the sine signal in (4.7). Figure 4.23 (a) shows that, after 20 s, agent 6 is already separated too far from agents 9 and 10 to maintain signal transmission. After 30 s, more links are broken (Figure 4.23 (b)), which leads to a disruption of the formation after 35 s (Figure 4.23 (c)). Since none of the agents 7-11 can receive the reference signal,

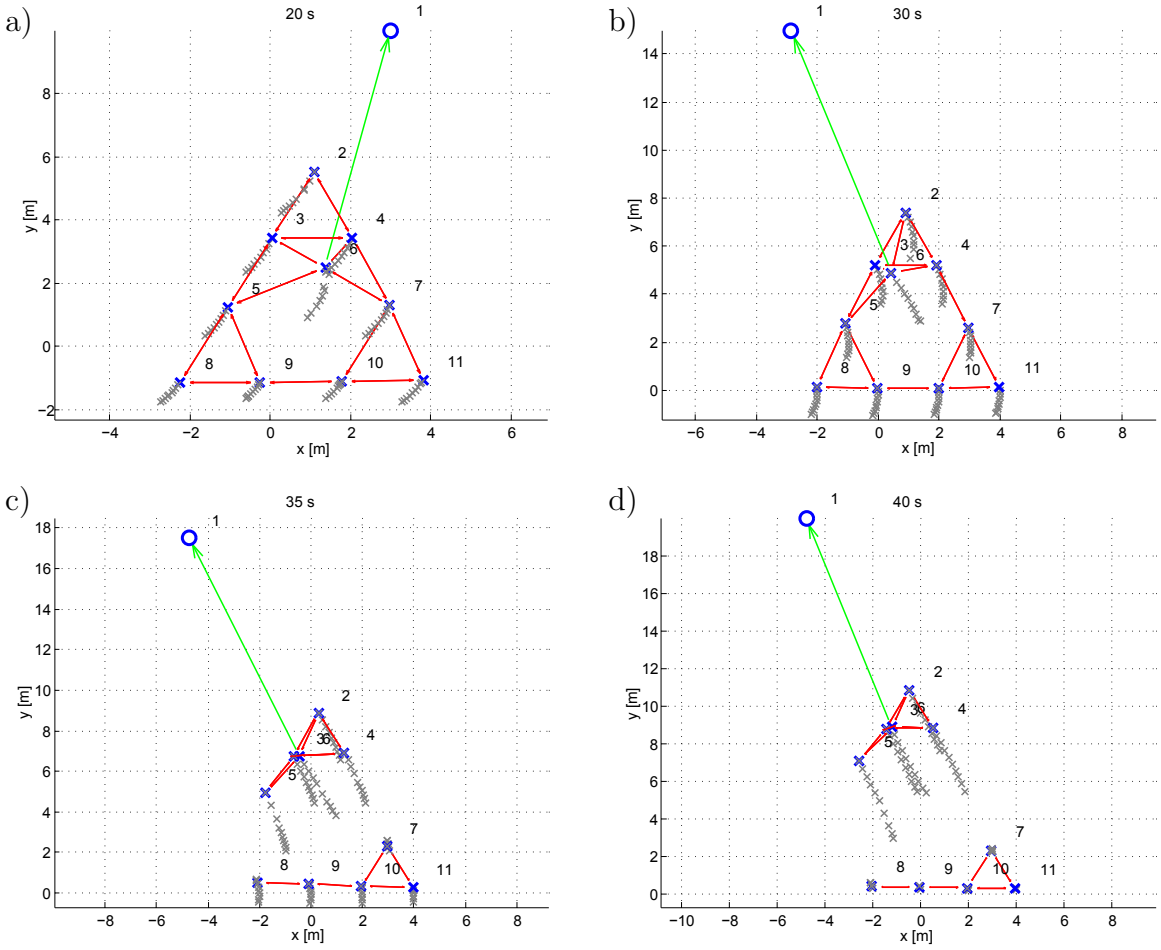

Figure 4.23: Delta-formation in the x-y-plane, limited communication range, one agent receives the reference input, a) after 20 s, b) after 30 s, c) after 35 s, d) after 40 s

this group is left behind by the remaining formation and stops moving (Figure 4.23 (d)). The controller $K_{complete}$ can stabilize the two partial formations.

This simulation illustrates that restrictions such as a limited communication range pose threads to formation control. For real-world implementations, secure signal transmission is a major requirement to guarantee formation performance and stability. It has to be ensured that, in case of communication failures, the controllers can stabilize the sub-formations.

4.4 Directed Vectors versus Distances

Section 2.3 explains two ways to define a formation. All simulations in this chapter up to this point implement directed vectors as reference input. That is, each agent knows which x-, $y-$, and $z-$position it has to maintain relative to the other agents (if there is a communication link between those agents). A second strategy to define the formation shape is explained in Section 2.3.2. Instead of directed vectors, only undirected distances are defined for each two agents. The two different approaches are compared in this section.

A control law for point agents in the plane, where the formation is defined by undirected distances, is derived in [YADF09]. Since the proposed control law is non-linear, depends on the agents' positions, and is only valid for small motions, it has not been implemented for this simulation. To allow for comparison with the control framework of [FM04], using the controllers from previous analyses in this chapter, the undirected distances reference input is adapted to this framework. The weighted sum of errors for each agent still computes as in (2.8). The $\boldsymbol{e}_{ij}$ are now modified to include distances,

$$\boldsymbol{e}_{ij} = \frac{\boldsymbol{y}_i - \boldsymbol{y}_j}{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|} \left(d_{ij} - \|\boldsymbol{y}_i - \boldsymbol{y}_j\|\right) \tag{4.42}$$

with d_{ij} being the reference distance between agents i and j. Note that this adaption follows a solely intuitive understanding and does not found on a theoretic background. It's only purpose is to compare the two given ways of defining inter-agent distances.

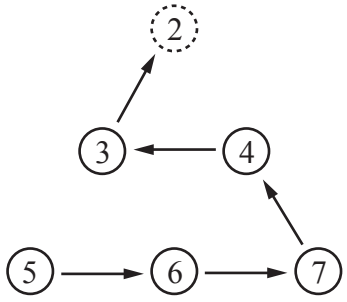

Figure 4.24: Delta-formation with six agents, minimal communication topology

For the first simulation, the communication topology in Figure 4.24 is defined. This arrangement of agents also illustrates the desired formation shape. With each agent only receiving from one neighbor, it is the most reduced communication structure to define a coherent formation. K_{robust} is used as controller.

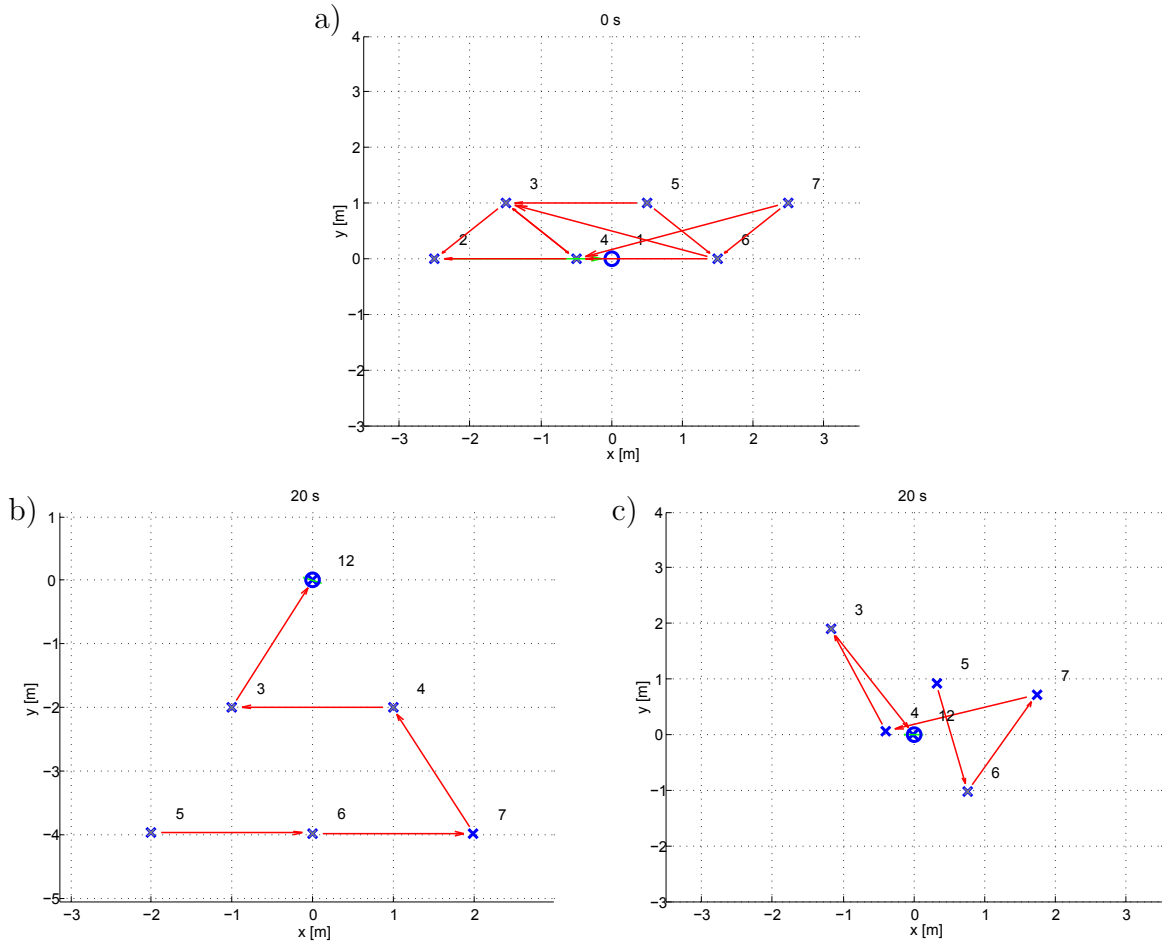

Figure 4.25: Delta-formation in the x-y-plane, minimal communication, one agent receives the reference input, a) initial positions, b) directed vectors as reference input, after 20 s, c) undirected distances as reference input, after 20 s

The initial positions of all agents are shown in Figure 4.25 (a). For directed vectors as reference input, the formation after 20 s is plotted in Figure 4.25 (b). As expected, K_{robust}, which is robust for any given communication topology of directed vectors, can stabilize the formation in its specified shape. The formation errors, for only one timestep at 20 s, confirm this observation:

$$\begin{aligned}\bar{\epsilon}_{f1} &= \frac{2}{6(6-1)1} \sum_{k=2000}^{2001} \sum_{i=1}^{6} \sum_{j=i+1}^{6} \left\| (\boldsymbol{y}_{j,ref,k} - \boldsymbol{y}_{i,ref,k}) - (\boldsymbol{y}_{j,k} - \boldsymbol{y}_{i,k}) \right\| \\ &= 0.0141 \end{aligned} \tag{4.43}$$

$$\begin{aligned}\overline{\epsilon}_{f2} &= \frac{2}{6(6-1)1}\sum_{k=2000}^{2001}\sum_{i=1}^{6}\sum_{j=i+1}^{6}\Big|\big\|\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k}\big\|-\big\|\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k}\big\|\Big| \\ &= 0.0114 \end{aligned} \tag{4.44}$$

The same simulation is conducted for undirected distance reference inputs, with controller K_{robust} and the error formula as defined in (4.42). Not surprisingly, the formation is not able to take the desired shape, since it is geometrically under-determined. However, it stabilizes in a random shape with the specified distances between the agents as shown in Figure 4.25 (b). The formation errors are

$$\begin{aligned}\overline{\epsilon}_{f1} &= \frac{2}{6(6-1)1}\sum_{k=2000}^{2001}\sum_{i=1}^{6}\sum_{j=i+1}^{6}\big\|(\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k})-(\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k})\big\| \\ &= 2.9133 \end{aligned} \tag{4.45}$$

$$\begin{aligned}\overline{\epsilon}_{f2} &= \frac{2}{6(6-1)1}\sum_{k=2000}^{2001}\sum_{i=1}^{6}\sum_{j=i+1}^{6}\Big|\big\|\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k}\big\|-\big\|\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k}\big\|\Big| \\ &= 1.2336 \end{aligned} \tag{4.46}$$

While for directed vectors, the formation takes its desired shape even for the reduced communication topology in Figure 4.24, specifying distances does not provide enough information to define a unique formation for the given communication topology.

To define a communication topology that is suitable for distance reference inputs, the concept of minimally persistent graphs as explained in Section 2.3.2 is applied. A corresponding communication structure is defined in Figure 4.26.

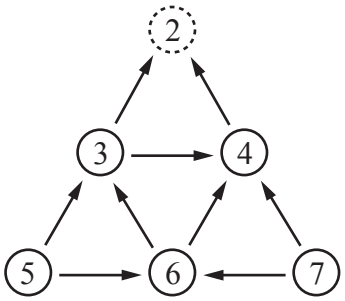

Figure 4.26: Delta-formation with six agents, minimally persistent communication topology

Starting with the initial conditions in Figure 4.27 (a), which are the same as for the previous simulation, the formation with K_{robust} and distance reference inputs can now take the desired delta-shape, see Figure 4.27 (c). However, the formation is rotated compared to the result for directed vector references. This phenomenon illustrates the additional DoF of minimally persistent formations in the plane. Agent 4 has the role of the *first follower* (see Section 2.3.2). It only maintains the distance to agent 2 and can rotate the whole formation around agent 2 in its second DoF.

The formation errors for this setup after 20 s are

$$\begin{aligned}\bar{\epsilon}_{f1} &= \frac{2}{6(6-1)1}\sum_{k=2000}^{2001}\sum_{i=1}^{6}\sum_{j=i+1}^{6}\left\|(\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k})-(\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k})\right\| \\ &= 3.7655 \end{aligned} \tag{4.47}$$

$$\begin{aligned}\bar{\epsilon}_{f2} &= \frac{2}{6(6-1)1}\sum_{k=2000}^{2001}\sum_{i=1}^{6}\sum_{j=i+1}^{6}\left|\left\|\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k}\right\|-\left\|\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k}\right\|\right| \\ &= 0.1830 \end{aligned} \tag{4.48}$$

Those very different formation errors emphasize that the formation is well maintained in terms of the inter-agent distances, expressed by $\bar{\epsilon}_{f2}$, but rotated. Thus, the directed distances, taken into account by $\bar{\epsilon}_{f1}$, are not correct, which leads to a high error $\bar{\epsilon}_{f1}$. A more detailed comparison of both formation errors is given in Section 4.5.

The same analysis has been carried out with slightly different initial conditions, as depicted in Figure 4.27 (b). After 20 s, the formation has stabilized in a different shape (Figure 4.27 (d)). The formation errors give

$$\begin{aligned}\bar{\epsilon}_{f1} &= \frac{2}{6(6-1)1}\sum_{k=2000}^{2001}\sum_{i=1}^{6}\sum_{j=i+1}^{6}\left\|(\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k})-(\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k})\right\| \\ &= 2.9833 \end{aligned} \tag{4.49}$$

$$\begin{aligned}\bar{\epsilon}_{f2} &= \frac{2}{6(6-1)1}\sum_{k=2000}^{2001}\sum_{i=1}^{6}\sum_{j=i+1}^{6}\left|\left\|\boldsymbol{y}_{j,ref,k}-\boldsymbol{y}_{i,ref,k}\right\|-\left\|\boldsymbol{y}_{j,k}-\boldsymbol{y}_{i,k}\right\|\right| \\ &= 0.6645 \end{aligned} \tag{4.50}$$

$\bar{\epsilon}_{f2}$ is larger compared to the previous simulation, since the formation does not completely take its specified shape. Surprisingly, $\bar{\epsilon}_{f1}$ is smaller. This is due the fact, that some agents are closer to their desired final position, compared to the previous analysis.

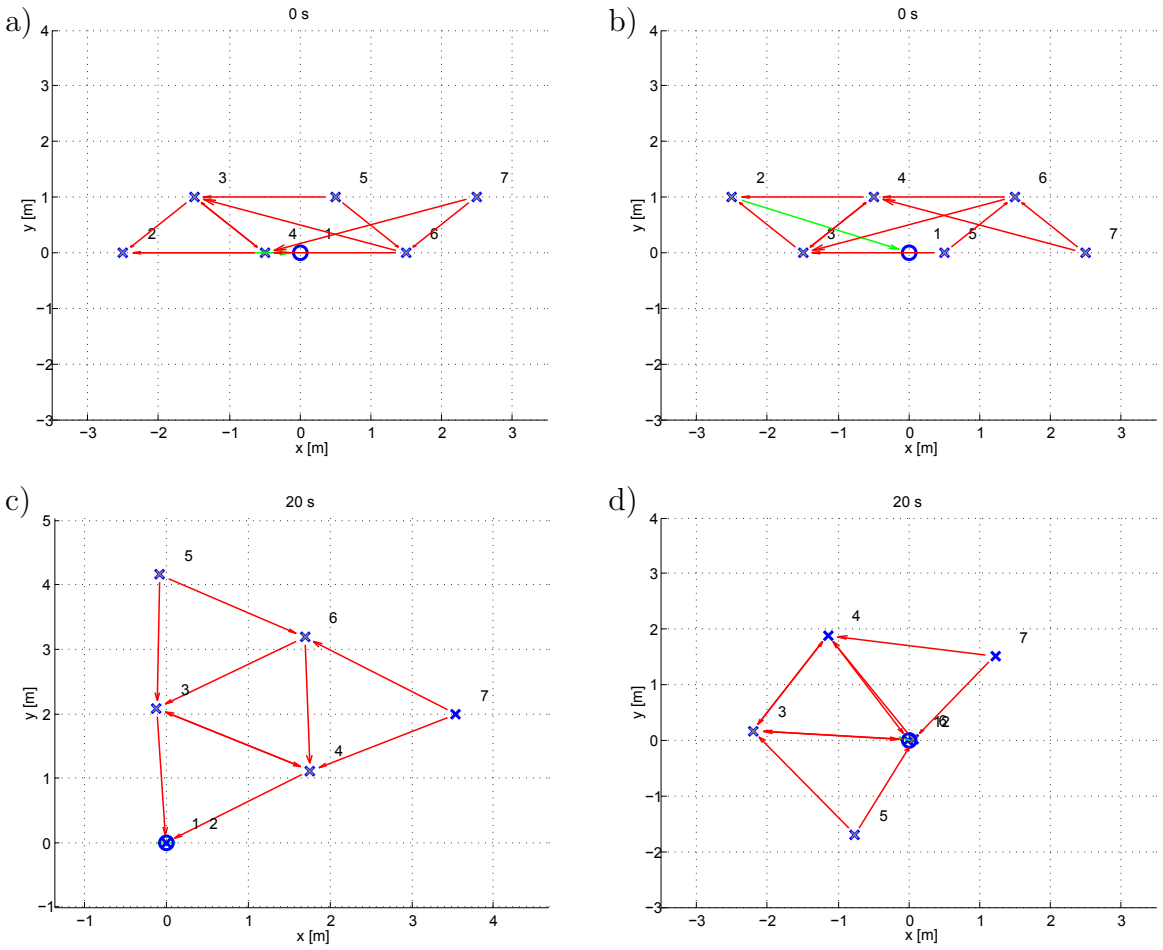

Figure 4.27: Delta-formation in the x-y-plane, undirected vector as reference input, minimal persistent communication, one agent receives the reference input, a) initial positions 1, b) initial positions 2, c) after 20 s with initial conditions 1, d) after 20 s with initial conditions 2

Comparing the last two simulations illustrates a significant downside of undirected distance reference inputs. Although the graph in Figure 4.26 is rigid, the formation does not necessarily take its desired shape for large motions and initial conditions, which are not close to the specified final positions. Thus, further requirements to the communication topology have to be formulated to ensure a rigid *and* unique formation shape.

4.5 Evaluation of Performance Measures

Several performance measures have been defined in Section 4.1 and were used to evaluate different simulation results in the previous sections. In the following, these measures are reviewed and analyzed with respect to their significance and applicability.

Tracking Error The tracking error aims at giving a measurable dimension that indicates how well a formation tracks a moving reference signal, or follows a reference trajectory respectively.

Both tracking errors defined in Section 4.1 provide equal results for almost every analysis carried out in this chapter. Slight differences occur for the formations in Section 4.3.3. $\overline{\epsilon}_{t2}$ gives a slightly lower error than $\overline{\epsilon}_{t1}$. Those differences originate from the fact that $\overline{\epsilon}_{t2}$ considers the tracking performance of the geometrical center of a formation, while $\overline{\epsilon}_{t1}$ compares every agent separately. That is, a distorted formation shape influences $\overline{\epsilon}_{t1}$, even if the formation as a whole tracks its reference signal very well. Also rotations of a formation relative to its reference signal influence $\overline{\epsilon}_{t1}$, but not $\overline{\epsilon}_{t2}$.

Hence, $\overline{\epsilon}_{t2}$ is considered to be a more precise measure, if only the tracking performance, separated from deformations or rotations of the formation, is to be analyzed. $\overline{\epsilon}_{t1}$ provides a measure to incorporate the formation shape and orientation into the tracking error.

Formation Error Independently from a formation's position, the formation errors should evaluate how well a formation maintains its specified shape. Thus, they shall provide a numerical measure to support the visual observation, if a formation holds its specified configuration.

$\overline{\epsilon}_{f1}$ and $\overline{\epsilon}_{f2}$ lead to similar results for very well-shaped formations in Section 4.3.1. For the formations in Sections 4.3.2 and 4.3.3, the difference between both errors becomes evident. While $\overline{\epsilon}_{f2}$ only considers the undirected distance between each two agents, $\overline{\epsilon}_{f1}$ includes the difference between the directed vectors of agent positions and reference values. Thus, if an agent moves orthogonal to another agent, as for example agent 6 compared to agent 2 in Figure 4.20, $\overline{\epsilon}_{f1}$ accounts for this deviation. $\overline{\epsilon}_{f2}$ does not consider this orthogonal movement, as long as the distance between both agents is maintained. Thus, $\overline{\epsilon}_{f2}$ always results in a smaller error than $\overline{\epsilon}_{f1}$.

The different meaning of $\overline{\epsilon}_{f1}$ and $\overline{\epsilon}_{f2}$ becomes obvious in Section 4.4. The formation in Figure 4.27 (c) has its desired shape, but is rotated by a large angle from its original orientation. $\overline{\epsilon}_{f1}$ leads to a large error, while $\overline{\epsilon}_{f2}$ is almost negligible.

Both $\overline{\epsilon}_{f1}$ and $\overline{\epsilon}_{f2}$ provide measures for the formation error, but with significantly different meaning. While $\overline{\epsilon}_{f1}$ includes directed vectors, it can be used conveniently to evaluate a formation's shape *and* orientation. If only the shape, independently from any overall rotation of the formation, should be analyzed, $\overline{\epsilon}_{f2}$ provides a powerful measure.

Critical-Links Ratio As a measure for the robustness of a communication topology, the critical-links ratio was defined. Investigating the robustness of a certain communication structure is an important extension to the performance criteria for formations.

A non-zero critical-links ratio is an indicator that the formation might be at risk, as soon as one communication link breaks. Also, this ratio illustrates that formations with very good tracking performance, as the star formation in Figure 4.21, might have a reduced robustness compared to other constellations. For this example, a critical-links ratio of 1 indicates that any link failure destroys the specified formation shape.

The critical-links ratio is a simple but efficient way to get a first insight into the robustness of a formation's communication topology. The ratio could be split into separate measures for the inter-agent communication and the reference-input communication.

Critical-Agents Ratio Next to communication link failures, real-world formations might also be exposed to agent failures. The critical-agents ratio indicates how many agents are critical to the existence of the remaining formation.

Analogously to the critical-links ratio, a non-zero critical-agents ratio indicates that a formation might be at risk. Simultaneously this means that agents in the formation have a role as a (partial) leader, that is, they are followed by at least one other agent, which only receives from this particular agent. However, the formations in Figures 4.16 and 4.20 show that this ratio does not evaluate the fatality of an agent failure. While for the first formation, the critical-agents ratio is $\alpha_a = 0.5$, most agent failures would only affect one neighbor. For the star formation, $\alpha_a = 0.1$, the only critical agent is number 6. But since this agent is a single leader, the whole formation shape would be destroyed by its failure.

The critical-agents ratio serves as a first indicator, saying if failure of one agent might set the remaining formation at risk. It does not include how severe this impact is, which would be a valuable future extension.

Chapter 5

Conclusions and Outlook

5.1 Thesis Summary

Simulator In this work, a simulation environment for formation control of MAS has been developed. The object-oriented design and declarative naming convention provide a framework that is flexible, intuitively understandable, and easily extendable. Agents and their dynamics and controllers are implemented as separate classes, which allows for simulations with different controllers and models within one formation. The GUI provides a convenient interface to run and visualize simulations. Complex scenarios, including adding or deleting of agents, changes of the communication topology, communication obstacles, and time-depending formation reference inputs can be defined by the user in separate input files.

Simulation and Analysis To validate the simulation environment, various test simulations have been carried out, which illustrate the influence of controller design, structure and changes of the communication topology, and reference input on formation stability and performance. The necessity for controllers, which are robust to changes of the communication structure, and which can handle transmission time delays, has been demonstrated. It was shown that the communication topology has a large influence on tracking performance, and that performance and robustness can be contradictory attributes that pose a trade-off in the design of MAS formations in the given framework.

Performance Measures To analyze the tracking capability of formations, several performance measures were introduced. The proposed tracking errors and formation errors

provide the possibility to measure how well a formation can track a reference trajectory, and how well it can maintain its formation shape. The two ratios given to measure robustness are basic indicators to evaluate the reliability of a communication topology.

5.2 Outlook and Future Work

Simulator The proposed simulation environment is meant to be a starting point, with various possibilities for future extensions. While constant transmission time delays have already been implemented, real-world agents would experience delays that depend on the distance between two agents. Thus, distance-depending time delays could be investigated and included into the simulator.
The formation control framework used to test and validate the simulator uses directed vectors as reference input. Undirected distances have been implemented as a second approach to define formations. However, no controllers designed for this case have been tested yet. A field for future extension would be the derivation of suitable control laws and their implementation in the simulator.
While all simulations in this thesis analyze linear controllers, the modular structure of the simulator allows for straightforward implementation of non-linear controllers as well. Thus, classes for non-linear control laws could be derived and tested.

Performance Measures The performance measures defined in this thesis proved to be a good support to analyze different scenarios. However, they just represent a limited selection of possible criteria. Especially for the robustness of communication topologies, the proposed measures could be extended and more advanced criteria could be derived.

Bibliography

[BA98] T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[BPR99] F. Bellifemine, A. Poggi, and G. Rimassa. Jade - a FIPA-compliant agent framework. *Proceedings of the Practical Applications of Intelligent Agents*, 1999.

[BRJ05] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 2. edition, 2005.

[Cle06] M. Clement. Development of a Matlab orbit propagator for formation flying applications. Master thesis, Cranfield University, School of Engineering, 2006.

[CPR01] P. R. Chandler, M. Pachter, and S. Rasmussen. Uav cooperative control. *Proceedings of the American Control Conference*, pages 50–55, 2001.

[Cyb09] http://www.cyberbotics.com/products/webots/index.html, 13.10.2009. Cyberbotics Homepage.

[dCLF93] D. de Champeaux, D. Lea, and P. Faure. *Object-Oriented System Development*. Addison-Wesley Publishing Company, 1. edition, 1993.

[DFK+02] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825, 2002.

[Die05] R. Diestel. *Graph Theory*. Number 173 in Graduate Texts in Mathematics. Springer-Verlag, 3. edition, 2005.

[Eck06] B. Eckel. *Thinking in Java*. Prentice Hall, 4. edition, 2006.

[Fax02] J. A. Fax. *Optimal and Cooperative Control of Vehicle Formations*. PhD thesis, California Institute of Technology, 2002.

[FM04] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, 49(9):1465–1476, 2004.

[Fou09] J. Fournier. *Graph Theory and Applications*. ISTE Ltd, London and John Wiley & Sons, Inc., Hoboken, 2009.

[GR04] C. Godsil and G. Royle. *Algebraic Graph Theory*. Number 207 in Graduate Texts in Mathematics. Springer Science+Business Media, LLC, New York, 2004.

[GY06] J. L. Gross and J. Yellen. *Graph Theory and Its Applications*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, 2. edition, 2006.

[Hac09] A. Hackbarth. A multi-agent simulation framework and control structure for collaborative searching and tracking. Diplomarbeit, Technische Universtität Hamburg-Harburg, Institut für Regelungstechnik, 2009.

[Jet09] http://dst.jpl.nasa.gov/control/testbeds.htm, 13.10.2009. Jet Propulsion Laboratory Homepage.

[Kri07] A. Krishnamurthy. *Coordinated Control and Maneuvering of a Network of Micro-satellites in Formation*. PhD thesis, University of Paderborn, 2007.

[LCRPS04] S. Luke, C. Cioffi-Revilla, L. Panait, and K. Sullivan. Mason: A new multi-agent simulation toolkit. *Proceedings of the 2004 SwarmFest Workshop*, 2004.

[LIS09] http://sci.esa.int/science-e/www/area/index.cfm?fareaid=27, 08.10.2009. European Space Agency Homepage.

[Mas09] http://cs.gmu.edu/ eclab/projects/mason/, 13.10.2009. Mason Homepage.

[Mat09] http://www.matsim.org/, 14.10.2009. MATSim Homepage.

[Mur07] R. M. Murray. Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129:571–583, 2007.

[Net09] http://ccl.northwestern.edu/netlogo/, 13.10.2009. NetLogo Homepage.

[OS06] R. Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on Automatic Control*, 51(3):401–420, 2006.

[OSFM07] R. Olfati-Saber, J. A. Fax, and R. M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[PAT09] http://www.path.berkeley.edu/, 08.10.2009. California Partners for Advanced Transit and Highways Homepage.

[Pop09] A. Popov. Personal notes and communication, September - December 2009.

[PPW09] U. Pilz, A. Popov, and H. Werner. Robust controller design for formation flight of quad-rotor helicopters. *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009.

[PRS07] M. Porfiri, D. G. Roberson, and D. J. Stilwell. Tracking and formation control of multiple autonomous agents: A two-level consensus approach. *Automatica*, 43(8):1318–1328, 2007.

[Reg07] A. H. Register. *A Guide to Matlab Object-Oriented Programming*. Chapman & Hall/CRC, Boca Raton, 2007.

[Sim09] http://www.simwalk.ch/, 14.10.2009. SIMWALK Homepage.

[Swa09] http://www.swarm.org/index.php/swarm_main_page, 13.10.2009. Swarm Homepage.

[Wei08] A. Weidlich. *Engineering Interrelated Electricity Markets: An Agent-Based Computational Approach*. Contributions to Management Science. Physica Verlag, Heidelberg, 2008.

[Wer09] H. Werner. *Control Systems Theory and Design*. Lecture Notes Hamburg University of Technology, 2009.

[XA05] X. Xi and E. H. Abed. Formation control with virtual leaders and reduced communications. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, pages 1854–1860, 2005.

[YADF09] C. Yu, B. Anderson, S. Dasgupta, and B. Fidan. Control of minimally persistant formations in the plane. *SIAM Journal on Control and Optimization*, 48(1):206–233, 2009.