Network-Decentralized Simultaneous Localization and Mapping by Multi-Agent Systems

T. J. Witte

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Network-Decentralized Simultaneous Localization and Mapping by Multi-Agent Systems

Master of Science Thesis

For the degree of Master of Science in Embedded Systems at Delft University of Technology

T. J. Witte

November 27, 2019

Faculty of Electrical Engineering, Mathematics & Computer Science (EEMCS) · Delft University of Technology

Abstract

Multi-Agent Systems (MAS) can be used in the exploration and mapping of unknown environments. To cooperate autonomously, each agent of the MAS must know its own location precisely within such an environment. Simultaneous Localization and Mapping (SLAM) techniques are commonly used when the Global Positioning System (GPS) is unavailable or does not provide sufficient accuracy in positioning the agents. In multi-robot SLAM additional challenges, related to data sharing, arise as the number of robots increases. These additional challenges prevent the multi-robot SLAM solutions to be scaled easily.

Network-decentralized state estimation can be used to overcome this problem. The information sharing within multi-agent systems can be modelled based on the topology of a graph. The agents form the nodes and communication is only along the edges of this network. Network-decentralized state estimation consists of designing local state observers for a network of agents to asymptotically estimate their own state based on information exchanges with neighboring agents only.

In this thesis, the concepts of network-decentralized position estimation and SLAM are combined to form a novel network-decentralized SLAM which can be used by a multi-agent system in unknown environments to build a map of the surroundings. The network-decentralized SLAM is simulated in MATLAB and evaluated based on different metrics. A volume error metric as well as different timing metrics are introduced. Based on the evaluation of these metrics, it is shown that the proposed network-decentralized SLAM can be easily scaled to larger formations to cover unknown areas faster and in a more robust and accurate way.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank my supervisor Dr. ir. Giulia Giordano for her guidance and her positive support during this thesis. Her supervision helped me a lot, especially to keep myself on track in the final months of my thesis. I am very grateful for the opportunity she gave me to do the thesis work under her supervision.

Secondly, I would like to thank the other committee members Dr.ir Manuel Mazo and Dr.ir Chris Verhoeven for their effort in reading and evaluating my thesis.

Finally, I would like to thank everyone who I have met and who supported me during my studies. You have made my time here in Delft unforgettable.

Delft, University of Technology T. J. Witte
November 27, 2019

Chapter 1

Introduction

In the last decades the worldwide interest in robotic systems has increased tremendously. Industrial robots, room cleaning robots or even a robot which assists in a heart surgery are a few of the various applications where single robotic systems are used for nowadays [1, 2, 3]. On the other hand, just as a group of humans can be more productive in completing a collective task than one individual, using a group of robots can have many benefits too. The tasks assigned to a multi-robot group are usually the ones which would be hard or even impossible for a single agent to complete, these Multi-Agent Systems (MAS) are more robust against failures of individual agents and are able to complete tasks faster [4]. Multi-agent systems are found in different fields of application, such as warehouse automation, surveillance, military systems, mobile sensor networks and vehicle platooning [5, 6, 7, 8, 9].

Multi-agent systems are usually seen as a dynamical-networked system [10, 11, 12]. Each agent forms a subsystem and behaves according to its own internal dynamics. In order to accomplish a collective task, the agents have to share information between one another. The information sharing within multi-agent systems can be modelled based on the topology of a graph where the agents form the nodes of a network and communication is only along the edges of this network [13]. By using this information the multi-agent system can be controlled either in a centralized or a decentralized approach. A centralized controller is just a single controller which uses information about every agent to determine a global optimal strategy. An increase of the number of agents will most likely lead to an inefficient, too expensive or even physically impossible system. Furthermore, a centralized controller is more sensitive to a complete failure of the system, as only one controller has to malfunction. On the other hand, a decentralized controller is based on several control units, each acting on a single agent and determining a local control law by relying on local information only. While, in general, decentralized controllers are associated with the nodes of the graph, in the network-decentralized control case the controllers are associated with the edges of the network. The actions of each controller can only affect the agents connected to the corresponding edge [14, 15, 16]. The dual problem of network-decentralized control is network-decentralized state estimation [17, 18]. The difference between the dual problems is the location where the control action takes place. In the case of network-decentralized state estimation, the agents

implement local state observers and are associated with the nodes in a graph, thus having information from the related edges to that node only.

The focus in this thesis will be on network-decentralized state estimation, thus on designing local observers for a network of agents to asymptotically estimate their own state based on information exchanges with neighboring agents only. The agents have no knowledge of other agents, except for the ones within their communication range. This should not be confused with the more common network-distributed estimation problem, where each agent aims at estimating the state of the whole system [19, 20].

Another task for a multi-agent system which has not yet been explicitly mentioned is the exploration and mapping of unknown environments [21, 22, 23]. In this case, for these multi-agent systems to cooperate autonomously, it is important to make sure that each agent gets to know precisely its own location within the environment where the agents are operating. Usually, to obtain a precise location of the agent the Global Positioning System (GPS) is used. However, in many environments, such as indoors, underwater, in remote areas or in extraterrestrial terrains, GPS will not be able to provide the agent with an accurate location [24, 25]. In these cases Simultaneous Localization and Mapping (SLAM) can be used. The aim of this concept is to localize an autonomous mobile agent in a previously unexplored environment and at the same time construct a consistent map of this environment. However, doing this for a multi-agent system in an unknown GPS-denied environment is still an active field of research [26, 27].

The goal of this thesis is to combine the concept of network-decentralized position estimation and SLAM such that the exploration and mapping of unknown environments can be dealt with in a scalable, robust and accurate way.

1-1 Thesis contribution

The main contribution of this thesis is combining the two concepts, network-decentralized position estimation and SLAM, to create a novel network-decentralized SLAM algorithm which can be used by a multi-agent system in unknown environments to build a map of the surroundings. By using a multi-agent system in a network-decentralized way the task of creating a map can be accomplished in a faster and more robust way than when just one single agent is used. On top of that, the network-decentralized characteristics ensure that limited data sharing is accomplished, making it easy to scale the system to larger formations. The multi-agent system consisting of N agents is able to fully autonomously scan an unknown environment. The agents coming from different areas in the simulation space will be able to avoid each other and form a predefined formation before they start scanning the environment. This is all achieved in a network-decentralized way.

The network-decentralized SLAM by a multi-agent system is simulated in MATLAB and evaluated based on different metrics. A volume error metric which describes the volume error between the ground-truth map and the reconstructed map is introduced as well as different timing metrics. Based on the evaluation of these metrics, it is shown that the proposed network-decentralized SLAM can be easily scaled to larger formations to cover unknown areas faster and in a more robust and accurate way. To get a more realistic feeling on how the algorithm would perform in a real-time physical system, noise is also added to the simulations.

1-2 Outline

The outline of this thesis is as follows:

- **Chapter 2** presents the definition of SLAM in more detail and why it is important for a group of robots to be able to solve this problem. Different state-of-the-art solutions to the SLAM problem are presented for multi-agent systems, as well as ways of representing a reconstructed map.
- **Chapter 3** shows the differences between the proposed network-decentralized SLAM and the SLAM algorithms described in Chapter 2. It provides also a more detailed description of the problem formulation which is used to simulate and evaluate the proposed algorithm.
- **Chapter 4** commences with some important background on graph theory, after which the network-decentralized position estimation is described.
- **Chapter 5** describes some techniques which can be used to acquire an absolute position of the agent in the absence of GPS as well as different ways of obtaining relative distance measurements between agents.
- **Chapter 6** gives an overview of the used dynamics, the underlying control and the collision avoidance algorithm for the multi-agent system.
- **Chapter 7** shows the used simulation environment and describes different parameters in more detail. Various case-studies are performed in this simulation environment to evaluate the network-decentralized SLAM algorithm.
- **Chapter 8** concludes on the presented work and gives some recommendations for future work.

Chapter 2

Existing solutions for the SLAM problem

In general, Simultaneous Localization and Mapping (SLAM) consists of the simultaneous estimation of the states of a robot and the construction of a model (the map) of the environment. It is considered a fundamental problem for robots to become truly autonomous [28]. The states of the robot can be described by, for example, its pose (position and orientation), but also by the robot's velocity, sensor biases and calibration parameters. The map, on the other hand, describes the environment where the robot operates. It is a representation of different interesting aspects about the surroundings such as the position of landmarks and other perceptions which could describe the environment the robot operates in [26].

SLAM has gained popularity since it can be used in many and various real-world applications. In numerous of these applications, modern robotic systems are used instead of humans because of the often risky and hazardous environments they must operate in. For instance, in security and surveillance or in rescue operations during natural disasters, the goal of the robotic system is to explore an environment and report a map of the surroundings to a human operator [29, 30]. In these applications, as well as in many more such as for indoor mobile robotics or self-driving cars, the robotic system requires a globally consistent map of the environment it operates in. SLAM techniques are commonly used when infrastructure based solutions, such as the Global Positioning System (GPS), are unavailable or do not provide sufficient accuracy in positioning the robotic system.

There is, unfortunately, not a single SLAM algorithm that can solve each and every localization and mapping problem. Different tasks can be identified where one SLAM formulation is more suited than another. For instance, a topological map can be used to analyze reachability of a given place, but such a map is less suited for motion planning. A locally consistent metric map can be used for obstacle avoidance and local interactions with the environment, but it may be inaccurate. On the other hand a globally consistent metric map allows the robot to perform global path planning, but the computational effort to compute and maintain this may be high.

Multi-robot SLAM algorithms have been built on top of single-robot SLAM algorithms and mostly outperform single-robot SLAM algorithms. However, in multi-robot SLAM, additional challenges arise as the number of robots increases. These challenges are related to data sharing. What kind of data is shared and how this is done among the agents are some of the key issues, which prevents the multi-robot SLAM solutions to be scaled easily [27, 31]. Furthermore, any SLAM algorithm has to deal with three main issues, namely: data processing, map representation and sensors. Once the data is captured by suitable sensors, it can be processed via different algorithms. Then eventually the environment can be modelled in a suitable map representation.

In the following subsections, underlying thoughts about data processing of the different state-of-the-art SLAM algorithms will be presented as well as some of the different ways of representing the map.

2-1 SLAM techniques

In this section some of the main solutions to the SLAM problem will be discussed. In the literature, filtering, smoothing and other solutions have been used to solve this problem. Most of these solutions are based on a probabilistic form of the SLAM problem. It requires that the probability distribution, which describes the joint posterior density of the landmark locations and vehicle states, is computed for every time step.

2-1-1 EKF-SLAM

One of these solutions is the Extended Kalman Filter (EKF)-SLAM. It makes use of the extended version of the normal Kalman Filter (KF) and was first introduced in a series of seminal papers [32, 33]. When applying a normal KF, the state of a system, $x(t)$, is usually represented by a multivariate Gaussian distribution with the assumption that the system model is linear. The multivariate Gaussian is expressed by the expected value $\hat{x}(t) = \mathbb{E}(x(t))$ of the state variable and its covariance matrix $\Sigma(t)$.

In most SLAM cases, however, the used system models are non-linear. Therefore, the EKF must be used, which linearizes the non-linear models using a Taylor Series expansion around the estimate of the current state. These non-linear models can typically be described by:

$$\begin{aligned} x(t) &= f(x(t-1), u(t)) + w(t) \\ y(t) &= h(x(t)) + v(t) \end{aligned} \tag{2-1}$$

where $f(\cdot)$ defines the system's dynamics, $u(t)$ is the input, $h(\cdot)$ is the measurement function, $w(t)$ and $v(t)$ are the process and measurement noise, respectively. The EKF iterates in a prediction-update cycle. The prediction of the state $\hat{x}(t|t-1)$ is estimated using the model of the system with input $u(t)$ and the posterior estimate from the last time step. The estimate of the covariance matrix is updated according to the Jacobian of $f(x(t))$, $\nabla f = \frac{\delta f}{\delta \hat{x}(t)}$, and the covariance of the process noise $Q(t)$. The predictor step is given by:

$$\begin{aligned}\hat{x}(t|t-1) &= f(\hat{x}(t-1|t-1), u(t)) \\ \Sigma(t|t-1) &= \nabla f(t-1)\Sigma(t-1)\nabla f^T(t-1) + Q(t-1)\end{aligned} \tag{2-2}$$

The updated estimates for the state vector are in the posterior, $\hat{x}(t)$. The Kalman gain, $K(t)$, and the residual, $\tilde{y}(t)$, are used to calculate this posterior estimate. The update step is given by:

$$\begin{aligned}\tilde{y}(t) &= y(t) - h(\hat{x}(t|t-1)) \\ S(t) &= \nabla h(t)\Sigma(t|t-1)\nabla h^T(t) + R(t) \\ K(t) &= \Sigma(t|t-1)\nabla h^T(t)S^{-1}(t) \\ \hat{x}(t) &= \hat{x}(t|t-1) + K(t)\tilde{y}(t) \\ \Sigma(t) &= (I - K(t)\nabla h(t))\Sigma(t|t-1)\end{aligned} \tag{2-3}$$

$S(t)$ is the residual of the covariance and gives an indication of the certainty that the output model $h(x(t))$ will match the actual measurements. Matrix I is the identity. The predicted state is updated using the residual $\tilde{y}(t)$, which is the difference between the measured process output $y(t)$, and the predicted output of the model $h(\hat{x}(t|t-1))$. ∇h is the Jacobian of $h(x(t))$ and $R(t)$ is the covariance matrix of the zero mean white Gaussian noise.

At each time-step the EKF does not take the previous estimate in consideration, which incorporates the linearization errors and data associations into the state. Therefore, the estimated covariance matrix tends to underestimate the state uncertainties which can lead to inconsistency. For this reason, the Unscented Kalman Filter (UKF)-SLAM was introduced [34]. The UKF propagates a small set of samples through the nonlinear model to form new covariance estimates. Therefore, it avoids the need of calculating the Jacobians.

EKF-SLAM scales with $\mathcal{O}(M^2)$, where M is the number of landmarks in the map. This leads to a limit of available landmarks that can be used, as the covariance matrix will increase significantly. This scaling problem prevents EKF-SLAM from being used in large-scale environments.

2-1-2 EIF-SLAM

The Extended Information Filter (EIF)-SLAM can be seen as the mathematical inverse of the EKF-SLAM. The EIF is more suitable for a multi-robot system than EKF, because the information matrix of the EIF has the additivity property [35, 36]. Robots can integrate their information by simply adding it together. Likewise, the fusion of duplicate landmarks can also be accomplished by using EIF.

The difference between the EIF and EKF solutions is the way the information is represented. EIF maintains an information matrix which is the inverse of the covariance matrix of the one the EKF uses. The multivariate Gaussian state vector $\hat{x}(t)$ and covariance matrix $\Sigma(t)$ are replaced by the information vector, $\hat{y}(t) = \Sigma^{-1}(t)\hat{x}(t)$ and information matrix, $\Omega(t) = \Sigma^{-1}(t)$, respectively.

The additive property avoids the $\mathcal{O}(M^2)$ scale factor, but the information matrix still grows in size. Therefore, the Sparse Extended Information Filter (SEIF) was introduced by Thrun

et al. [37]. It differs from the EIF in the fact that it keeps a sparse information matrix. It is bounded by a constant that is independent of the number of landmarks in the map. In addition, the same prediction-update cycle as in EKF can be used.

2-1-3 RBPF-SLAM

As shown before, the SLAM approaches described above encounter certain limitations. Some of these limitations can be overcome by using particle filters. However, the high dimensional state-space of the SLAM problem makes direct application of particle filters computationally infeasible. By using the so called Rao-Blackwellized Particle Filters (RBPFs), the sample-space can be reduced [38, 28]. The main idea behind Rao-Blackwellisation is that a joint state can be partitioned based on the product rule $P(x_1, x_2) = P(x_2|x_1)P(x_1)$. If $P(x_2|x_1)$ can be computed analytically, only $P(x_1)$ has to be sampled. By filtering the entire trajectory of a robot instead of excluding old poses in the filter update, RBPFs differ from the previously described solutions.

The aim of the RBPF-SLAM problem is to compute the posterior probability over the robot trajectory and map given some initial pose. This can be done by assuming that map landmarks are conditionally independent given the trajectory of the robot. It can be written as the product of two factors:

$$\underbrace{p(x_{1:t}, m|z_{1:t}, u_{0:t-1}, x_0)}_{Posterior} = \underbrace{p(m|x_{1:t}, z_{1:t}, u_{0:t-1}, x_0)}_{Map}\underbrace{p(x_{1:t}|z_{1:t}, u_{0:t-1}, x_0)}_{Trajectory} \tag{2-4}$$

where $z_{0:t}$ are the observations and $u_{0:t-1}$ the actions executed by the robot. The posterior over the map and trajectory is approximated by applying a particle filter, where each particle represents a complete trajectory and a separate map is conditioned on each sample.

The implementation of the particle filter is often done via a Sampling - Importance - Resampling (SIR)-algorithm. New particles are drawn according to a proposal-distribution that approximates the target distribution as close as possible, calculating importance weights and a resampling method. The particles with a low importance weight will be replaced by particles with a higher importance.

The single robot RBPF-SLAM can be extended to a multi-robot SLAM, however, every initial pose of the robots must be known. The aim is to simultaneously estimate the posterior probability over N robot trajectories and one map. This can also be factorised as follow:

$$p_{Posterior} = p_{Map}\, p_{Trajectoryofrobot1}\, p_{Trajectoryofrobot2} \cdots p_{TrajectoryofrobotN} \tag{2-5}$$

it is crucial that it is assumed that the trajectories are independent and that observations by one robot do not depend on the pose of the other.

Attempts to use RBPF- SLAM for more than two robots has not led to the success needed for autonomously exploring a large environment. This is due to the large number of particles required to avoid inconsistencies. The number of particles required to maintain accurate posterior distributions increases with the size of the environment, the complexity of the environment and the number of agents.

2-1-4 GraphSLAM

One of the most popular SLAM algorithms in today's applications is the GraphSLAM solution [26, 27]. This solution constantly re-linearizes around the current state estimate. The underlying thought of the GraphSLAM is that the SLAM problem is formulated as a Maximum a Posteriori (MAP) estimation problem. The unknown variable X, which consists of the trajectory of the robot as well as of the position of the landmarks in the environment, needs to be estimated. Furthermore, there is a set of measurements Z where each measurement can be expressed as a function of X. This can be described as $z_k = h_k(X_k) + \epsilon_k$ where h_k is a known function and ϵ_k random measurement noise.

The MAP estimation problem is defined as:

$$X^* = \underset{X}{\operatorname{argmax}}\, p(X|Z) = \underset{X}{\operatorname{argmax}}\, p(Z|X)p(X) \tag{2-6}$$

$p(Z|X)$ is the likelihood of the measurements Z provided X and $p(X)$ is the prior probability over X. If there is no prior knowledge about X then $p(X)$ becomes a constant and the MAP problem becomes a maximum likelihood estimation problem. Furthermore, if it is assumed that the measurements Z are independent then the MAP estimation can be factorised into:

$$X^* = \underset{X}{\operatorname{argmax}}\, p(X) \prod_{k=1}^{m} p(z_k|X_k) \tag{2-7}$$

If it is assumed that the measurement noise is zero-mean Gaussian noise then the measurement likelihood can be written as $p(z_k|X_k) \propto \exp\left(-\frac{1}{2}||h_k(X_k) - z_k||^2_{\Omega_k}\right)$ where Ω_k is the information matrix of the measurement noise ϵ_k. By also assuming that the prior can be written as $p(X) \propto \exp\left(-\frac{1}{2}||h_0(X) - z_0||^2_{\Omega_0}\right)$, then, since maximizing the posterior is the same as minimizing the negative log-posterior, the MAP estimate becomes:

$$\begin{aligned} X^* &= \underset{X}{\operatorname{argmin}} - \log\left(p(X) \prod_{k=1}^{m} p(z_k|X_k) \right) \\ &= \underset{X}{\operatorname{argmin}} \sum_{k=0}^{m} ||h_k(X_k) - z_k||^2_{\Omega_k} \end{aligned} \tag{2-8}$$

This becomes a nonlinear least squares problem, as h_k is most of the times a nonlinear function.

2-1-5 CPS

In Cooperative Positioning System (CPS), a group of robots, which do not necessarily have to be homogeneous, will usually have one parent robot on which all the sensors are applied. The other robots, children, are used as moving landmarks, such that the parent robot can perform localization. First the child robots move and stop. The parent robot identifies the child robots. Then the parent robot moves and stops, and identifies the child robots again. Via triangulation, the poses of all robots can be found. CPS-based mapping can be used in an unknown environment, when good coordination of the agents to switch between both roles is performed [39].

2-2 Map representation

Different representations of the reconstructed map can be found in literature. There are six commonly used types of maps: grid maps, feature maps, topological maps, semantic maps, appearance maps and hybrid maps [40, 41, 42, 43, 44]. A short description of each map is provided below:

- In **Grid maps** the environment is represented by a matrix of cells, where each cell represents a section of the space as a small rectangle. A binary random variable is assigned to each cell which indicates the probability that an object exists in that cell. In 2D-SLAM algorithms occupancy grid maps are one of the most used maps. This type of map can also be used for 3D modelling. In 3D these maps are called voxel maps.
- In **Feature maps** the absolute positions of extracted features from the environment are presented in a map. These features or characteristics are often described as landmarks in an environment. The position of such a landmark is accompanied by some sort of signature that uniquely characterizes it. An example of a feature map is a 3D point-cloud map.
- **Topological maps** are commonly used in every day life to, for example, navigate, determine routes and locations, or to avoid obstacles. When topological maps are used for perception or autonomy they can be seen as symbolic maps. The environment is modelled in an abstract way, where a clear distinction can be seen in the form of connected paths and intersections.
- **Semantic maps** are seen as the most abstract maps. They contain relationships between objects in the environment. Semantic maps are very similar to topological maps. They include, however, more detailed information about present objects in the environment.
- **Appearance maps** are usually developed with a vision system and include different views of the environment. Different images are captured and connected to each other to construct one consistent map.

Chapter 3

Problem formulation

The goal of this thesis is to design an approach that enables a multi-agent system to reconstruct an unknown environment, fully autonomously, in a network-decentralized way. As stated in the previous chapter, one of the challenges, for multi-robot Simultaneous Localization and Mapping (SLAM), that prevents the solutions to be easily scaled is the way data are shared between the agents. Therefore, in this thesis, a network-decentralized approach is used to overcome this problem. This enables the system to be scaled more easily compared to the previously described algorithms where mostly a centralized approach was used. The system can be scaled to incorporate more agents due to the fact that only local interactions between the agents are used, leading to limited information sharing for the whole multi-agent system.

Furthermore, the proposed novel network-decentralized SLAM algorithm does not make use of landmarks to create a map of the unknown environment, instead it uses the estimated absolute positions of the agents to build this map. By doing the localization simultaneously and using this information to build a map, the proposed algorithm will be called a network-decentralized Simultaneous Localization and Mapping algorithm. In the following section a more detailed description of the problem formulation can be found.

Another aspect which is different compared to the previously described SLAM algorithms is that a Luenberger observer will be used for the absolute position estimation of the agents. The multi-agent system will be able to simultaneously localize each agent based on network-decentralized position estimation. To obtain an estimate of the absolute position of each agent, only relative distance measurements between the agents are used. It must be noted that one of the agents will have to be connected to the *external environment*. This concept and the theory behind the network-decentralized position estimation will be explained in more detail in Chapter 4.

3-1 Detailed problem description

A group of N identical agents, all having their own internal dynamics and consisting of a certain mass m, start in different positions, typically grouped in clusters, in the environment. The agents in the different clusters will have to produce a single formation in order to be able to scan the unknown environment they are deployed in. During the formation producing phase the agents are moving freely in the 3D environment, although they do have a destination location within the predefined formation. The group of agents can only communicate with other agents if these other agents are inside a certain communication range, thus having no knowledge about agents outside this range. The agents have a collision avoidance algorithm in order to avoid each other while forming a predefined grid. After the formation is produced, the formation as a whole will be able to scan the environment. The scanning is done in a trajectory which ensures that the whole environment is scanned. The agents will hover over the area, keeping a certain height above the surface while maintaining the grid structure. Each agent will keep track of their own estimated absolute position obtained via the network-decentralized position estimation algorithm. In the end, every trajectory of each agent is merged into a single map. Where every aspect of the work presented is done in a network-decentralized manner, it must be noted that the map merging at the end is created in a centralized way. In Figure 3-1 the block scheme of the procedure can be seen.

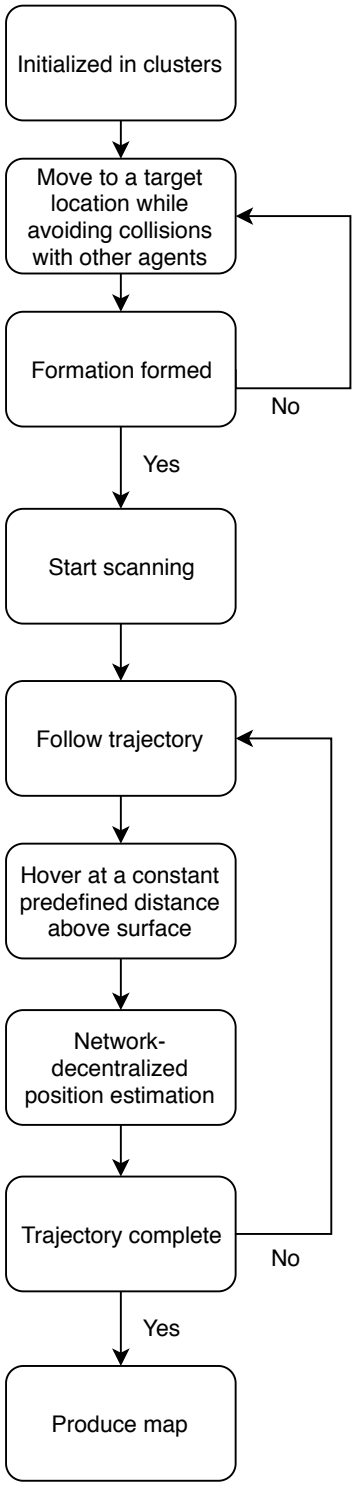


Figure 3-1: Block scheme of the procedure

Chapter 4

Network-decentralized position estimation

The main idea behind network-decentralized position estimation presented in this thesis is based on the concept described by Giordano et al. in [17]. The algorithm aims at designing local observers for a network of agents to asymptotically estimate their own state based on information exchanges with neighboring agents only. This is done through communication channels. Since transmitting and receiving information requires energy, which is usually scarce, only agents within a certain communication range can exchange information. In this way information propagates through the system via intermediate agents. These local interactions between agents form a network and can be described by a graph. In the network-decentralized estimation framework, it is assumed that the global network topology is unknown to the agents and may change over time. Algebraic graph theory can be used to effectively model the communication between agents [13, 14].

To get a better understanding of the framework for the network-decentralized position estimation, some background information about graph theory will be provided in the following section.

4-1 Basic graph theory concepts

In a graph the agents are represented by nodes (vertices) and the interactions between agents by edges (arcs). A set of nodes is denoted by $\mathcal{N}$, a set of edges by $\mathcal{E}$. The topology of a graph is therefore defined by the ordered pair $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and mathematically by the Adjacency, Degree, Incidence and Laplacian matrices.

- **Degree matrix D:** The degree of a node in a graph is denoted by the number of neighbors of that node. The degree of every node in a graph can be written in the diagonal degree matrix $D \in \mathbb{N}^{n \times n}$, whose diagonal entries d_{ii} are the degree of the ith node respectively.

- **Adjacency matrix A:** The Adjacency matrix of an undirected graph is a symmetric $n \times n$ matrix, which describes the adjacency relationships of a graph. If there is an edge between node i and j, then entry a_{ij} is 1, and 0 otherwise.
- **Incidence matrix B:** The Incidence matrix, $B \in \{-1, 0, 1\}^{n \times m}$, describes the orientation of the edges between the nodes. Its entries b_{ij} are 1 if the jth edge enters node i, -1 if it leaves node i and 0 otherwise. The Incidence matrix has zero-sum columns.
- **Laplacian matrix L:** The Laplacian matrix is defined as the difference between the Degree and the Adjacency matrices: $L = D - A$, or in the case of an undirected graph via the Incidence matrix and its transpose: $L = BB^T$. The Laplacian matrix and its eigenvalues convey useful information about a graph, such as a representation of the graph connectivity.

Throughout this thesis a bidirectional communication between agents will be assumed. This means that every link between agents is both incoming as well as outgoing. Such a graph is called a symmetric directed graph, which is in fact the same thing as an undirected graph [45].

Consider the following Figure 4-1, consisting of five agents, where an agent is only able to exchange information with another agent when the other agent is within a predefined communication range r_{comm}, i.e. within a certain sphere of that agent.

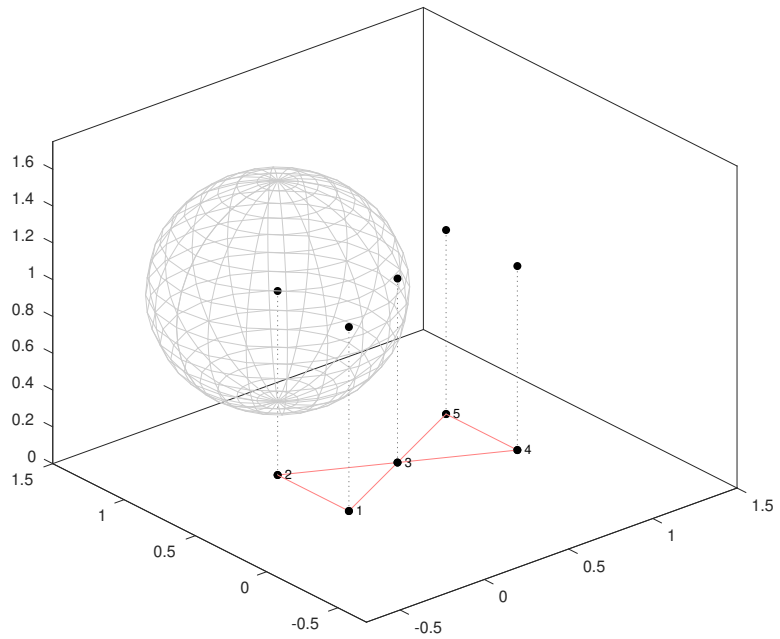


Figure 4-1: System consisting of five nodes, where agents are able to communicate with each other if they are within a certain communication range (r_{comm} of 0.7 m)

The underlying graph topology is shown in the xy-plane, where each line between two nodes is a symmetric directed edge pair. The bidirected graph topology is shown in Figure 4-2.

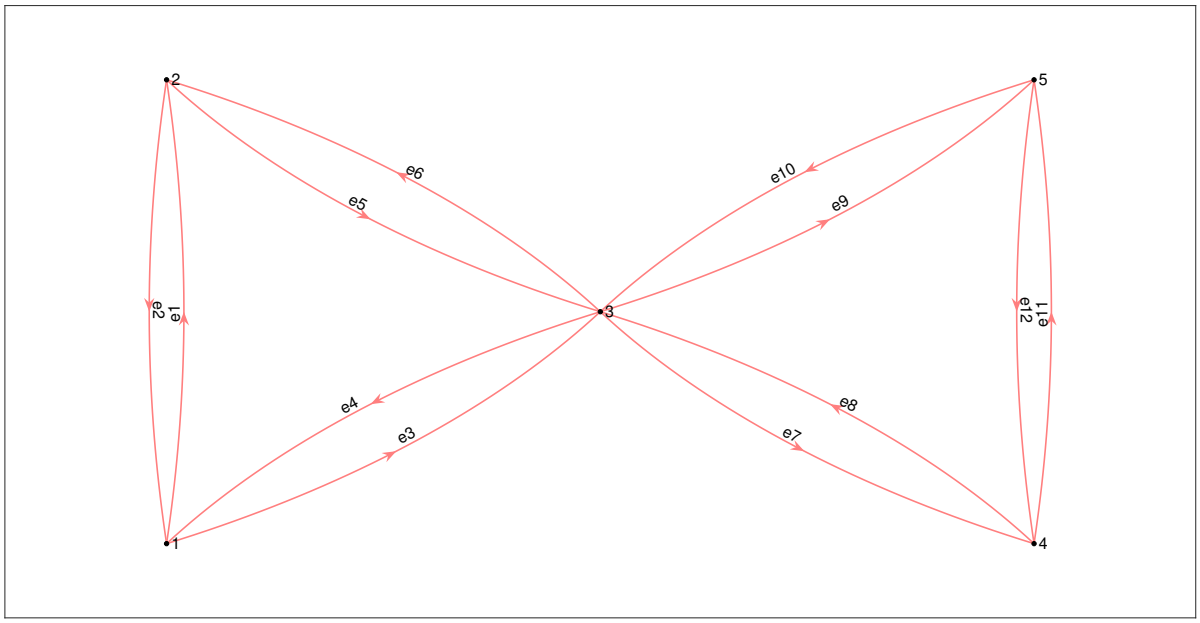


Figure 4-2: Topology of the graph with 5 nodes and bidirectional communication

The Incidence matrix characterising this graph is given by:

$$B = \begin{array}{c} \begin{matrix} e1 & e2 & e3 & e4 & e5 & e6 & e7 & e8 & e9 & e10 & e11 & e12 \end{matrix} \\ \begin{bmatrix} -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix} \end{array} \begin{matrix} n1 \\ n2 \\ n3 \\ n4 \\ n5 \end{matrix} \tag{4-1}$$

4-2 External connected graph

Following Giordano et al. [17], the generalized Incidence and generalized Laplacian matrices are introduced. These matrices play an important role in the network-decentralized state estimation algorithm, which will be described in Section 4-3.

In order for a group of agents to reconstruct their positions, at least one of the communicating agents must have information about its actual absolute position, so one of the agents must be connected to the *"external environment"*. A connection to the external environment could for example mean that one of the agents has a connection to an anchor point, or that one of the agents has the ability to know its absolute position directly through sensors. Different methods to obtain an absolute position in the absence of Global Positioning System (GPS) will be described in Chapter 5.

In order for information to propagate through a whole network, the graph must be connected. A graph is defined as connected when it is both internally as well as externally connected. Internally connected means that it is possible from each node in the graph to reach each other node via a sequence of distinct adjacent nodes (path). A graph is externally connected if, for each node, a path exists connecting it to a node adjacent to the external environment. A connection to the external environment can be accomplished by adding a fictitious node 0, an external edge connects node 0 to a node in the graph. Continuing on the example of Figure 4-1, now with two of the agents being connected to the external environment. The following topology is obtained, see Figure 4-3.

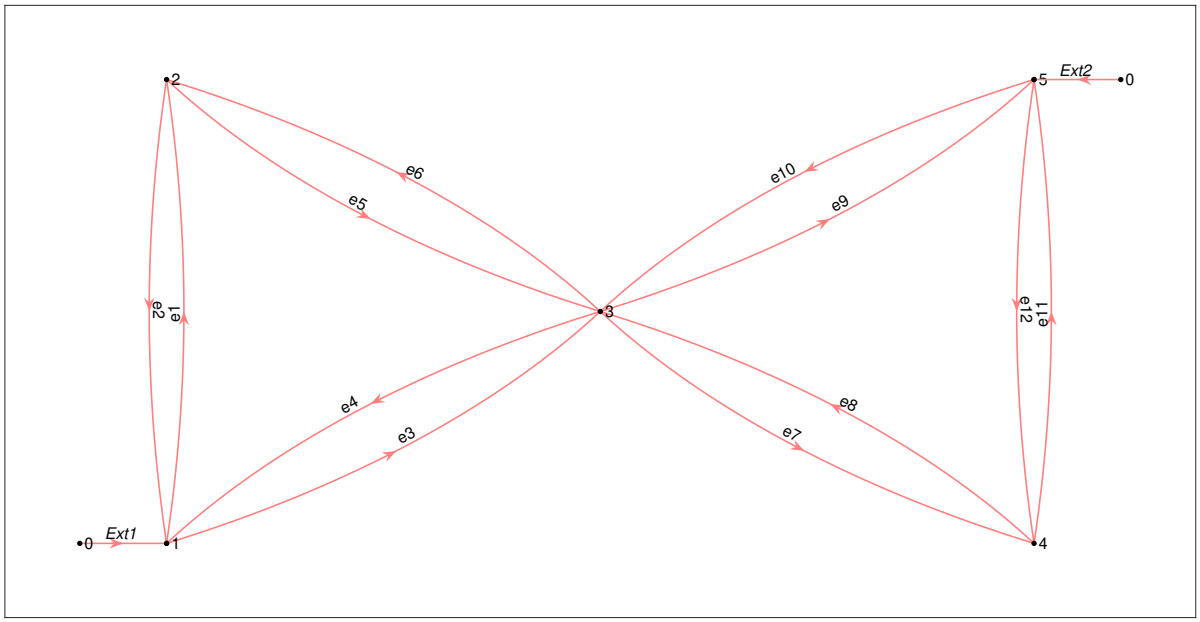


Figure 4-3: Topology of the graph with external connected nodes

The corresponding standard Incidence matrix G_0 is given by:

$$G_0 = \begin{array}{c} \begin{matrix} Ext1 & e1 & e2 & e3 & e4 & e5 & e6 & e7 & e8 & e9 & e10 & e11 & e12 & Ext2 \end{matrix} \\ \left[\begin{matrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 \end{matrix}\right] \end{array} \begin{matrix} \\ n0 \\ n1 \\ n2 \\ n3 \\ n4 \\ n5 \end{matrix}$$

The generalized Incidence matrix is obtained by removing the row associated with node 0 of the standard Incidence matrix. The generalized Incidence matrix G for the graph in Figure 4-3 is then given by:

$$G = \begin{array}{c} \begin{array}{ccccccccccccc} Ext1 & e1 & e2 & e3 & e4 & e5 & e6 & e7 & e8 & e9 & e10 & e11 & e12 & Ext2 \end{array} \\ \left[\begin{array}{ccccccccccccc} 1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 \end{array}\right] \end{array} \begin{array}{c} \\ n1 \\ n2 \\ n3 \\ n4 \\ n5 \end{array}$$

The generalized Laplacian matrix $\mathcal{L}$ is defined as

$$\mathcal{L} = GG^T \tag{4-2}$$

The generalized Laplacian is for externally connected graphs a positive definite matrix. The smallest eigenvalue of the generalized Laplacian, denoted by λ_{min}, is a measure of robustness for the networked-decentralized estimation algorithm if the observer gain is greater than a certain lower bound that depends on this eigenvalue [17]. As shown in [17], this ensures robust stability even with unknown or switching network topologies.

4-3 General framework network-decentralized state estimation

A multi-agent system consisting of N agents, where each agent has dynamics as formulated in Chapter 6, can be described in general by the system of equations:

$$\dot{x}_i = A_i x_i + B_i u_i, \quad i = 1, ..., N \tag{4-3}$$

A single state-space equation for the multi-agent system can be formed by stacking all dynamics of each individual agent in the following way:

$$\begin{bmatrix} \dot{x}_1(t) \\ \vdots \\ \dot{x}_N(t) \end{bmatrix} = \mathcal{A} \begin{bmatrix} x_1(t) \\ \vdots \\ x_N(t) \end{bmatrix} + \begin{bmatrix} B_1 \\ \vdots \\ B_N \end{bmatrix} u(t) \tag{4-4}$$

where $\mathcal{A}$ is a diagonal matrix, so that the dynamics of each single agent are decoupled from one another:

$$\mathcal{A} = \begin{bmatrix} A_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & A_N \end{bmatrix} \tag{4-5}$$

Relative measurements y between the agents are denoted by:

$$y = \mathcal{C}x \tag{4-6}$$

These measurements are associated with the arcs of the network. The inter-agent communication is described in the output matrix $\mathcal{C}$ and is based on the topology of the network, i.e. the generalized incidence matrix G. So that:

$$\mathcal{C}(t) = G^T(t) \otimes C_{ij} \tag{4-7}$$

where $\otimes$ denotes the Kronecker product, which multiplies C_{ij} with each entry in $G(t)$. C_{ij} are generic matrices and their sizes are dependent on the states that should be used for the relative measurements. As the topology of the system could change over time, this output matrix is also time dependent.

Each agent runs a local estimator in order to estimate its absolute position. This estimator is defined as:

$$\dot{z}_i = A_i z_i + B_i u_i + \sum_{j \in \mathcal{O}_i} L_{ij}(\hat{y}_j - y_j) \tag{4-8}$$

where z is the estimated absolute position of the agent, $\mathcal{O}_i$ is the set of neighbors for each agent and L_{ij} is the ijth entry of the observer gain matrix. In order to have a network-decentralized observer, L must have the same block structure as the generalized incidence matrix. The estimated measurement $\hat{y}_j$ of each arc is given by:

$$\hat{y}_j = \sum_{k \in \mathcal{N}_j} C_{jk} z_k \tag{4-9}$$

where $\mathcal{N}_j$ is the set that indexes the nodes connected to arc j.

The dynamics of the whole multi-agent system, including observer, can be described by:

$$\begin{aligned} \dot{x} &= \mathcal{A}x + Bu \\ y &= \mathcal{C}x \\ \dot{z} &= \mathcal{A}z + Bu - Ly + L\mathcal{C}z \end{aligned} \tag{4-10}$$

where L is the Luenberger observer gain.

The estimation error $e = x - z$ has the following dynamics:

$$\dot{e} = \mathcal{A}e + L(y - \mathcal{C}z) = (\mathcal{A} + L\mathcal{C})e \tag{4-11}$$

Example

Consider the following undirected graph from Figure 4-4, which is a simplified version of the previously used network in Figure 4-1. A connection with the external environment is indicated with an X. Here y_1 is associated with a relative measurement depending on the states of agent 1 and the external environment, y_2 is the relative measurement between the states of agents 1 and 2, y_3, y_4 are the relative measurement between the states of agents 1 and 3 and agents 2 and 3 respectively.

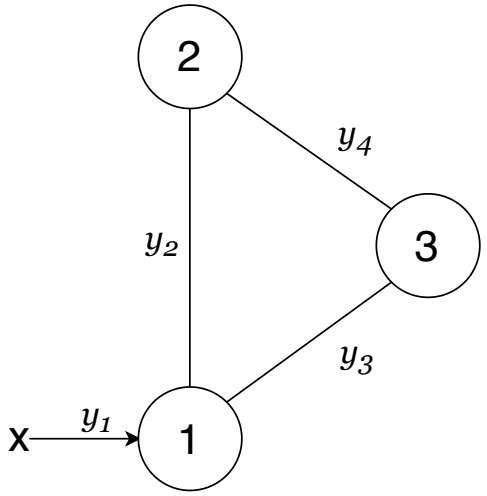


Figure 4-4: Topology of the undirected graph with 3 nodes, where the external connection is indicated by X.

As agent 1 is connected to the external environment the relative measurement for y_1 purely depends on the state of agent 1, thus agent 1 is capable of measuring its absolute position directly. For agents 2 and 3 their absolute position cannot be determined directly. Thus these states have to be reconstructed by interacting with neighboring agents.

Using Equation 4-6 and Equation 4-7 it can be obtained:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} C_{11} & 0 & 0 \\ C_{21} & C_{22} & 0 \\ C_{31} & 0 & C_{33} \\ 0 & C_{42} & C_{43} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{4-12}$$

where x_i are the states of the ith agent. As identical agents are being used C_{ij} is the same for each entry. However, the C_{ij} matrices in each row have opposite signs if there are more than one in a row.

The overall estimator matrix is given by:

$$L = \begin{bmatrix} L_{11} & L_{12} & L_{13} & 0 \\ 0 & L_{22} & 0 & L_{24} \\ 0 & 0 & L_{33} & L_{34} \end{bmatrix} \tag{4-13}$$

which has the same block structure as $\mathcal{C}^T$.

Using Equations 4-8 and 4-9, the local observer for agent 1 can be described as:

$$\begin{aligned} \dot{z}_1 = {} & A_1 z_1 + B_1 u_1 + L_{11}(C_{11} z_1 - y_1) \\ & L_{12}(C_{21} z_1 + C_{22} z_2 - y_2) \\ & + L_{13}(C_{31} z_1 + C_{33} z_2 - y_3) \end{aligned} \tag{4-14}$$

Agent 1 measures the relative distances y_2 and y_3, receives the estimated outputs $C_{22}z_2$ from agent 2 and $C_{33}z_3$ from agent 3, computes its estimated outputs $C_{21}z_1$ and $C_{31}z_1$ which it transmits to agent 2 and 3 respectively. Note that since agent 1 is externally connected, it receives its absolute position y_1 and compares it with the estimated position $C_{11}z_1$.

4-3-1 Computation of the observer gain

If in the system described in Equation 4-10 we have $A_i = A_1$, $C_{ij} = C_1$ and (A_1, C_1) is detectable, then the system is network-decentralized detectable if either A_1 is asymptotically stable or the system is externally connected. By assuming that there will be an external connection to the environment, the considered system will be network-decentralized detectable and thus a suitable observer gain can be obtained which asymptotically drives the error dynamics to zero. For a single subsystem (A_1, C_1), if the previous assumptions holds, a local observer gain can be designed as follows:

$$L_1 = -P_1^{-1} C_1^T \tag{4-15}$$

where P_1 satisfies

$$A_1^T P_1 + P_1 A_1 - 2C_1^T C_1 < 0 \tag{4-16}$$

Once the matrix P_1 is obtained, global stability can be achieved by applying the network-decentralized filter gain:

$$\begin{aligned} L &= -\gamma P^{-1}\mathcal{C}^T \\ P &= \text{blockdiag}\{P_1, P_1, ..., P_1\} \end{aligned} \tag{4-17}$$

for some $\gamma > 0$ and $\mathcal{C}$ as in Equation 4-7. Provided that γ is larger than $\frac{1}{\lambda_{min}}$, where λ_{min} is the smallest eigenvalue of the generalized Laplacian, the error system with L as in Equation 4-17 is asymptotically stable [17].

Chapter 5

Positioning techniques

For a real-life deployment, for example, a group of quadcopters could be used as the multi-agent system. For the network-decentralized Simultaneous Localization and Mapping (SLAM) to work properly at least one of the agents must be able to determine its absolute position on a common reference frame. Different techniques can be used to obtain this absolute position. In outdoor environments this is usually done through the well-known Global Positioning System (GPS). This thesis mainly focuses on environments where GPS can not be used for this purpose. Therefore, other techniques have to be exploited. Besides the absolute position that has to be determined for one of the agents, also the relative distances between agents should be obtained. In this chapter some techniques to obtain the absolute position as well as the relative distances between agents will be described.

The most common techniques are based on Vision, Infrared (IR), Ultrasound, and Radio Frequency (RF) positioning systems [46, 47, 48, 49, 50, 51]. These techniques use measurement methods based on radio signal metrics such as Received Signal Strength Indication (RSSI), Angle of Arrival (AoA), Time of Arrival (ToA), and Time Difference of Arrival (TDoA). For a more in depth explanation of these concepts, the reader is referred to [52].

As stated in Chapter 3, the agents will keep a certain height above the surface. To maintain this height, for example, a laser altimeter can be used. Besides the ability to keep a certain height above the surface, these measurements can also, indirectly, be used to measure the relative height between agents.

5-1 Summary of absolute positioning based on different techniques

5-1-1 Vision

Nowadays, vision based positioning techniques are used more due to the improvement and miniaturization of actuators (e.g. lasers) and more advanced detectors. Besides these improvements, the increase in both data transmission rates and computational capabilities, as

well as the development of high performance image processing algorithms, make this technology more efficient. There are two ways how vision based positioning works:

- **Fixed camera**: The environment is equipped with cameras at fixed locations. In this way a moving object can be located through images captured by the different cameras. In order to do this the object should be distinguished via salient features. If these features are captured by the camera, its location can be calculated with respect to the camera's fixed position. The position of the target is estimated based on its position within the captured image and the spatial distribution of its salient features.
- **Mobile camera**: The goal is to estimate the position and orientation of the mobile camera. The object is equipped with a camera and the localization is performed by placing landmarks in known orientations and positions. If the camera detects two or more landmarks, it can find its own position and orientation. Another method is by extracting environment features. The camera captures an image and features are extracted and compared with features from stored images in order to estimate the location of the camera. This method requires a database of pre-captured images of the surroundings.

5-1-2 Infrared

In an infrared positioning system the agent is equipped with an infrared transmitter which emits an IR signal at regular intervals. This signal carries a unique identifier code (ID). Infrared receivers, at least three, are placed at fixed positions in the environment. They detect the ID and by triangulation retrieve the position. IR technology requires a Line of Sight (LoS) between transmitter and receivers in order to function properly.

5-1-3 Ultrasound

Ultrasound positioning systems use ultrasonic waves to obtain distances between fixed-point stations and the agent. The transmitter sends a radio signal and an ultrasonic wave at the same time. The radio signal reaches receivers almost instantaneously, while the ultrasonic wave travels at a lower speed. The difference between the arrival times at the different receivers is then used to derive the relative bearing.

5-1-4 Radio frequency

Different radio frequency based positioning systems can be distinguished.

Radio Frequency Identification

Radio Frequency Identification (RFID) positioning systems use radio frequency waves at Ultra High Frequencies (UHF) to exchange information. The agent is equipped with a tag and the environment consists of UHF-RFID readers at fixed locations. With these readers it is then possible to locate the tag via geographical calculations, such as triangulation, trilateration, and multilateration.

Bluetooth

Bluetooth is a technology that allows electronic devices to communicate wireless. Location fingerprinting, which is a technique to identify a location based on a record of radio signals, can be used. Each agent measures the RSSI of a set of radio signals coming from all visible beacons. With a predefined radio map consisting of several RSSI values a position can be obtained via trilateration.

5-1-5 UWB

Most radio frequency positioning systems are afflicted by multi-path distortion of radio signals reflected by walls and obstacles in indoor environments. Due to the Ultra-wideband (UWB) pulses the reflected signals can be filtered from the original signal more easily, resulting in a higher accuracy. The sensors, located at known positions, receive UWB signals from an agent with an active tag.

5-1-6 Data fusion

Data fusion strategies can be used to integrate the positioning data measured by different positioning techniques into a more accurate representation. For example, a Kalman filter can be used to improve the overall localization precision. By combining a 3D laser scanner, UWB and Inertial Navigation System (INS), the positioning accuracy is increased significantly compared to INS-only and UWB-only approaches [53].

5-2 Relative measurements between UAVs

In Table 5-1 an overview of papers is presented where different techniques, among the previously described ones, are used to measure relative distances between UAVs. It is further shown with what level of accuracy this has been achieved. This accuracy will be used in Chapter 7 for the simulations where noise is added to the relative distance measurements between the agents to obtain a more realistic view on how the system will work in a real-time physical setting.

Table 5-1: Summary of different relative measurement techniques

Paper	Technique	Mean error
System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization [54]	Vision	0.15 m
3-D reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning [55]	Vision	0.4 m
3-D relative positioning sensor for indoor flying robots [56]	Infrared	0.18 m
On-board Bluetooth-based Relative Localization for Collision Avoidance in Micro Air Vehicle teams [57]	Bluetooth	1.14 m
Ultra-Wideband and Odometry-Based Cooperative Relative Localization With Application to Multi-UAV Formation Control [58]	UWB	0.15 m
On-board range-based relative localization for micro air vehicles in indoor leader-follower flight [59]	UWB	0.25 m

Chapter 6

Multi-agent system

In this thesis a multi-agent system consisting of N identical agents is considered, where each agent has its own dynamics. For the overall system to work autonomously, the agents should be capable of hovering through the whole environment in a formation. Forming this formation without causing collisions is a must, as this may otherwise lead to casualties which could disturb the system and lead to a failure or even a shutdown of the complete system. Because of this, the agents should be equipped with a collision avoidance algorithm. In this chapter, the agent dynamics of each individual agent is presented as well as the backstepping control method which is used to control the multi-agent system. Finally, the collision-avoidance and the formation producing protocol are presented.

6-1 Agent dynamics

The considered agents have 3D holonomic dynamics. The agents can be seen as spheres with radius R and an evenly distributed mass m. As the agents have holonomic dynamics they can directly move in any direction. The holonomic agent dynamics can be represented by the following linear state-space system:

$$\underbrace{\begin{bmatrix} \dot{r}_{x,i} \\ \dot{r}_{y,i} \\ \dot{r}_{z,i} \\ \ddot{r}_{x,i} \\ \ddot{r}_{y,i} \\ \ddot{r}_{z,i} \end{bmatrix}}_{\dot{x}_i} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{A_i} \underbrace{\begin{bmatrix} r_{x,i} \\ r_{y,i} \\ r_{z,i} \\ \dot{r}_{x,i} \\ \dot{r}_{y,i} \\ \dot{r}_{z,i} \end{bmatrix}}_{x_i} + \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_i} & 0 & 0 \\ 0 & \frac{1}{m_i} & 0 \\ 0 & 0 & \frac{1}{m_i} \end{bmatrix}}_{B_i} u_i(t) \tag{6-1}$$

In Equation 6-1, $r_{x,i}$, $r_{y,i}$ and $r_{z,i}$ are the local coordinates of agent i, while m_i is the mass of an agent and the input u_i is an acceleration acting on the agent. Furthermore, the speed of each agent is limited and can be represented by:

$$S_i(t) \triangleq \sqrt{\dot{r}_{x,i}^2 + \dot{r}_{y,i}^2 + \dot{r}_{z,i}^2} \tag{6-2}$$

where speed is denoted as the length of the agent's 3D velocity vector.

6-1-1 Backstepping control

Now that the dynamics of each individual agent are defined, the controller, which also embeds the collision avoidance protocol, for each agent will be explained in more detail. As the speed of the agents is limited the controller must also be able to deal with this. Therefore, a saturation function must be included. Recall that an acceleration component is used as input and the fact that the agents have a speed limit will introduce non-linearities to the system. Backstepping control is, therefore, used to stabilize the system by using non-linear feedback. It can be seen as a double nested proportional controller, where the inner-loop generates a reference velocity, $\bar{\dot{x}}_i(t)$, based on the current position and goal position of the agent. The reference velocity is tracked by the outer-loop which produces the input, $u_i(t)$, for the system.

$$\begin{aligned} \bar{\dot{x}}_i(t) &= \underset{S_{lim,i}}{\text{sat}} \left[(\bar{r}_i - x_i(t))k_1\right] \\ u_i(t) &= (\bar{\dot{x}}_i(t) - \dot{x}_i(t))k_2 m_i \end{aligned} \tag{6-3}$$

$\bar{r}_i$ is the goal position, x_i(t) and $\dot{x}_i(t)$ are the current position $(r_{x,i}, r_{y,i}, r_{z,i})$ and velocity $(\dot{r}_{x,i}, \dot{r}_{y,i}, \dot{r}_{z,i})$ of agent i at time t respectively. k_1 is a gain for the difference between the current location and destination and k_2 is the gain for the difference between the reference that is calculated and the actual velocity. By substituting u_i of Equation 6-3 in Equation 6-1 the masses cancel out. As a result, the closed-loop stability, based on the eigenvalues of the system, is independent of the mass [60].

6-1-2 Collision avoidance protocol

The collision avoidance protocol is part of the system as for obvious reasons the agents should be able to avoid each other during the formation producing phase. The used collision avoidance protocol is based on the one described in a previous thesis [61], where a network-decentralized collision avoidance in 3D was created.

The used collision avoidance protocol for a multi-agent system in 3D is based on the 2D "avoid by turning right" principle [60]. Each agent is equipped with two distinct collision detection regions. The outer detection region, which is facing the direction the agent is moving in, is limited by a certain field of view of the agent. This outer detection zone is time-varying, it scales with the agent's velocity and it is defined as:

$$R_{cz}(t) = c + \beta v_i(t) \tag{6-4}$$

where c and β are tuning parameters. The second region, the emergency detection region which has a 360 degrees in sweep width, is not time-varying. If an obstacle is detected by one of the regions a vector is drawn from the center of the agent to the intersection point with the obstacle. From this intersection point a preferred avoidance region circle is drawn. The obstacle vector is rotated until this vector is projected on the preferred avoidance region circle. This point becomes the new velocity setpoint of the agent. The visualization of this concept is shown in Figure 6-1.

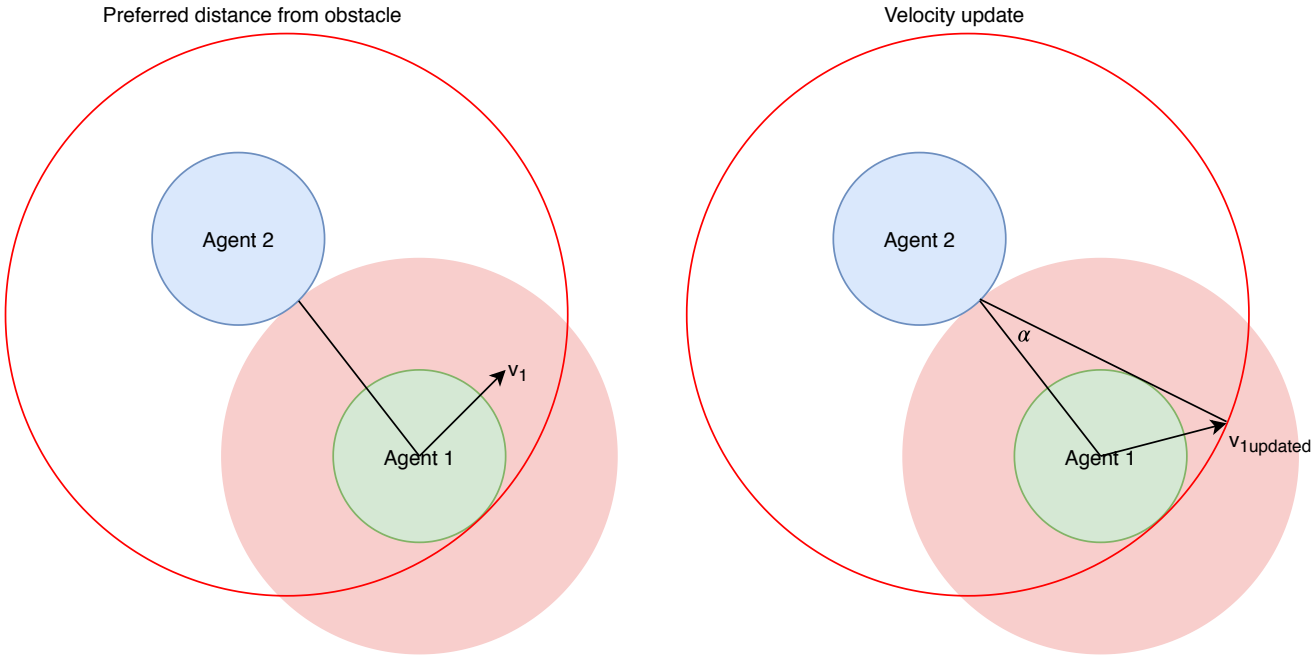


Figure 6-1: 2-D collision avoidance protocol visualized

In the extended concept for 3D, the detection regions are expanded to spheres. The obstacle's position is projected onto the XY plane to see if the obstacle falls within the outer detection region of the agent. The turn right principle is not defined in 3D, therefore, a 2D plane must be defined for each and every collision situation. To perform the avoidance manoeuvre the new avoidance velocity vector must be rotated so it is defined in the 2D plane relative to the agent-obstacle interaction. The vector is rotated relative to this new coordinate system and then transformed back. This can be done mathematically via the following transformation:

$$\begin{aligned} y &= AR(\theta)A^{-1}v_1 \\ A &= \begin{bmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{bmatrix} \end{aligned} \tag{6-5}$$

v_1 is the vector between the intersection point and the agent, $v_2 = (y_1, -x_1, 0)$ is the 2D plane, and $v_3 = v_1 \times v_2$ is the orthogonal vector between them. $R(\theta)$ is the 3D rotation matrix about the z-axis.

6-2 Formation producing and tracking

The agents will scan the environment by following a trajectory in a predefined formation. This consists of two different control problems, a formation producing and a formation tracking problem. In formation tracking the multi-agent system keeps its formation fixed and as a whole tracks a trajectory. In the formation producing case each individual agent follows its own trajectory to produce a formation. With the network-decentralized position estimation properly implemented, each agent will have an accurate estimated absolute position. By taking the difference between the estimated absolute position, $r(t)$, and the reference position, $\bar{r}$, a reference error can be defined as $e_r = \bar{r} - r(t)$ for each agent. This reference can be used for the formation producing as follows. Each agent shares its reference error with neighbouring agents. In this way a new reference can be tracked by each agent:

$$\begin{aligned} \bar{r}_{new,i} &= e_{r,i} - e_{r,j} \\ \bar{r}_{new,j} &= e_{r,j} - e_{r,i} \end{aligned} \tag{6-6}$$

if $e_{r,i}$ and $e_{r,j}$ are the same, the new reference will be zero and these agents will have produced a formation. For the multi-agent system this can be written as:

$$\underbrace{\begin{bmatrix} \bar{r}_{new,1} \\ \bar{r}_{new,2} \\ \vdots \\ \bar{r}_{new,N} \end{bmatrix}}_{r_{new}} = \frac{1}{2} G^T G \underbrace{\begin{bmatrix} e_{r,1} \\ e_{r,2} \\ \vdots \\ e_{r,N} \end{bmatrix}}_{\bar{r} - r(t)} \tag{6-7}$$

where G has the same form as the output matrix described in Equation 4-7. It must be noted that Equation 6-7 only produces the desired formation and does not make the agents go to their final destination. To drive the agents to their final destination a weighted offset ρ is added to Equation 6-7:

$$r_{new} = \left(\frac{1}{2}G^T G + \rho I_{3N}\right)(\bar{r} - r(t)) \tag{6-8}$$

where I_{3N} is the identity matrix of size $3N$ x $3N$.

A low ρ can be used to make the agents form the formation faster, whereas a high ρ is used to let each agent fly to the destination faster. Thus, for the formation producing phase a low ρ must be used, whereas for the formation tracking a higher ρ is preferred. The new reference r_{new} will be used in the backstepping controller to obtain a new reference velocity.

Chapter 7

Simulations

In Chapter 3 the problem formulation was defined. In this chapter different simulations will be presented to determine the scalability, robustness and accuracy of the network-decentralized Simultaneous Localization and Mapping (SLAM) algorithm by a multi-agent system. First, a short introduction to the simulated environment is provided, in which a number of different parameters will be explained. The simulations were done in MATLAB R2019a, the hardware specifications can be found in Appendix A-1.

7-1 Simulation environment

An example used to simulate the unknown environment can be seen in Figure 7-1. The multi-agent system should be able to reconstruct such an environment based on the network-decentralized SLAM algorithm.

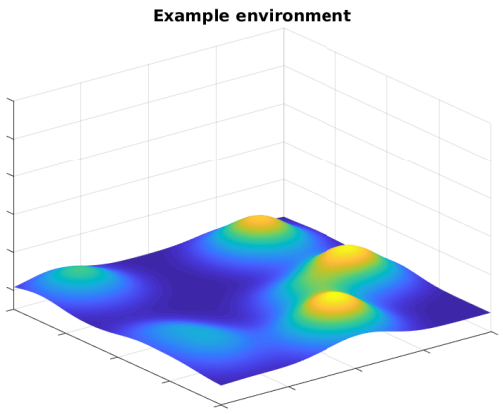


Figure 7-1: Example of the unknown environment

The environment is randomly generated with n hills of various sizes. The map is built using a grid in the x and y directions. The grid consists of a 400x400 matrix, where each entry of the matrix is 0.1 m in reality, so the area to be scanned is 40x40 m. To each (x, y) coordinate a certain height is appointed, with a maximum height of 3 m.

The used multi-agent system consists of N agents. Additionally, in the simulations the agents are represented by spheres. The radius of each sphere is set to 0.15 m, which is a common size for micro-UAVs. While hovering the agents will keep a distance of 2 m above the surface. The multi-agent system will hover over the area in a predefined formation. This formation has the grid structure which is shown in Figure 7-2, where 9 agents are used for showcase purposes. The inter-agent distance d_{inter} as well as the communication radius r_{comm} are also visualized. The agent at the left bottom of the formation is connected to the external environment and thus, in principle, knows its absolute position. The total width/height of the formation is defined by d_{width}.

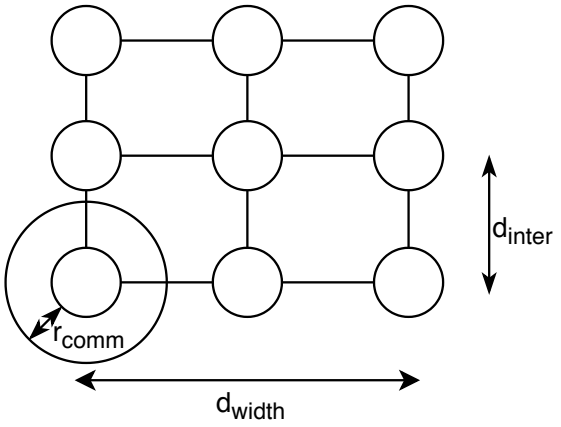


Figure 7-2: Formation structure, shown for 9 agents

In Figure 7-3 the trajectory which the formation will follow is shown. The trajectory is seen from the middle of the formation. It is chosen so that every corner and the middle section of the environment is covered at least once by one of the agents.

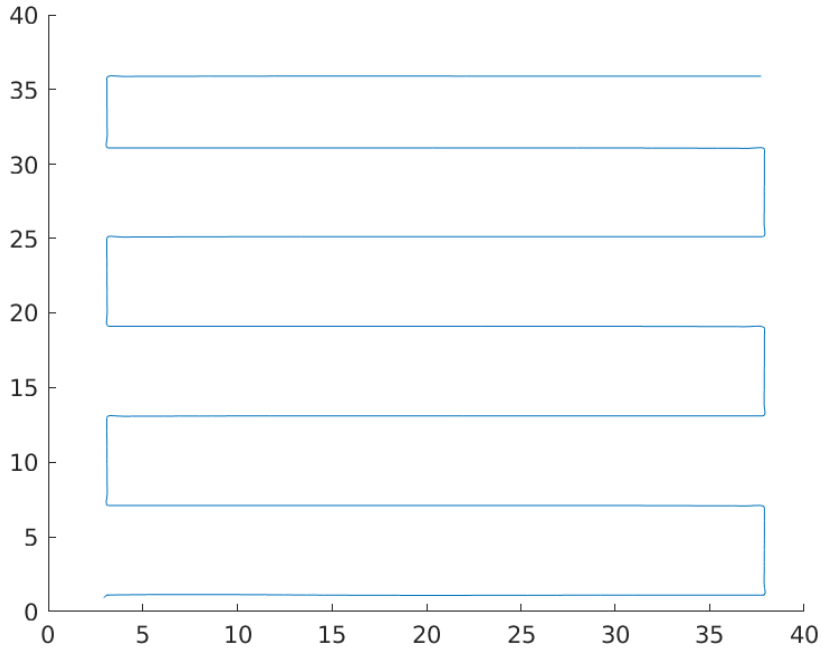


Figure 7-3: Indication for the trajectory of the formation

It must be noted that the simulations are done in a synchronous way, meaning that vehicle dynamics, control calculations as well as the state estimation are all updated in each time step. The time-step size is set to 0.01 seconds. This time-step has been chosen to make the simulations run fast enough and more efficient than when a smaller time-step size would be used.

In the upcoming simulations, the γ coefficient of the observer gain described in Equation 4-17 has been chosen so that the error system is stable and there are no numerical issues arising given the chosen time-step size. Numerical issues would occur if a higher γ coefficient were chosen due to instability of the Euler discretization with the given time-step size.

A summary of the used parameters is given in Table 7-1. In order to limit the communication between agents, in the first place, to the direct neighboring agents only, the communication radius r_{comm} is set to be the same size as the inter-agent distance d_{inter}. The design parameters will be discussed more in detail in the different case-studies.

Table 7-1: Set of parameters

Parameter	Description	Value
N	Number of agents	Variable
R_{agent}	Size of agents	0.15 m
d_{inter}	Inter-agent distance in the grid	Variable
r_{comm}	Communication radius	Variable
d_{width}	The width/height of the formation	Variable
t_d	Time step size	0.01 s
A	Area width	1600 m^2
v_{max}	Maximum speed limit	2 m/s
γ	Observer gain	5

7-1-1 Evaluation metrics

The following evaluation metrics are introduced to analyze the network-decentralized SLAM algorithm:

- **Total Time Taken (TTT):**
$$TTT = t_f - t_s \tag{7-1}$$
The difference between the start time and the final time of the simulation.

- **Time to Convergence (TC):**
$$TC = t_c - t_s \tag{7-2}$$
The time it takes for the network-decentralized estimation to converge to within a 0.05m neighborhood of the true value.

- **Total Time to Scan Area (TTSA):**
$$TTSA = TTT - TC = t_f - t_c \tag{7-3}$$
The time it takes to scan the area. This is defined as the difference between the total time taken and the convergence time of the observer.

- **Volume error (V_{err}):**
$$V_{err} = \frac{V_{real} - V_{recon}}{V_{real}} 100\% \tag{7-4}$$
A volume error in percentage is given by the difference between the total volume of the ground-truth map and the total volume of the reconstructed map, divided by the total volume of the ground-truth map.

7-2 Simulation of network-decentralized SLAM

In the following sections the performance of the algorithm will be determined based on speed, accuracy and robustness. The goal of the case-studies is not to provide an optimal setting for the problem, but more to see how different parameters can be tuned to get to an optimal setting.

As a square formation is used, only 4, 9, 16, 25, 36, 64, 81 and 100 agents are considered for the simulations. In Figure 7-4 a snapshot of a group of 16 agents in the environment is shown. Figures 7-5 and 7-6 contain snapshots which are taken from different angles to show that the agents are keeping a distance of 2 m above the surface.

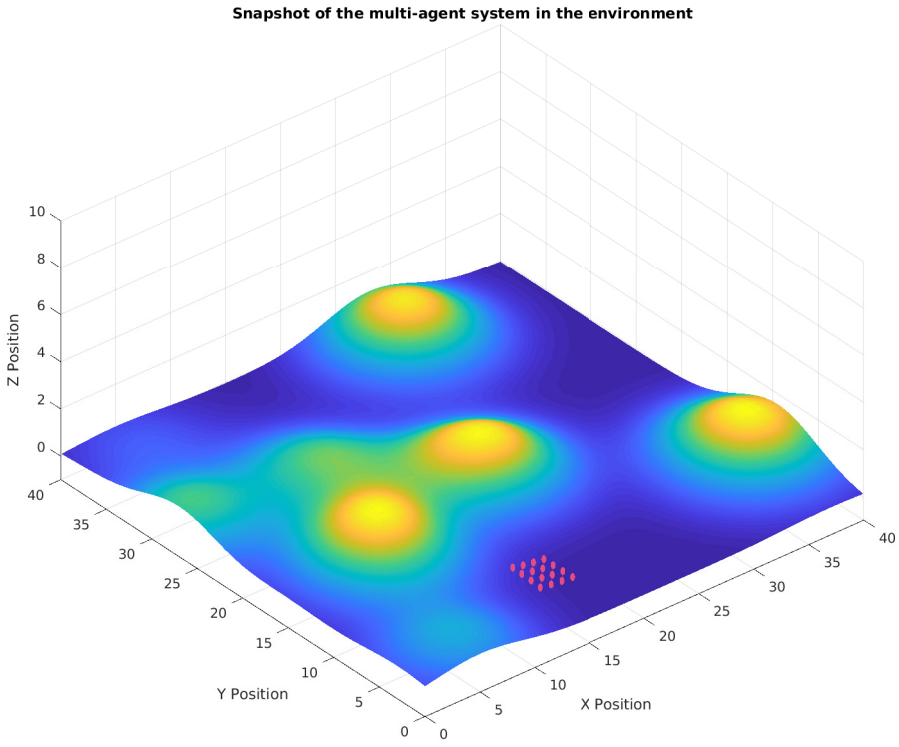


Figure 7-4: Snapshot of the multi-agent system in the environment

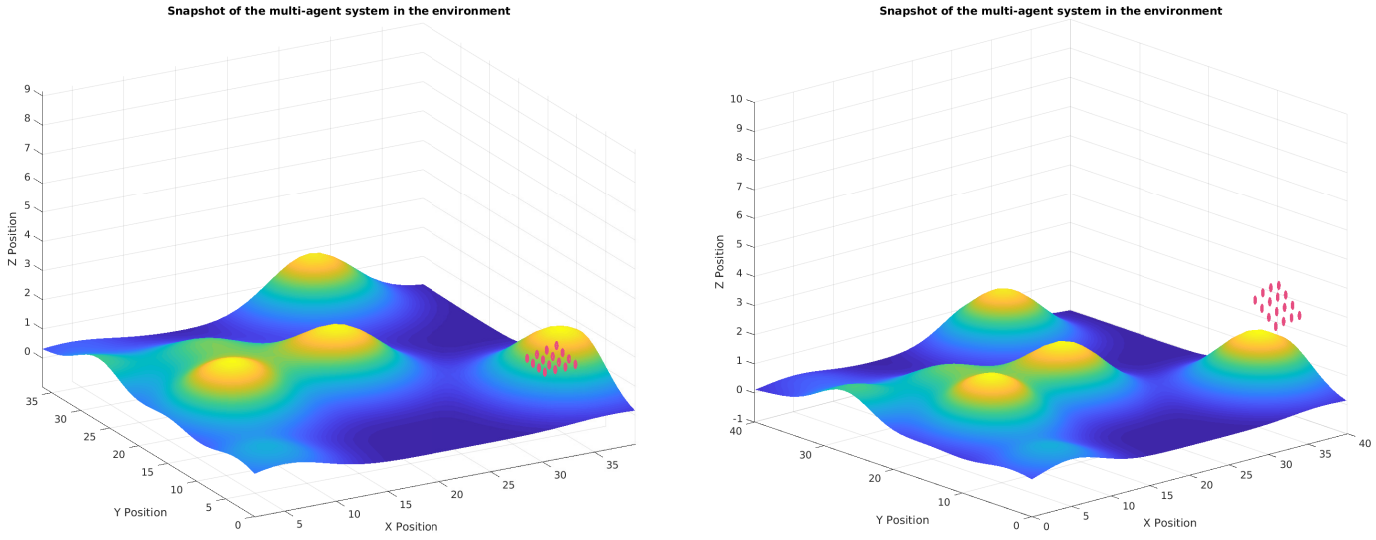


Figure 7-5: Snapshots of the multi-agent system, seen from different angles (1)

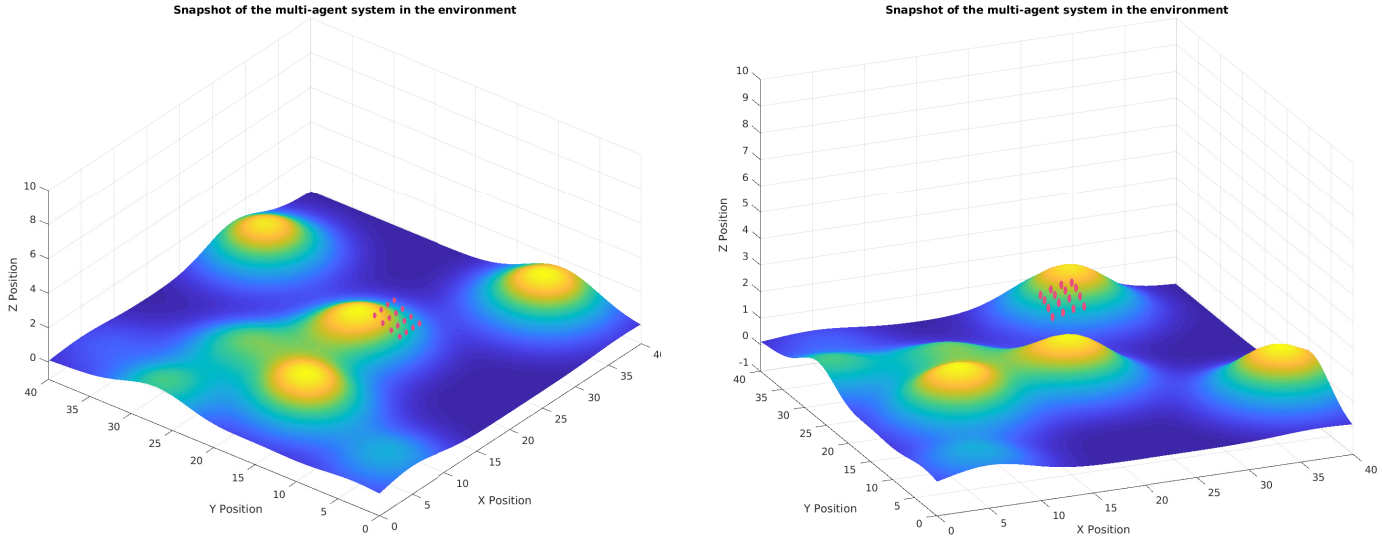
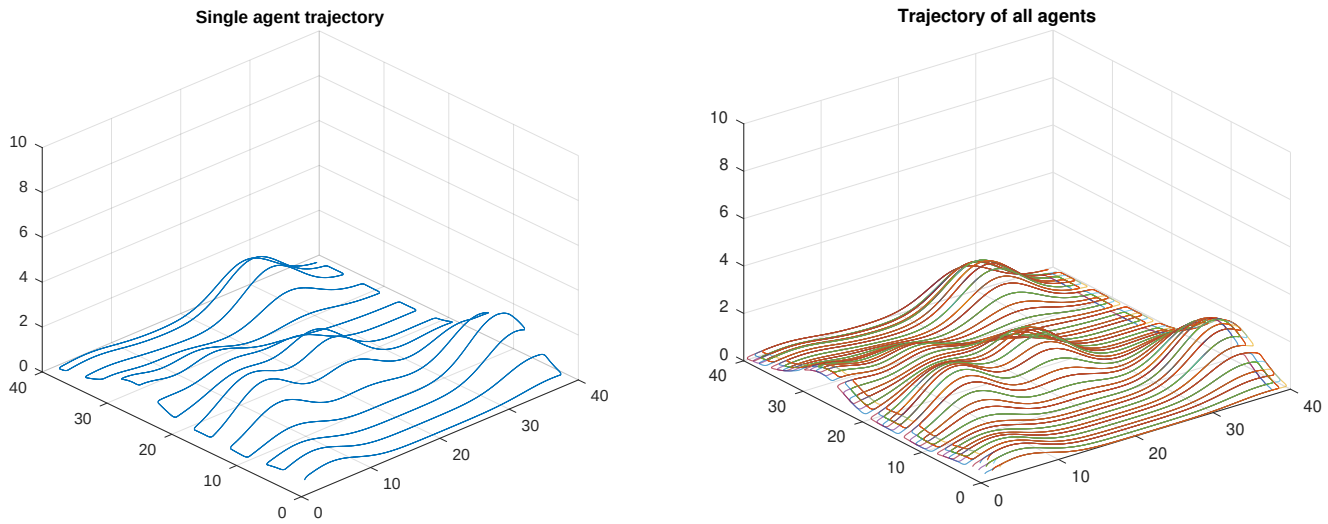


Figure 7-6: Snapshots of the multi-agent system, seen from different angles (2)

7-2-1 Same inter-agent distance

To start with, first the inter-agent distance as well as the communication radius are kept the same for each scenario. This results in an increased d_{width} when more agents are being used.

In Figure 7-7 the complete trajectory of a single agent is shown as well as a view of the complete trajectory of all the agents. These trajectories are based on the estimated absolute positions of the agents.



(a) 3D-Single agent trajectory

(b) 3D-Trajectory of all agents

Figure 7-7: Plots of a single trajectory of one agent, Figure (a), and of all agents, Figure (b)

In Figure 7-8 the convergence of the error is shown for a multi-agent system of 9 agents. As can be seen the error converges to zero, meaning that the estimated positions become closer to the real positions.

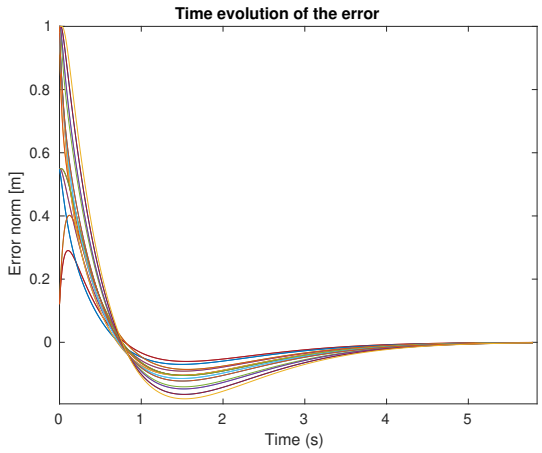


Figure 7-8: Convergence of the error for a multi-agent system of 9 agents

The reconstructed map, the ground-truth map and the difference between these two maps are shown in Figure 7-9 for the 3D view and in Figure 7-10 for the top-down view. Furthermore, the results of the evaluation metrics are presented in Table 7-2.

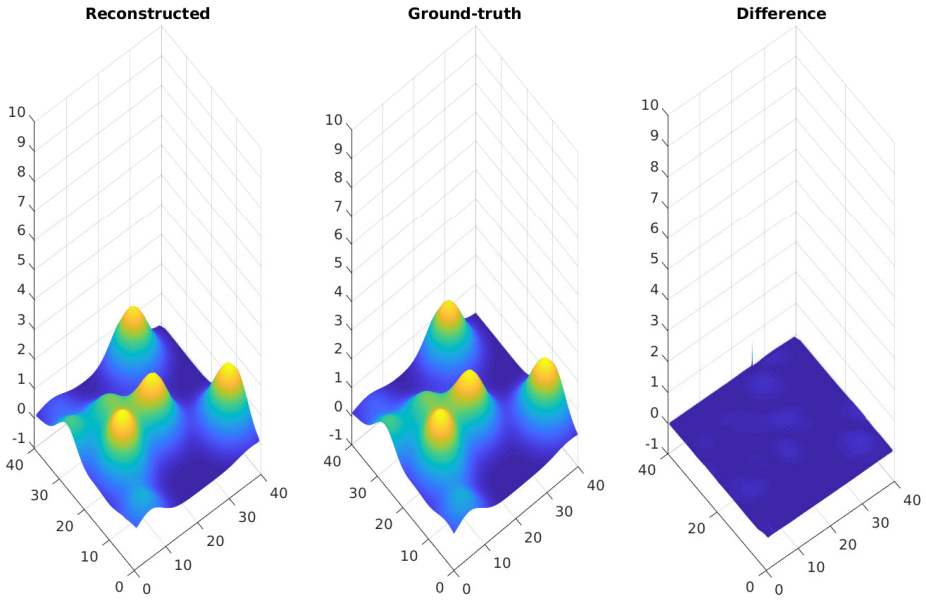


Figure 7-9: 3D view of the reconstructed map, the ground-truth map and difference between these two maps for a multi agent system of 9 agents with the same inter-agent distance

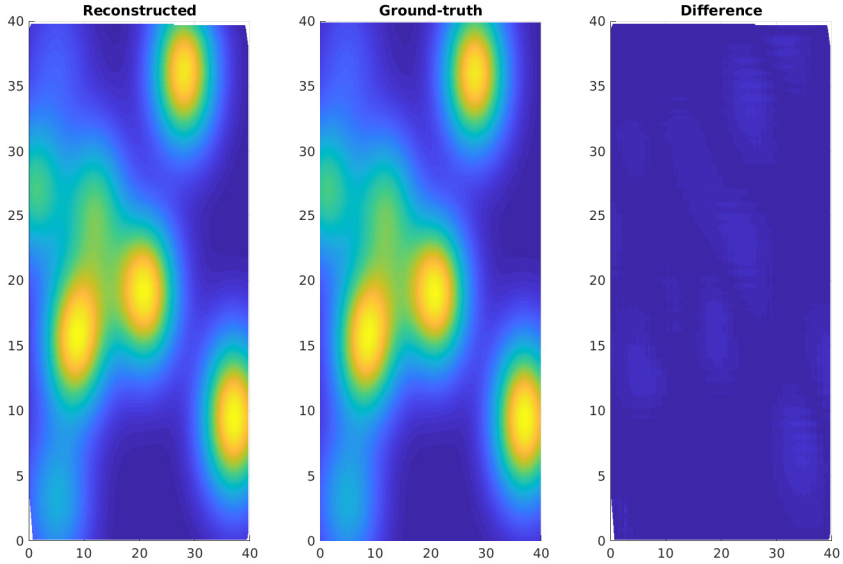


Figure 7-10: Top-down view of the reconstructed map, the ground-truth map and difference between these two maps for a multi agent system of 9 agents with the same inter-agent distance

Table 7-2: Results when using same inter-agent distance, $d_{inter} = 1$ m, $\gamma = 5$

N	4	9	16	25	36	64	81	100
TTT	429.65 s	307.49 s	227.93 s	186.74 s	156.14 s	147.62 s	135.36 s	149.47 s
TC	4.89 s	5.87 s	6.01 s	7.22 s	15.41 s	30.72 s	35.39 s	51.87 s
TTSA	424.76 s	301.62 s	221.92 s	179.52 s	140.73 s	116.90 s	99.97 s	97.60 s
V_{err}	1.84%	1.856%	1.831%	1.882%	1.903%	2.11%	2.46%	2.58%
d_{width}	1 m	2 m	3 m	4 m	5 m	7 m	8 m	9 m

From the results presented in Table 7-2 it can be concluded that, in case a same inter-agent distance is kept, an increase of agents in the multi-agent system leads to a decrease in the total time spent to scan the whole area (TTSA). On the other hand, the convergence time (TC) for the estimated positions to converge to the true positions increases when the number of agents is increased. An optimum for the Total Time Taken (TTT) can be seen when 81 agents are being used. In the case of using more agents the increase in convergence time is larger than the decrease in TTSA, leading to a longer total time.

Furthermore, it can be seen that the volume error of the difference between the ground-truth map and the reconstructed map also slightly increases when the number of agents increases. This is due to the fact that the multi-agent system starts scanning the area when the norm of the error has come within a 0.05 m margin. With a higher number of agents the error keeps slightly oscillating after this point is reached. This is shown in Figure 7-11 where the convergence of the error is presented for the multi-agent system consisting of 64 agents, resulting in a higher V_{err}. This volume error could be decreased by letting the multi-agent system wait for a longer period after the convergence time, before they actually start with the scanning of the area. However, this would also lead to a longer Total Time Taken (TTT).

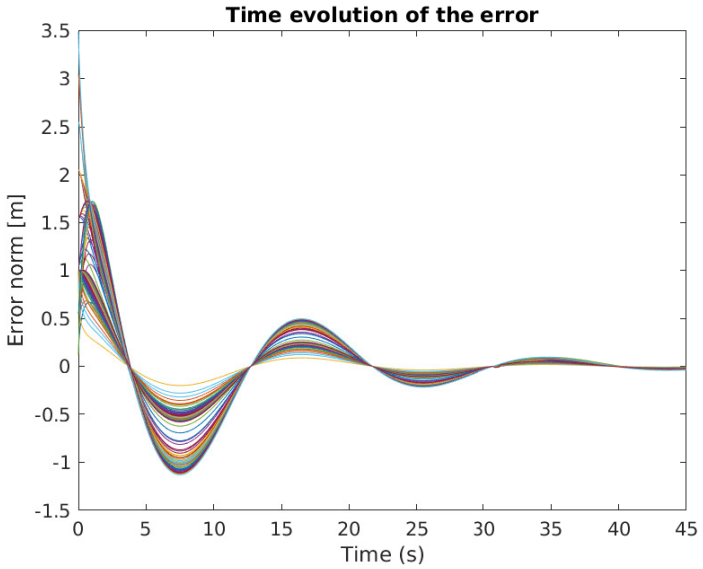


Figure 7-11: Oscillation of the error for a multi-agent system of 64 agents

7-2-2 Same grid-width

In the following part the number of agents increases while the same grid-width is kept, resulting in a smaller inter-agent distance every time the number of agents increases. The communication between the agents is still limited to only the nearest neighbors, meaning $r_{comm} = d_{inter}$.

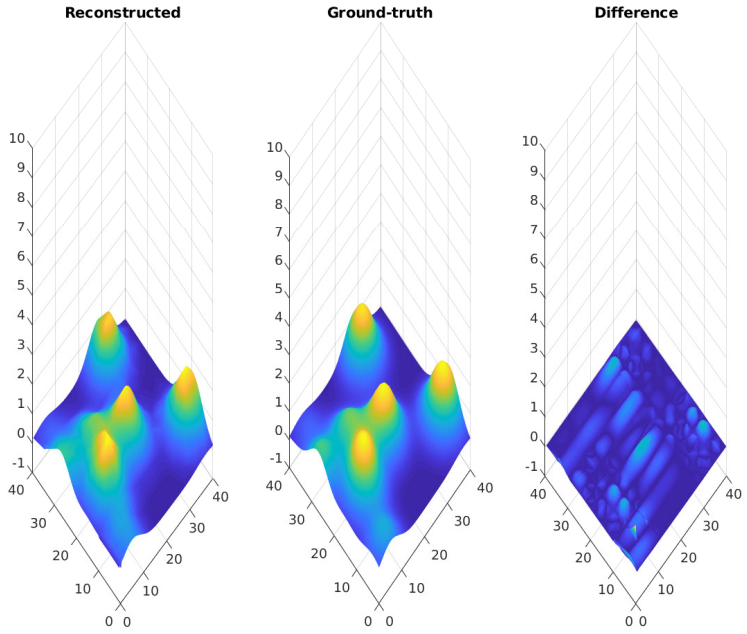


Figure 7-12: 3D view of the reconstructed map, the ground-truth map and difference between these two maps for a multi agent system of 9 agents with the same grid-width

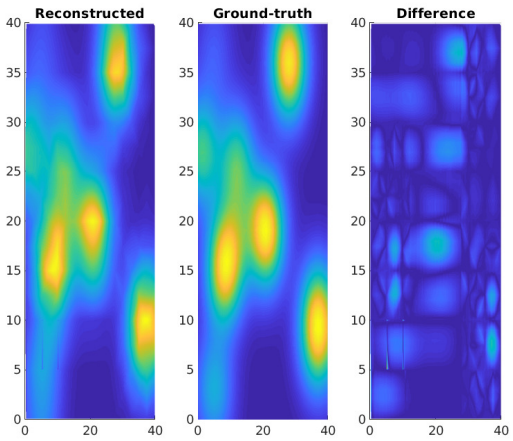


Figure 7-13: Top-down view of the reconstructed map, the ground-truth map and difference between these two maps for a multi agent system of 9 agents with the same grid-width

The evaluation metrics for the case-study where the same grid-width is used, are presented in Table 7-3.

Table 7-3: Results when using same grid-width, $d_{width} = 10$ m, $\gamma = 5$

N	4	9	16	25	36	64	81	100
d_{inter}	10 m	5 m	3.3 m	2.5 m	1.96 m	1.5 m	1.23 m	1.1 m
TTT	80.38 s	81.85 s	85.81 s	88.37 s	91.40 s	107.32 s	112.60 s	129.13 s
TC	5.34 s	6.50 s	10.20 s	12.57 s	15.42 s	30.72 s	35.39 s	51.87 s
TTSA	75.04 s	75.35 s	75.61 s	75.80 s	75.98 s	76.60 s	77.21 s	77.26
V_{err}	16.74%	6.69%	3.59%	2.58%	2.34%	2.27%	2.58%	2.68%

From the results it is evident that, by increasing the number of agents, which decreases the inter-agent distance, the volume error between the volumes of the ground-truth map and the reconstructed map decreases. An optimum for the volume error is shown for a system with 64 agents. The increase in volume error with 81 and 100 agents comes from the same oscillating effect described previously. In Figures 7-12 and 7-13, the different maps are shown for the case where 9 agents are used with an inter-agent distance of 5 m. As can be seen on the difference map, the trajectory of the agents is clearly visible. Due to the large inter-agent distance, only a smaller total area of the map is covered by the agents. This results in a larger error of the volume, due to the fact that the points between the trajectories have to be interpolated.

Furthermore, the results show that the convergence time of the observer increases as the number of agents increase, just as was presented in Table 7-2. This can be explained by the fact that there are more agents between the far-most agent, the agent which is the most distant seen from the agent with a connection to the external environment, and the agent with a connection to the external environment when the number of agents increases. As the estimation of the absolute position of the far-most agent depends on more intermediate estimates, it will take longer for this agent to obtain its true absolute position and therefore this results in a longer overall convergence time.

7-3 Improving convergence time of the state estimation for larger number of agents

By looking at the previous case-studies some promising results are obtained. The conclusion that can already be drawn is that if a smaller inter-agent distance is used, the error between the ground-truth map and the reconstructed map becomes smaller. Besides that, an increased number of agents may be faster in covering the whole area if the convergence time of the observer for larger number of agents can be shortened. Therefore, in the following section attempts are made to bring down this convergence time.

7-3-1 Increasing communication radius

First, the communication radius of the agents will be changed to see what influence this has on the convergence time of the estimated states. This will be tested for a multi-agent system of 36 agents where the grid-width is kept the same, so that the agents have an inter-agent distance of 1.96 m. The maximum number of neighboring agents that the communication radius will enclose is denoted by N_{in} and is based on the number of in/out going arcs, as can be seen in Figure 7-14, where the situation is shown for a r_{comm} of 2.9 m.

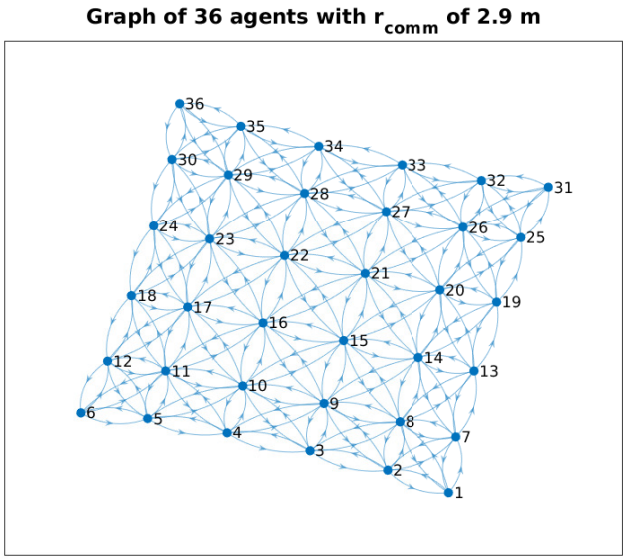


Figure 7-14: Communication graph for a formation with an inter-agent distance of 1.96 m and a communication radius of 2.9 m

Table 7-4: Results of an increased communication radius, with N = 36, an inter-agent distance of 1.96 m and $\gamma = 5$

r_{comm}	2.1 m	2.9 m	4 m	9 m	15 m
N_{in}	4	8	12	35	35
TTT	91.40 s	87.49 s	86.75 s	81.44 s	81.49 s
TC	15.42 s	11.51 s	10.77 s	5.46 s	5.51 s
TTSA	75.98 s	75.98 s	75.98 s	75.98 s	75.98 s
V_{err}	2.09%	2.11%	2.09%	2.09%	2.10%

From the results presented in Table 7-4, it can be concluded that an increase in communication radius will significantly decrease the convergence time of the estimated states to their absolute values. A small increase of 0.8 m in radius already leads to a decrease of convergence time of $\frac{15.42-11.51}{15.42} \approx 25\%$, while four extra agents are included within the communication radius, i.e. all agents within a square around the main agent. A decrease of $\frac{15.42-5.51}{15.42} \approx 64\%$ of the convergence time is seen when the communication radius is increased even further. As a first thought this may seem to have a positive outcome on the network-decentralized SLAM algorithm. However, the increase of communication radius comes at the price of more inter-agent communication, leading to the question if one of the characteristics of the network-decentralized SLAM algorithm, namely the limitation of data sharing, is still valid when the communication radius is increased, for example, in the case where all other agents are within the communication radius of one agent. Moreover, for a real-world implementation it is doubtful if this approach will work as it may be harder to communicate with other agents that are not in a clear line of sight from the main agent, let alone to be able to get a reliable relative distance measurement between the agents, because of the interference between communication and measurement signals. Therefore, we typically choose to restrict the communication radius to agents that are in a direct encirclement of the main agent, see Figure 7-15.

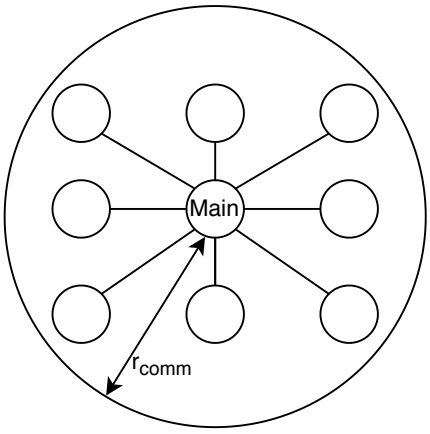


Figure 7-15: Communication graph for a communication radius, r_{comm}, which encloses all nearby agents

7-3-2 Dynamic communication radius

To restrict the data sharing even more, that is to only the nearest neighbors, we can use a dynamic communication radius, which changes after the estimated states have converged to their absolute values. The results of this concept are shown in Table 7-5. The initial communication radius is adjusted, where after convergence of the states the communication radius is set back to only enclose the nearest neighbors, i.e. $r_{comm} = 2.1$ m in this case.

Table 7-5: Results of a dynamic communication radius, with N = 36, inter-agent distance of 1.96 m and $\gamma = 5$

Initial r_{comm}	2.9 m	4 m	9 m	15 m
Computation time keeping same radius	617.08 s	687.75 s	697.38 s	617.37 s
Final r_{comm}	2.1 m	2.1 m	2.1 m	2.1 m
Computation time dynamic radius	426.45 s	443.33 s	442.64 s	430.21 s
TTT	87.49 s	86.75 s	81.61 s	81.49 s
TC	11.51 s	10.77 s	5.63 s	5.51 s
TTSA	75.98 s	75.98 s	75.98 s	75.98 s
V_{err}	2.13%	2.09%	2.11%	2.10%

From the results can be seen that when using a dynamic communication radius the computation time is reduced relative to keeping a fixed communication radius. This is as one would expect, because of the fact that there is less inter-agent communication. What is remarkable, however, is the fact that enclosing all agents in the communication radius has approximately the same computation time as enclosing only the nearest neighbors. An explanation for this could be that when enclosing all agents, also the agent which has a connection to the external environment is enclosed, leading to a more direct relative measurement with the absolute position. As stated before the last option (fully connected agents) is probably not desired because it will take away the decentralized characteristics. Thus, by using a dynamic communication radius the convergence time can be reduced, leading to a faster Total Time Taken (TTT) for larger multi-agent systems, while keeping the data sharing between agents to a minimum.

7-3-3 Changing location and number of agents with absolute position

Another approach that may influence the convergence time is the location in the formation of the agent which is connected to the external environment. Furthermore, we can check whether the number of agents with a connection to the external environment can also influence the convergence time. First, a formation of 36 agents is used where the location of the agent with a connection to the external environment in the formation is switched between three positions 1, 2 and 3 denoted by the corresponding numbers in the agents as shown in Figure 7-16. A grid-width of 10 m, an inter-agent distance of 1.96 m and a communication radius of 2.1 m are used during the simulation.

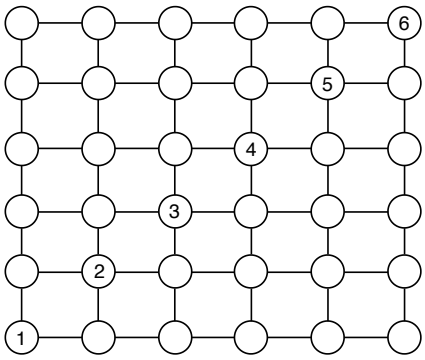
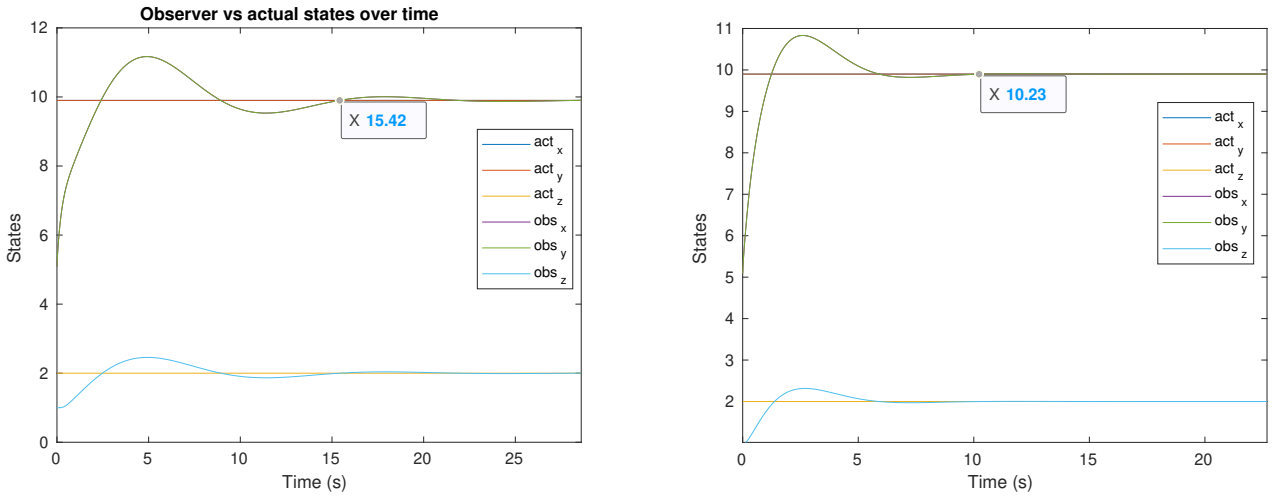


Figure 7-16: Position of the agent with a connection to the external environment

Table 7-6: Results of using different agents in the formation with an external connection

Position of agent with external connection	1	2	3
TTT	91.40 s	87.26 s	86.21 s
TC	15.42 s	11.28 s	10.23 s
TTSA	75.98 s	75.98 s	75.98 s
V_{err}	2.09%	2.09%	2.09%

From the results presented in Table 7-6 can be concluded that the convergence time is reduced when the position of the agent with an external connection to the environment is placed more to the center of the formation. This can be explained due to a shorter path of this agent to the most remote agents, i.e. a higher closeness centrality, $C(x)$. This is a measure of centrality for an agent in a network. The closeness centrality is calculated by the reciprocal of the sum of the length of the shortest paths between the agent with the connection to the external environment and all other agents in the formation. This results in a closeness centrality for the agent with a connection to the external environment in position 1 of $C(1) = \frac{1}{180}$, for this agent in position 2 of $C(2) = \frac{1}{132}$ and for this agent in position 3 of $C(3) = \frac{1}{108}$. For the agent with an external connection on position 1, there are at least 9 intermediate agents between this agent and the far-most agent. For position 2 and 3 there are respectively 7 and 5 intermediate agents. Thus, less intermediate agents lead to a higher closeness centrality and a lower convergence time. In Figure 7-17 this effect is also visualized, where in Figure 7-17a the convergence of the observed states to the true states can be seen for the most remote agent seen from position 1. In Figure 7-17b this is shown for the far-most agent seen from position 3.



(a) Externally connected agent in position 1

(b) Externally connected agent in position 3

Figure 7-17: State convergence of the most remote agent, when the externally connected agent is located in position 1, Figure (a), and when it is located in position 3, Figure (b)

Changing number of external connections

In the following part the number of external connections is increased.

Table 7-7: Results of multiple agents with an external connection

Position	1 & 2	1 & 3	1 & 6	1 & 2 & 3	1 & 3 & 6	1-6
TTT	87.57 s	85.91 s	86.69 s	85.76 s	84.41 s	83.95 s
TC	11.59 s	9.93 s	10.71 s	9.78 s	8.43 s	7.97 s
TTSA	75.98 s	75.98 s	75.98 s	75.98 s	75.98 s	75.98 s
V_{err}	2.09%	2.09%	2.09%	2.09%	2.09%	2.09%

From the results presented in Table 7-7 it can be concluded that using more agents with an external connection will reduce the convergence time of the estimation error. Due to more available information about the absolute positions of the agents the network-decentralized position estimation will converge faster for the other agents without such a connection.

Based on the previous results it can be concluded that when it is possible to actively position the agent with known absolute position more in the middle of the formation, the convergence time and thus the total time can be reduced. Besides, using more agents with external connections also leads to a smaller convergence time for the system, and furthermore it confers more robustness to the system (in case of failure of one of the externally connected agents, the agent network remains externally connected due to the presence of other externally connected agents, and convergence is still ensured, in Section 7-5 this will be discussed in more detail). Thus, if more information about the absolute positions is available then it should be used such that the overall duration of the process can be reduced. The agents should thereby be placed in the formation in an optimal way. How to place a single agent or several agents in the middle of a formation or in an optimal way within the formation, so it can be used for a real-life application, has not been further researched in this thesis and is left for future work.

7-4 Formation producing

To test the multi-agent system in producing the formation, the multi-agent system is split up in two clusters with no initial communication between these clusters, as can be seen in Figure 7-18.

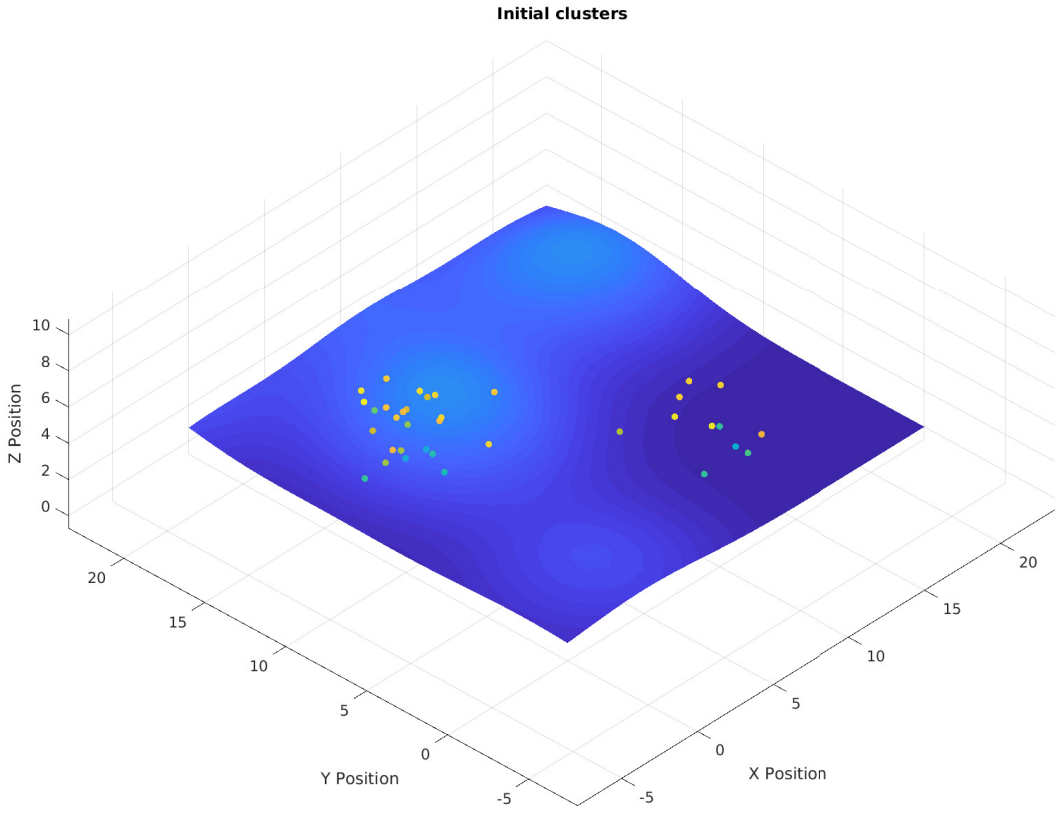


Figure 7-18: Snapshot of the multi-agent system split in two clusters

In each cluster an agent with a connection to the external environment is placed to make sure the network-decentralized position estimation works. In addition, every agent is placed at a random location within the cluster with a minimum distance between the agents such that each agent is initially spaced enough. Every agent has a randomly appointed target location within the to be produced formation. Starting in the different clusters, the agent will have to go to its target location without causing collisions, using the collision avoidance and formation producing protocols as described in Chapter 6. An already partially formed formation can be seen in Figure 7-19.

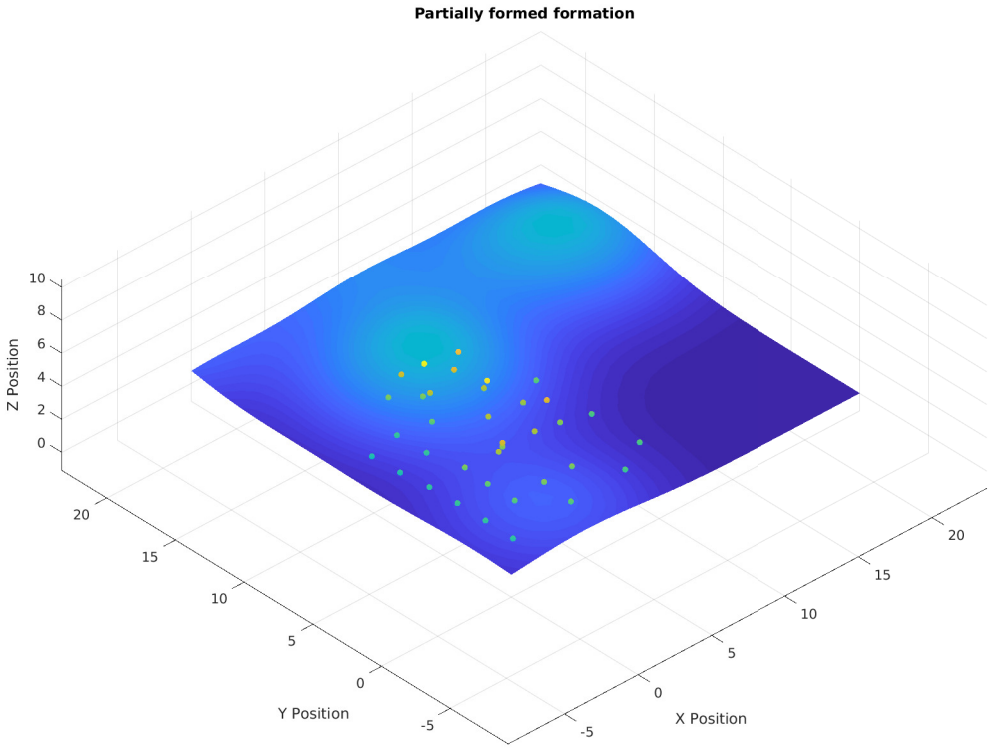


Figure 7-19: Snapshot of the multi-agent system in a partially formed formation

The formation producing has been tested for a multi-agent system of 36 agents with different inter-agent distances in the final formation. The average has been taken over 20 runs, where each time the agents have been placed, randomly, in the clusters and a random final position in the formation has been assigned. No communication radius has been used in this situation; the communication has been limited to the three nearest neighbors to make sure that each agent in a cluster is externally connected via a path. The used parameters for the collision avoidance and formation production are defined in Table 7-8. The extensive tuning of these parameters is not considered the main focus for this thesis, the values are therefore taken from a previous thesis. To see how these parameters are tuned, the reader is referred to [61].

Table 7-8: Parameter settings for formation producing

Parameter	Description	Value
α	Rotation angle of the avoidance vector	Varies on-line
R_{cz}	Radius of outer collision detection region	Varies on-line
c	Constant defining R_{cz}	0.45
β	Constant defining R_{cz}	1.3
θ	Window size of outer collision detection zone	120°
R_{pref_dist}	Radius of preferred distance from obstacle intersection	$R_{cz} - 0.05$
R_{emer}	Radius of emergency detection zone	0.35 m

Table 7-9: Results of exploiting the collision avoidance and formation producing protocol

d_{inter}	1.0 m	1.2 m	1.5 m	2 m
Number of collisions	0	0	0	0
Average producing time	N/A	75.82 s	31.94 s	29.77 s
Formation formed	0/20	17/20	19/20	20/20

The results for the formation producing are presented in Table 7-9. The average producing time increases and the number of completed formations reduces when smaller inter-agent distances between the desired formation location are used. This is due to the fact that live-lock situations are more frequent as the inter-agent distance decreases. For the agents it becomes harder to get in formation due to the smaller space which is in between the agents that are already at their final positions in the grid. This is also the reason why, with an inter-agent distance of 1.0 m, the formation cannot be formed as two agents in position have a combined span-width of 0.7 m, each emergency detection zone has 0.35 m radius. For a third agent to fit through these two agents a minimal inter-agent distance of 1.05 m is required for these settings otherwise a live-lock situation will occur. This problem can be solved in different ways. One solution is to first produce a formation with a larger inter-agent distance and then as a whole formation decrease the inter-agent distances. Other solutions, which are not further implemented, are, for instance, for the agents to also be able to avoid in the Z-direction when the agents are already on the same height and are trying to avoid each other in the XY-plane.

It might happen that one agent has a destination assigned in the middle of the formation, while all agents around that destination have already reached their own target location. To overcome this live-lock and to reduce the total distance the agents have to travel, the target destination for each agent could be interchanged by the agents through solving local optimization problems so the total distance travelled will be minimized [60]. As optimal formation producing is not considered a main focus of this thesis, it is left for future research to work out this concept.

7-5 Robustness test

Some different tests have been performed to see what would happen when some of the agents, for example, lose the ability to communicate or when they completely stop working. First, we test what happens when the agent with an external connection loses this connection for a few seconds, before and after the rest of the agents have estimated their absolute positions. These tests have been performed for a multi-agent system of 16 agents, with a grid-width of 10 m and an inter-agent distance of 3.3 m.

First the agent with an external connection loses this connection at 3.5 seconds and regains it at 8.5 seconds. Then, it is found that the other agents are not able to estimate their absolute positions anymore during the time that the connection with the external environment is lost, keeping the same error between the actual and estimated position. After the connection is restored all the other agents are able to estimate their absolute positions again. This is also visualized in Figure 7-20 where the error converges to 0 after the connection is restored. It confirms the theory in Chapter 4 that an external connection with the environment is required in order for the network-decentralized position estimation to function properly.

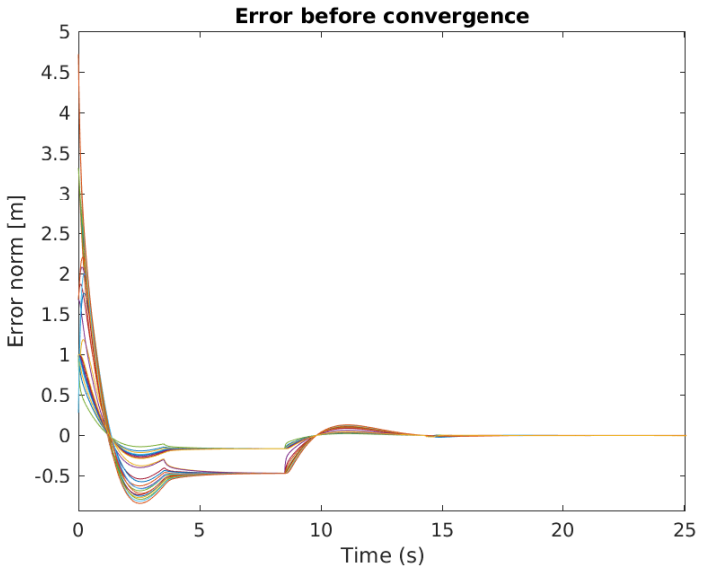


Figure 7-20: Error of the system where the connection with the external environment is lost before the convergence of the states

In the second case the agent with the external connection loses its ability to communicate after 3.5 s after the convergence time (10.22 s) and regains it 5 seconds later. From Figure 7-21 it can be concluded that after convergence, the system is robust against a short communication loss with the external environment and is capable of continuing the scanning of the area without degradation of the previous established volume error for this scenario.

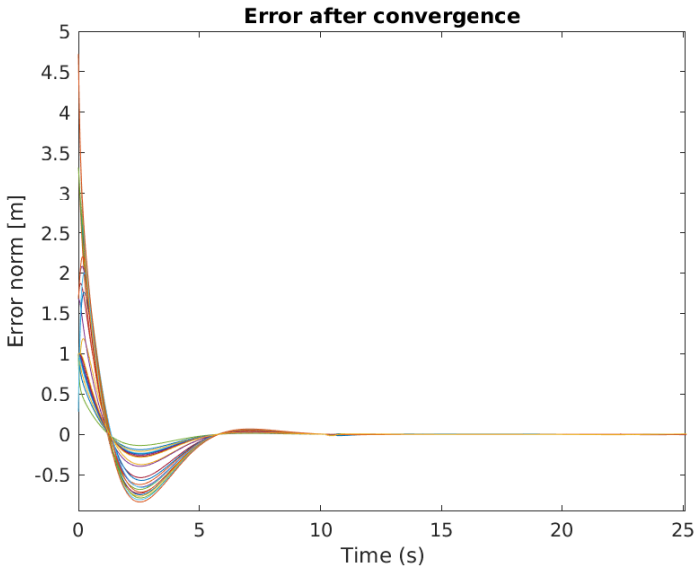


Figure 7-21: Error of the system where the connection with the external environment is lost after the convergence of the states

7-5-1 Disconnecting the agent with an external connection from the rest of the formation

The agent with an external connection is disconnected from the rest of the formation after the other agents have estimated their absolute positions. From Figures 7-22 and 7-23 it can be seen that as the rest of the formation continues to scan the area, the network-decentralized position estimation has trouble with keeping a correct estimate of the actual position due to the lack of an external connection in the formation, leading to an increase of volume error from 3.59%, of an externally connected formation, to 7.52%, of an externally disconnected formation. Therefore, when the agent with an external connection is disconnected from the formation, completely or for a longer duration, we may think of establishing a connection to the external environment via another agent in the formation.

From the previous test it can be concluded that if the communication of the agent with the external connection can be restored in time, then the other agents are able to keep scanning the environment without degradation of the volume error. When the agent with an external connection is completely disconnected, the rest of the formation can still complete the scanning of the environment, however, this will lead to an increase in the volume error.

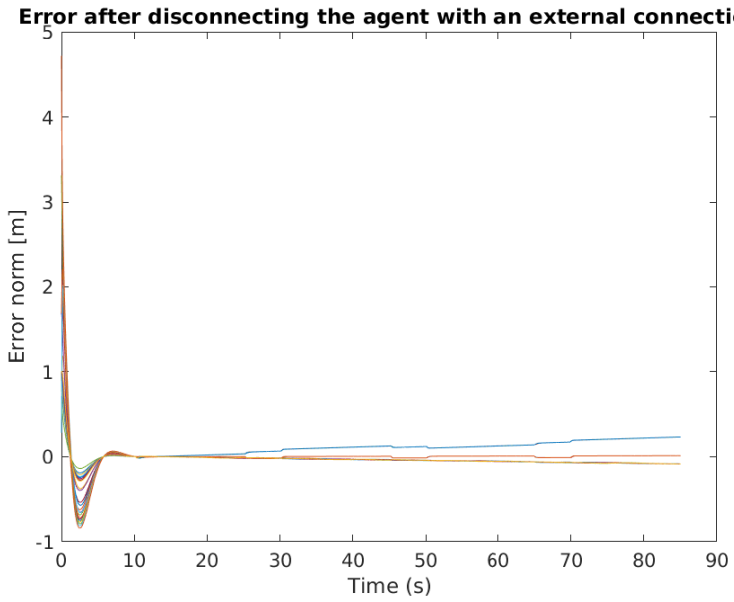


Figure 7-22: Error of the system where connection with external environment is lost completely

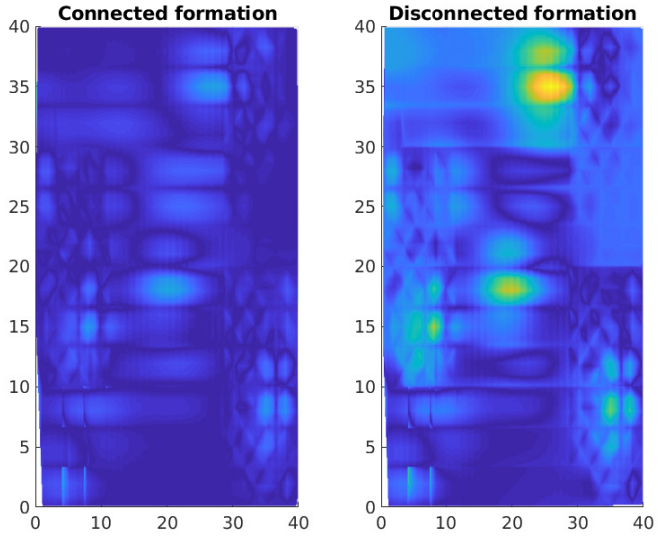


Figure 7-23: Difference map of the externally connected formation and of the externally disconnected formation

7-5-2 Losing multiple agents which are not connected to the external environment

In the following simulations, we test what happens when multiple other agents are disconnected from the formation. In the simulations a path from every remaining agent in the formation to the agent with the external connection has been ensured.

From the simulations it can be seen that the remaining agents in the formation are able to estimate their absolute positions throughout the whole scanning of the environment as long as there is a path to the agent with a connection to the external environment. The reconstruction of the map can be ensured, but the completeness and accuracy with which this can be achieved depends on which agents in the formation are disconnected. For example, when the right lower triangle of the formation disconnects the following reconstructed map is obtained, see Figure 7-24. It shows that a part of the map cannot be reconstructed due to the lack of data points, but the rest of the environment can be reconstructed with the remaining agents in the formation.

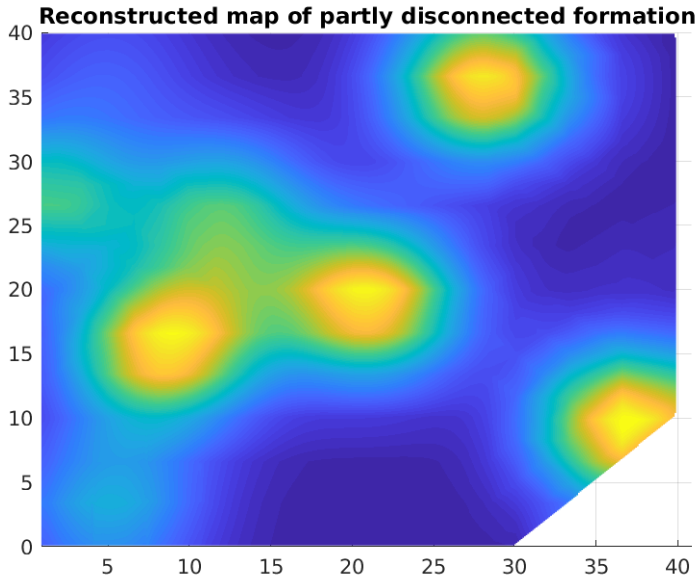


Figure 7-24: Reconstructed map where the right lower triangular part of the formation is disconnected

7-6 Noise addition

In the previous presented simulations, there was no measurement noise added to the relative distance measurements between the agents. To make the simulations more realistic a white Gaussian measurement noise, w, is added to the system. The dynamics of the whole multi-agent system become in this way:

$$\begin{aligned} \dot{x} &= \mathcal{A}x + Bu \\ y &= \mathcal{C}x - w \\ \dot{z} &= \mathcal{A}z + Bu - Ly + L\mathcal{C}z \end{aligned} \tag{7-5}$$

Where the dynamics of the estimation error $e = x - z$ are now:

$$\dot{e} = \mathcal{A}e + L(y - \mathcal{C}z) = (\mathcal{A} + L\mathcal{C})e + Lw \tag{7-6}$$

Based on the techniques presented in Table 5-1, the measurement noise was determined and tested for different settings. Most of the techniques have an average error for relative measurements below 0.2 m and almost all have one below 0.4 m. Therefore, it was simulated how the network-decentralized SLAM works, based on noise levels with zero mean and different standard deviations in a range of 0.05 m to 0.4 m. This was done for a multi-agent system consisting of 36 agents, an inter-agent distance of 1.96 m and a grid-width of 10 m. It must be noted that the Time to Convergence norm had to be enlarged to a norm of 0.1 m and 0.5 m of the true value for the cases where noise was added for a standard deviation below 0.2 m and above 0.2 m respectively, otherwise the formation was not able to start scanning the area.

Table 7-10: Results for the network-decentralized SLAM with different relative measurement noise

Standard deviation of the noise	0 m	0.05 m	0.1 m	0.2 m	0.3 m	0.4 m
TTT	91.40 s	101.25 s	110.06 s	122.85 s	132.32 s	161.60 s
TC	15.42 s	19.76 s	27.71 s	33.82 s	41.89 s	51.23 s
TTSA	75.98 s	81.49 s	82.35 s	89.03 s	90.43 s	110.37 s
V_{err}	2.09%	2.92%	4.81%	6.24%	7.19%	8.54%

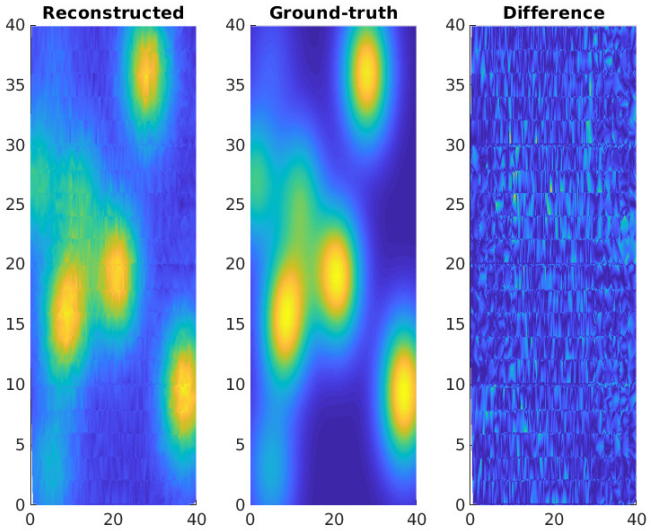


Figure 7-25: Top-down view of the reconstructed map, the ground-truth map and the difference map for a multi agent system of 36 agents and a measurement noise with a standard deviation of 0.30 m

The results for different deviations of measurement noise can be seen in Table 7-10. Furthermore, the reconstructed, ground-truth and the difference map are shown in Figure 7-25 for a noise with a standard deviation of 0.3 m. From the results it can be concluded that with an increase of noise the various timings as well as the volume error increase. The increase of Convergence Time can be explained because the noise makes it harder for the error $e = x - z$ to converge. The increase of volume error is a logical aftereffect of the noise as the estimated absolute position is affected by higher uncertainty. With a standard deviation of 0.3 m the reconstructed map still resembles the ground-truth map in a good way, the different hills are still clearly distinguishable. What is less obvious is why the total time to scan the area also increases. This is a consequence of the fluctuations of the estimated absolute position. In this way the reference error for the trajectory fluctuates as well, meaning that it is possible for the agents to come to a stand-still, until the absolute position has been estimated more accurately. Overall, it can be concluded that even with noise the multi-agent system is still capable of reconstructing an accurate map, but with higher measurement errors it will take longer to scan the area.

Chapter 8

Conclusions and Future work

8-1 Conclusions

In this thesis a novel, fully autonomous, network-decentralized Simultaneous Localization and Mapping algorithm for a multi-agent system was presented. The agents starting in two clusters are able to merge and produce one single formation in order to scan an unknown environment. This is achieved via a network-decentralized collision avoidance and formation producing protocol. When the initial inter-agent distance in the formation is set to be large enough, the agents are able to produce the formation without any collisions. In this way, the occurrence of live-lock situations that would prevent the formation from being achieved are avoided. Furthermore, it was shown that, by using a smaller inter-agent distance, a smaller error of the difference between the volumes of the ground-truth map and of the reconstructed map can be obtained. Therefore, it is important to reduce the inter-agent distance even further after the formation is formed. By using a larger number of agents the unknown area can be scanned more efficiently regarding the total time it takes to scan the area. However, using more agents has a negative effect on the convergence time for the estimated absolute position to converge to the actual absolute position. Two different approaches were presented to overcome this effect.

First, a dynamic communication radius was introduced. By increasing the communication radius to not only include the nearest neighbors, but also the agents within a direct line of sight, the convergence time can already be reduced by $\approx 25\%$. By increasing the initial communication radius even more the convergence time can be reduced even further, but this will take away one of the decentralized characteristics of the presented algorithm, namely limited information sharing. A solution to this problem is to scale down the communication radius again, after the state estimation has converged, to only the nearest neighbors, resulting in local interactions with the closest neighbors only.

The second approach that was thought of to reduce the convergence time for a larger group of agents is to have a proper positioning of the agent with a connection to the external environment. By placing such an agent more to the middle of the formation the convergence

time can be reduced significantly. Furthermore, it was shown that using multiple agents with connections to the external environment also had a positive effect on the downscaling of the convergence time.

The usage of multiple agents with a connection to the external environment will also contribute to the robustness of the system. Robustness of the algorithm is ensured as long as there is a path from the remaining agents in the formation to an agent with an external connection. When this externally connected agent loses its ability to communicate for a short period, then a reconstructed map can be obtained without degradation of the volume error. However, if this agent disconnects completely, a new connection with the external environment for the remaining agents of the formation must be established to ensure that no degradation of the volume error occurs. It was shown that the network-decentralized SLAM algorithm is robust against failure of multiple agents in the formation, but the completeness and accuracy of the reconstructed map depend on where these disconnected agents are located in the formation.

To have a more realistic view on the performance of the network-decentralized SLAM relative measurement noise was added to the simulations. This resulted in a higher volume error, however, the reconstructed map still resembled the ground-truth map in a clear way.

From this thesis, it can be concluded that the proposed novel network-decentralized Simultaneous Localization and Mapping algorithm can be easily scaled to use more agents so that the exploration and mapping of an unknown environment can be achieved in a faster, more accurate and more robust way.

8-2 Future work

The network-decentralized Simultaneous Localization and Mapping algorithm showed promising results in the simulation environment. It would be interesting to see if this work can be applied to a real-time physical system. First, in the simulation environment the holonomic dynamics of the individual agents should be replaced by non-holonomic dynamics, i.e. using the dynamics of a quadcopter, before testing it on an actual physical system. In Appendix A-2 such a dynamic model for a quadcopter is introduced.

Furthermore, in this thesis different formation structures and different trajectories of the formation to scan the area were not considered. Therefore, it could be that a different formation or a different trajectory is more suited for scanning an unknown area. For future work, it is recommended to test different formations and trajectories in order to see if this could lead to a better optimization of the network-decentralized SLAM.

Besides using different formations and trajectories, it must be investigated how the agent with an external connection can be placed in the middle of a formation, while having no knowledge about other agents except for the ones within its communication radius. On top of that, the producing of the formation can be improved by swapping the target destinations of the agents in the formation, so that the total distance the agents have to travel to produce the formation as well as the time it takes to produce this formation can be reduced. This could, for example, be accomplished by an optimization process which only uses local information of the agents, such that the optimization is performed in a network-decentralized way [60].

Finally, the network-decentralized Simultaneous Localization and Mapping could be extended in order to make it possible for the multi-agent system to produce different sorts of maps. For example, by equipping the agents with different sorts of sensors, it could be investigated whether the localization and mapping by using, for example, landmarks can still be performed via the network-decentralized state estimation.

Appendix A

Appendix

A-1 Hardware and software specifications

For the simulations MATLAB version R2019a was used on a computer with the following specifications:

- CPU: Intel Core i7-3610QM
- RAM: 4GB
- Operating system: Ubuntu 16.04 LTS
- Frequency: 2.30GHz to 3.30GHz

A-2 Quadcopter dynamics

Quadcopters can be modelled as a holonomic system [62, 63], but as these vehicles in reality must roll or pitch in order to acquire motion in the horizontal plane, they are actually non-holonomic. To have a more realistic simulation the quadcopters should be modelled and controlled accordingly [64, 65, 66, 67, 68].

To describe the dynamic model of a quadcopter also a kinematic model must be used. To derive these models, first a few common used assumptions have to be made; the vehicle is a rigid body, symmetric and has its Centre of Gravity (CoG) at the origin of the body frame of the quadcopter. Furthermore, quadcopters are highly non-linear and have 6 Degrees of Freedom (DoF) movement.

A-2-1 Kinematic model

In Figure A-1 the Body Fixed Frame (BFF) as well as the Earth-Fixed Frame (EFF) are shown. The EFF is in reality a fixed point which is defined as the origin of, for example, a simulation environment. It uses the North-East-Down (NED) convention where the positive axes points to the North, East and Downwards respectively. The BFF has its axis aligned as follows: the positive x-axis points towards propeller 1, the positive y-axis to propeller 2, and the positive z-axis to the ground. The translational movement is along these axes. The rotational movement of quadcopters in the x, y and z axes are described by the roll (ϕ), pitch (θ) and yaw (ψ) angles respectively. Furthermore, in Figure A-1 also $\Omega_i \quad \forall i \in \{1, 2, 3, 4\}$ is shown which describes the speed of each propeller.

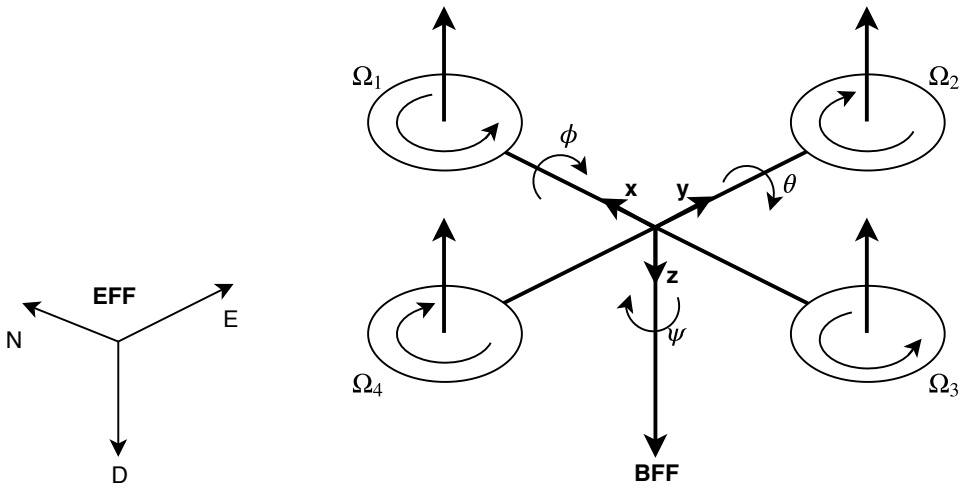


Figure A-1: Configuration of a quadcopter, where BFF and EFF denote the Body Fixed Frame and the Earth Fixed Frame, respectively

The orientation of the quadcopter can be described by the rotation matrix $R_{BE}(\phi, \theta, \psi)$ from the BFF to the EFF. By assuming the order of rotation to be first roll, pitch then yaw, the following rotation matrix can be derived:

$$R_{BE}(\phi,\theta,\psi) = R_x(\phi)R_y(\theta)R_z(\psi) = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi c_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{A-1}$$

where $c_\bullet = cos(\bullet)$ and $s_\bullet = sin(\bullet)$. As $R_{BE} \in SO(3)$, where $SO(3)$ is a special orthogonal group of orthogonal matrices which has three dimensions and a determinant of 1, the transformation from EFF to BFF is simply the transpose of $R_{EB} = (R_{BE})^T$.

To obtain the Euler angle rates ($\dot{\eta} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$) of the quadcopter in the Earth-Fixed Frame, typically an on-board Inertial Measurement Unit (IMU) is used. To relate these with the angular velocity ($\omega = \begin{bmatrix} p & q & r \end{bmatrix}^T$) in the Body Fixed Frame, a transformation is needed as follows:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{A-2}$$

This information can be used to show the behaviour of the quadcopter over time.

A-2-2 Dynamic model

The dynamic model for a quadcopter consists of the linear and angular dynamics, where the linear dynamics are underactuated as a quadcopter can only produce thrust along the z-axis. The dynamic model can be described by the Newton-Euler formalism for a 6 DoF rigid body, which describes the forces and torques experienced by the body:

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + R_{BE}F_B \tag{A-3}$$

$$\begin{aligned} \dot{R}_{BE} &= R_{BE}S(\omega) \\ I\dot{\omega} &= -\omega \times I\omega + \tau_b \end{aligned} \tag{A-4}$$

where $S(\omega)$ is a skew symmetric matrix of ω (the angular body rates).

Linear dynamics

Equation A-3 describes the linear dynamics for the quadcopter which are based on Newton's second law expressed in the EFF, where $r \in \mathcal{R}^3$ ($\{x, y, z\}$) is the position of the quadcopter, m is the mass of the quadcopter and g is the gravitational acceleration $g = 9.81\ m/s^2$. R_{BE} is the rotation matrix shown in Equation A-1 and F_B is defined as follows:

$$F_B = \begin{bmatrix} 0 \\ 0 \\ -(F_1 + F_2 + F_3 + F_4) \end{bmatrix} \tag{A-5}$$

The only non gravitational forces acting on the quadcopter are the thrusts produced by the rotation of the propellers. The negative sign is due to the fact that the thrusts are upwards while the NED convention is used. The rotation matrix is used to transform the thrust forces to the EFF so that the equation can be used for any orientation of the quadcopter.

Simplified formulas for the aerodynamic force F_i and moment M_i produced by the ith rotor can be used [69]:

$$\begin{aligned} F_i &= k_f \Omega_i^2 \\ M_i &= k_m \Omega_i^2 \end{aligned} \tag{A-6}$$

where k_f and k_m are the aerodynamic force and moment constants and Ω_i is the angular velocity of rotor i.

Angular dynamics

Equation A-4 describes the evolution of the angular dynamics. The first formula is expressed in the EFF and describes how the rotation matrix changes over time. The second formula shows Newton's second law for rotation which relates angular acceleration to net external torque on the body. The torque working on the body components of the quadcopter's body frame is given by the following 3x1 vector:

$$\tau_b = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} lk_f(-\Omega_2^2 + \Omega_4^2) \\ lk_f(\Omega_1^2 - \Omega_3^2) \\ k_m(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{bmatrix} \tag{A-7}$$

where l is the moment arm, which is the distance from the origin of the body to the centre of the propellers.

The reason behind deriving the angular dynamics in the Body Fixed Frame and not in the Earth Fixed Frame is to have the inertia matrix independent of time. The inertia matrix I for a quadcopter is a diagonal matrix, as the off-diagonal elements are zero due to the symmetry of the quadcopter and is given by:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{A-8}$$

where I_{xx}, I_{yy} and I_{zz} are the inertia moments about the axes in the BFF.

Following this and filling in Equation A-8 into Equation A-3 results in:

$$\ddot{\phi} = \frac{u_2 + \dot{\theta}\dot{\psi}(I_{yy} - I_{zz})}{I_{xx}} \tag{A-9}$$

$$\ddot{\theta} = \frac{u_3 + \dot{\phi}\dot{\psi}(I_{zz} - I_{xx})}{I_{yy}} \tag{A-10}$$

$$\ddot{\theta} = \frac{u_4 + \dot{\phi}\dot{\theta}(I_{xx} - I_{yy})}{I_{zz}} \tag{A-11}$$

Substituting Equation A-5 in 6-1 leads to:

$$\ddot{r}_x = \frac{u_1(s_\phi s_\psi + c_\phi c_\psi s_\theta)}{m} \tag{A-12}$$

$$\ddot{r}_y = \frac{u_1(c_\phi s_\psi s_\theta - c_\psi s_\phi)}{m} \tag{A-13}$$

$$\ddot{r}_z = g + \frac{u_1 c_\phi c_\theta}{m} \tag{A-14}$$

where the inputs are chosen as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} -k_f & -k_f & -k_f & -k_f \\ 0 & -lk_f & 0 & lk_f \\ lk_f & 0 & -lk_f & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} \tag{A-15}$$

A-3 Quadcopter Control

The model for the quadcopter is nonlinear due to the rotation matrix in the angular dynamics. Through linearization about the hover point of the quadcopter the model can be linearized at the points where the roll and pitch are 0 degrees and the total thrust cancels out the gravitational forces. This leads to the angular velocities of both frames to be equivalent. The linearized closed-loop control for a quadcopter consists of two cascaded PID controllers. The outer PID loop is the position controller, that sets a desired orientation for the attitude controller, which is the inner PID loop. The PID controllers generate the desired control inputs for the quadcopter. This control scheme is shown in Figure A-2.

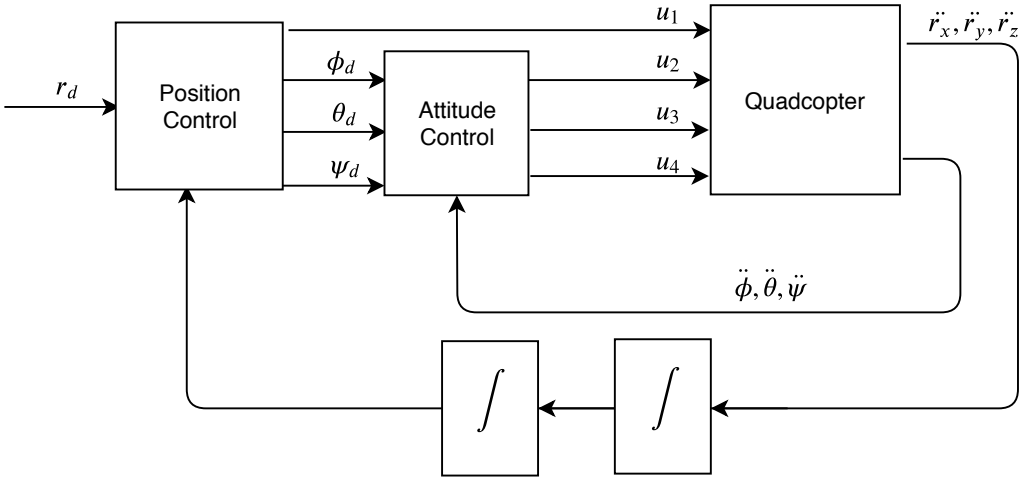


Figure A-2: Cascaded controller structure

Bibliography

[1] M. Ben-Ari and F. Mondada, *Robots and Their Applications*, pp. 1–20. Springer International Publishing, January 2018.

[2] T. Patel, S. Shah, and S. Pancholy, "Long distance tele-robotic-assisted percutaneous coronary intervention: A report of first-in-human experience," *EClinicalMedicine*, vol. 14, September 2019.

[3] W. Golnazarian and E. Hall, "Intelligent industrial robots," September 2002.

[4] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 427–438, February 2013.

[5] P. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI Magazine*, vol. 29, pp. 9–20, March 2008.

[6] A. Jaimes, S. Kota, and J. Gomez, "An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles UAV(s)," in *2008 IEEE International Conference on System of Systems Engineering*, pp. 1–6, June 2008.

[7] M. Schwager, B. J. Julian, M. Angermann, and D. Rus, "Eyes in the sky: Decentralized control for the deployment of robotic camera networks," *Proceedings of the IEEE*, vol. 99, pp. 1541–1561, September 2011.

[8] B. Chen, *A multi-agent based cooperative control model applied to the management of vehicles-trains*. PhD thesis, Université Bourgogne Franche-Comté, February 2017.

[9] F. Derakhshan and S. Yousefi, "A review on the applications of multiagent systems in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 15, no. 5, 2019.

[10] A. Rovira-Sugranes and A. Razi, "Predictive routing for dynamic UAV networks," in *2017 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pp. 43–47, October 2017.

[11] W. Ren and Y. Cao, *Distributed Coordination of Multi-agent Networks: Emergent Problems, Models, and Issues*. Springer International Publishing, January 2011.

[12] M. B. A. Barrat and A. Vespignani, *Dynamical Processes on Complex Networks*. Cambridge University Press, October 2008.

[13] M. Mesbahi and M. Egerstedt, *Graph Theoretic Methods in Multi-Agent Networks*. Princeton University Press, July 2010.

[14] F. Blanchini, E. Franco, and G. Giordano, "Network-Decentralized Control Strategies for Stabilization," *IEEE Transactions on Automatic Control*, vol. 60, pp. 491–496, February 2015.

[15] F. Blanchini, E. Franco, and G. Giordano, "Structured-LMI conditions for stabilizing network-decentralized control," in *52nd IEEE Conference on Decision and Control*, pp. 6880–6885, December 2013.

[16] F. Blanchini, E. Franco, G. Giordano, V. Mardanlou, and P. Montessoro, "Compartmental flow control: Decentralization, robustness and optimality," *Automatica*, vol. 64, pp. 18–28, February 2016.

[17] G. Giordano, F. Blanchini, E. Franco, V. Mardanlou, and P. L. Montessoro, "The Smallest Eigenvalue of the Generalized Laplacian Matrix, with Application to Network-Decentralized Estimation for Homogeneous Systems," *IEEE Transactions on Network Science and Engineering*, vol. 3, pp. 312–324, October 2016.

[18] G. Giordano, *Structural Analysis and Control of Dynamical Networks*. PhD thesis, Università degli Studi di Udine, April 2016.

[19] V. Ugrinovskii, "Distributed robust estimation over randomly switching networks using H-∞ consensus," *Automatica*, vol. 49, pp. 160–168, January 2013.

[20] H. Liu and T. Zhou, "Distributed observer design for networked dynamical systems," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pp. 3791–3796, May 2015.

[21] A. F. Scott and C. Yu, "Cooperative multi-agent mapping and exploration in Webots," in *2009 4th International Conference on Autonomous Robots and Agents*, pp. 56–61, February 2009.

[22] I. Hughes, G. Millan, C. Cubillos, and G. Lefranc, "Colony of robots for exploration based on multi-agent system," *International Journal of Computers Communications & Control*, vol. 9, p. 703, December 2014.

[23] M. Al khawaldah and A. Nuchter, "Multi-Robot Cooperation for Efficient Exploration," *Automatika*, vol. 55, p. 276, July 2014.

[24] Y. Gu, A. Lo, and I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," *IEEE Communications Surveys Tutorials*, vol. 11, pp. 13–32, January 2009.

[25] G. Taraldsen, T. A. Reinen, and T. Berg, "The underwater GPS problem," in *OCEANS 2011 IEEE - Spain*, pp. 1–8, June 2011.

[26] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[27] S. Saeedi, L. Paull, M. Trentini, and H. Li, "Multiple-Robot Simultaneous Localization and Mapping: A Review," vol. 33, pp. 853–858, September 2011.

[28] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics Automation Magazine*, vol. 13, pp. 99–110, June 2006.

[29] C. Mei, E. Sommerlade, G. Sibley, P. Newman, and I. Reid, "Hidden view synthesis using real-time visual SLAM for simplifying video surveillance analysis," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4240–4245, May 2011.

[30] F. R. Heukels, "Simultaneous Localization and Mapping (SLAM) : towards an autonomous search and rescue aiding drone," 2015.

[31] M. Couceiro, A. Lopes, N. Ferreira, A. Ferreira, and R. Rocha, *Toward the Concept of Robot Society: A Multi-Robot SLAM Case Study*, pp. 57–66. January 2014.

[32] P. Moutarlier and R. Chatila, "An experimental system for incremental environment modelling by an autonomous mobile robot," *Experimental Robotics I. Lecture Notes in Control and Information Sciences*, pp. 327–346, April 2006.

[33] R. Smith and P. Cheeseman, "On the Representation and Estimation of Spatial Uncertainty," *The International Journal of Robotics Research*, vol. 5, February 1987.

[34] J. Andrade-Cetto, T. Vidal-Calleja, and A. Sanfeliu, "Unscented Transformation of Vehicle States in SLAM," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 323–328, April 2005.

[35] E. Nettleton, P. Gibbens, and H. Durrant-Whyte, "Closed form solutions to the multiple-platform simultaneous localization and map building (SLAM) problem," *Sensor Fusion*, vol. 4051, April 2000.

[36] E. Nettleton, H. F. Durrant-Whyte, P. W. Gibbens, and A. Göktogan, "Multiple-platform localization and map building," in *SPIE Optics East*, 2000.

[37] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *I. J. Robotic Res.*, vol. 23, pp. 693–716, July 2004.

[38] A. Howard, "Multi-robot Simultaneous Localization and Mapping using Particle Filters," in *Robotics: Science and Systems*, 2005.

[39] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 1250–1257 vol.2, May 1994.

[40] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 125–137, April 2001.

[41] D. Asmar, "2D Occupancy-Grid SLAM of Structured Indoor -Environments using a Single Camera," *International Journal of Mechatronics and Automation*, vol. 2, pp. 112–124, January 2012.

[42] P. de la Puente and D. Rodriguez-Losada, "Feature based graph-SLAM in structured environments," *Autonomous Robots*, vol. 37, pp. 243–260, October 2014.

[43] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, "Semantic SLAM Based on Object Detection and Improved Octomap," *IEEE Access*, vol. 6, pp. 75545–75559, 2018.

[44] B. Bacca, J. Salvi, and X. Cufi, "Appearance-based SLAM for mobile robots," vol. 202, pp. 55–64, January 2009.

[45] G. Chartrand, *Introductory Graph Theory*. Dover Publications, Inc., 1985.

[46] R. Mautz and S. Tilch, "Survey of optical indoor positioning systems," in *2011 International Conference on Indoor Positioning and Indoor Navigation*, pp. 1–7, September 2011.

[47] N. Kirchner and T. Furukawa, "Infrared Localisation for Indoor UAVs," January 2005.

[48] M. Basiri, "Audio-based Positioning and Target Localization for Swarms of Micro Aerial Vehicles," January 2015.

[49] J. Wu, "Three-Dimensional Indoor RFID Localization System," December 2012.

[50] F. Subhan, H. Hasbullah, A. Rozyyev, and S. Tahir Bakhsh, "Indoor Positioning in Bluetooth Networks Using Fingerprinting and Lateration Approach," *Proc. ICISA*, April 2011.

[51] J. Tiemann and C. Wietfeld, "Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–7, September 2017.

[52] M. Vossiek, L. Wiebking, P. Gulden, J. Wieghardt, C. Hoffmann, and P. Heide, "Wireless local positioning," *IEEE Microwave Magazine*, vol. 4, pp. 77–86, December 2003.

[53] K. Li, C. Wang, S. Huang, G. Liang, X. Wu, and Y. Liao, "Self-positioning for UAV indoor navigation based on 3D laser scanner, UWB and INS," in *2016 IEEE International Conference on Information and Automation (ICIA)*, pp. 498–503, August 2016.

[54] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar, "System for deployment of groups of unmanned micro aerial vehicles in GPS-denied environments using onboard visual relative localization," *Autonomous Robots*, vol. 41, April 2016.

[55] P. Conroy, D. Bareiss, M. Beall, and J. P. van den Berg, "3-D Reciprocal Collision Avoidance on Physical Quadrotor Helicopters with On-Board Sensing for Relative Positioning," *ArXiv*, November 2014.

[56] J. Roberts, T. Stirling, J.-C. Zufferey, and D. Floreano, "3-D Relative Positioning Sensor for Indoor Collective Flying Robots," *Autonomous Robots*, vol. 33, August 2012.

[57] M. Coppola, K. Mcguire, K. Scheper, and G. Croon, "On-board Bluetooth-based Relative Localization for Collision Avoidance in Micro Air Vehicle Swarms," September 2016.

[58] K. Guo, X. Li, and L. Xie, "Ultra-wideband and Odometry-Based Cooperative Relative Localization With Application to Multi-UAV Formation Control," *IEEE Transactions on Cybernetics*, vol. PP, pp. 1–14, April 2019.

[59] S. Helm, M. Coppola, K. Mcguire, and G. Croon, "On-board range-based relative localization for micro air vehicles in indoor leader-follower flight," *Autonomous Robots*, March 2019.

[60] D. van Wijk, "Network-Decentralized Control with Collision Avoidance for Multi-Agent Systems." MSc Thesis, Delft University of Technology, 2018.

[61] P. Ledzian, "Network Decentralized Collision Avoidance with Applications in a Scalable Unmanned Aerial System Testbed." MSc Thesis, Delft University of Technology, 2019.

[62] E. Cetinsoy, "Design and simulation of a holonomic quadrotor UAV with sub-rotor control surfaces," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1164–1169, December 2012.

[63] E. Cetinsoy, "Design and initial flight tests of a holonomic quadrotor UAV with sub-rotor control surfaces," *20th Annual International Conference on Mechatronics and Machine Vision in Practice, M2VIP 2013*, pp. 63–69, January 2013.

[64] S. K. Phang, C. Cai, B. M. Chen, and T. H. Lee, "Design and mathematical modeling of a 4-standard-propeller (4SP) quadrotor," in *Proceedings of the 10th World Congress on Intelligent Control and Automation*, pp. 3270–3275, July 2012.

[65] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pp. 498–503, November 2016.

[66] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 153–158, October 2007.

[67] H. talla Mohamed Nabil Elkholy, "Dynamic modeling and control of a quadrotor using linear and nonlinear approaches," 2014.

[68] A. Joukhadar, M. Alchehabi, and A. Jejeh, *Advanced UAVs Nonlinear Control Systems and Applications*. June 2019.

[69] A. Nagaty, S. Saeedi, C. Thibault, M. Seto, and H. Li, "Control and Navigation Framework for Quadrotor Helicopters," *Journal of Intelligent & Robotic Systems*, vol. 70, April 2013.

Glossary

List of Acronyms

MAS	Multi-Agent Systems
GPS	Global Positioning System
RSSI	Received Signal Strength Indication
IR	Infrared
RF	Radio Frequency
AoA	Angle of Arrival
ToA	Time of Arrival
TDoA	Time Difference of Arrival
RFID	Radio Frequency Identification
LoS	Line of Sight
UHF	Ultra High Frequencies
UWB	Ultra-wideband
INS	Inertial Navigation System
SLAM	Simultaneous Localization and Mapping
EKF	Extended Kalman Filter
KF	Kalman Filter
UKF	Unscented Kalman Filter
EIF	Extended Information Filter
SEIF	Sparse Extended Information Filter

RBPFs	Rao-Blackwellized Particle Filters
MAP	Maximum a Posteriori
CPS	Cooperative Positioning System
CoG	Centre of Gravity
DoF	Degrees of Freedom
BFF	Body Fixed Frame
EFF	Earth-Fixed Frame
NED	North-East-Down
IMU	Inertial Measurement Unit