Robotics and Autonomous Systems 132 (2020) 103604

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Collective navigation of a multi-robot system in an unknown environment

Ertug Olcay *, Fabian Schuhmann, Boris Lohmann

Technical University of Munich, Department of Mechanical Engineering, Chair of Automatic Control, Boltzmannstr. 15, 85748, Garching bei München, Germany

ARTICLE INFO

Article history:
Received 3 February 2020
Received in revised form 13 July 2020
Accepted 14 July 2020
Available online 22 July 2020

Keywords:
Swarm intelligence
Cooperative control
Autonomous systems
Collective navigation

ABSTRACT

The navigation of autonomous, mobile multi-robot systems in changing environments is a challenging problem investigated over the past years. Cooperative, multiple robots are employed for many different tasks to increase the efficiency and success of a mission. However, many of the existing collective path planning approaches do not guarantee a reliable escape in environments with complex, non-convex obstacles without any prior knowledge. In this study, we developed a navigation framework for multi-robot systems in unknown areas that solely exploit the sensing information and shared data among the agents. The key contribution of this paper is the simultaneous, collision-free motion planning for fully autonomous robots in a collective manner. Furthermore, our communication architecture enables the robots to find an appropriate path to a desired, joint target position, despite the limited sensing and communication range.

1. Introduction

Control of multi-agent systems with the focus on swarm robotics has been under increasing levels of investigation for many different purposes. Autonomous multi-robot systems have a wide range of applications, such as map creation of unknown areas, space mining, discovery and exploration of certain regions on a planet. In addition, reconnaissance and rescue with swarms of autonomous mobile robots are other possible tasks for multi-robot systems [1]. Mobile robot swarms equipped with sensors not only allow large amounts of data to be collected in known and unknown environments [2], but also eliminate risks to humans in dangerous operation areas.

One of the biggest issues in exploration missions is the difficult areas with complex obstacles such as concave and cornered walls. Collision-free path planning of multi-robot systems is one of the keys to a successful mission. Perception of the environment, object recognition, position and velocity control are some of the constituent parts of robot guidance tasks. The focus of this paper lies in collective navigation with limited sensing and communication capability.

The research field of multi-robot systems and swarm robotics often draws inspiration from flocking behavior in nature. Flocking usually denotes the motion of a population with a common velocity. Besides, *swarming* is mainly referred to the phenomenon of cohesive motion, which does not necessarily require a common velocity [3]. In this paper, we study control of swarming agents by partly benefiting the robustness of flocking behavior. Three simple rules were defined to imitate flock-like behavior in a multi-agent system: *cohesion*, *separation*, and *alignment* [4]. The research field contains numerous studies, such as formation control [5,6], exploration and area coverage with mobile sensor networks [7–9]. For an overview of decentralized motion control of multi-agent systems, the reader is referred to [10].

Autonomous vehicles and robots utilize different kinds of navigation strategies. Motion planning approaches for autonomous robots can be divided into two categories. The first one is the global navigation approach, which assumes that the robot receives a map of the environment before path planning. This map can include all static obstacles and boundaries of the area. In this case, an optimal path is computed based on the prior knowledge of the obstacles. Optimal control-based schemes [11–13] and graph traversal algorithms such as A* algorithm [14] have been investigated for motion planning in this category. The weakness of these approaches is that the robot cannot handle environmental changes during an operation.

The second strategy is the local navigation approach (local motion planning), which does not require prior information about the environment. The robot can plan its actions autonomously based on sensor data and react to unexpected events. The proposed approach in this study falls into the category of local motion planning. Over the past two decades, several methods have been proposed for local navigation of multi-agent systems. The

* Corresponding author.
E-mail addresses: ertug.olcay@tum.de (E. Olcay), fabian.schuhmann@tum.de (F. Schuhmann), lohmann@tum.de (B. Lohmann).

https://doi.org/10.1016/j.robot.2020.103604
0921-8890/

ability of robots to localize themselves in a map and to plan elaborated motions are the basics of many local navigation approaches. There are several Simultaneous Localization And Mapping (SLAM) algorithms that allow robots to build a map of the environment and, at the same time, use this map to compute its location. Since some SLAM algorithms are based on probabilistic methods, combination of SLAM with other navigation mechanisms may increase efficiency [15]. One widespread approach is the potential field-based method, which generates artificial attraction, repulsion, and alignment forces [16–20]. Further similar approaches also exist, such as the Vector Field Histogram (VFH) [21–23], which exhibits problems in handling local minima (e.g., concave obstacles). In order to overcome this issue, escape-based methods have appeared in recent years. Performing a spiral motion around a specific point [24,25], generating elliptic trajectories [26] and adding a reactive rotational force field [27] are some of these strategies. Cooperative, decentralized navigation of multi-robots through information exchange among group members and data evaluation is another approach for planning optimal collective motions when simultaneously guiding multiple agents [28,29]. In order to achieve a common goal, multiple robots require cooperative control strategies. Based on the characteristics defined in [30, p. 1–2], a cooperative MAS should have the following features:

- Trajectories of all the agents evolve collaboratively toward accomplishing a common objective. The common objective can be interpreted as the equilibrium point of the system.
- Agents can usually exchange some information via a communication network.
- Agents can generally interact with a dynamically changing physical environment, which might have an influence on the system's dynamics and communication network.
- A state change of each agent may be subject to both constraints in kinematics and dynamics.

An important challenge in escape-based approaches are concave and labyrinth-like obstacles, which can appear in indoor applications.

Another method is Model Predictive Control (MPC) [31], which is applied to minimize a cost function repeatedly at each time step using a nonlinear dynamic model of the system. It is well suited for changing environments in path planning. However, this method is generally computationally intensive, especially in multi-agent settings.

In a collective navigation task, a possible collision or loss of some agents is associated with costs and failure of the mission. Therefore, it is important for unfamiliar or changing environments to be dealt with spontaneously and with a low computational cost. Many of the current multi-robot navigation algorithms dealing with rendezvous problem and based on the local path planning do not enable robots to escape from non-convex obstacles by preserving proximity.

The main contributions of this paper are summarized as follows.

(i) First, we formulate a single robot navigation strategy similar to that in study [32], which employs only sensors. This method allow a robot to navigate in an unknown static environment by using only range sensors. We extend it into a decentralized multi-robot navigation framework, where each robot only needs to communicate with its neighbors for motion planning in an independent way.

(ii) Second, by means of local communication and information exchange, we enable the robots to make early decisions and perform simultaneous, optimal collective maneuvers during a task in a distributed manner. Furthermore, using neighbor-to-neighbor communication increases flexibility and scalability with respect to the system size (agent number). In addition, we use potential fields to guarantee inter-agent collision avoidance, preserve proximity and increase safety in collision avoidance with static obstacles. Moreover, in this study we consider the limited sensor and communication range of the robots to be a challenging factor.

The paper is organized as follows: Theoretical background and dynamics of the multi-robot system under consideration are briefly introduced in Section 2. The local minimum problem in navigation is described in this section. Section 3 presents a navigation scheme for a single robot. In Section 4, we extend the scheme from the previous section and propose a new approach for the navigation of a multi-robot system. Subsequently, in Section 5, the collective navigation framework is explained in detail. Simulation results are presented in Section 6. Finally, concluding remarks are given in Section 7.

2. Theoretical background

In this section, the dynamics of the robots[1] are formulated. For the path planning, the robots are considered as point masses and the dynamics are based on potential functions. The robots receive virtual attractive and repulsive forces by following the gradient of a potential field. For the path planning, we consider only the position and velocity of each robot i in 2 dimensional space, $(\mathbf{p}_i, \mathbf{v}_i) \in \mathbb{R}^2 \times \mathbb{R}^2$. The dynamics of each mobile robot i are described using the following system of differential equations:

$$\dot{\mathbf{p}}_i = \mathbf{v}_i, \quad \text{Position des Roboters ändert sich mit seiner Geschwindigkeit} \tag{1}$$

$$\dot{\mathbf{v}}_i = \mathbf{u}_i, \quad \text{Geschwindigkeit des Roboters verändert sich mit der Steuerkraft ui, die auf ihn wirkt} \tag{2}$$

where $\mathbf{u}_i \in \mathbb{R}^2$ is the control input. In time-discrete implementation, however, the dynamics of the system are described by $\mathbf{p}(t_k+1) = \mathbf{p}_i(t_k) + \mathbf{v}_i \Delta t$ and $\mathbf{v}(t_k+1) = \mathbf{v}_i(t_k) + \mathbf{u}_i \Delta t$, where Δt is a finite step size and t_k is the kth time step. Since the robots have limited communication radius r_c, the neighborhood of a robot is defined as

$$\mathcal{N}_i^{\alpha} = \{j \in \mathcal{V} : \|\mathbf{p}_j - \mathbf{p}_i\| < r_c\}, \quad \text{Ein Roboter kann nur mit anderen Robotern sprechen, die in seiner Nähe (innerhalb Radius rc) sind} \tag{3}$$

where $\mathcal{V} = \{1, \ldots, N\}, N \in \mathbb{N}$, is a set of all robots. $\mathcal{N}_i^{\alpha}$ is an undirected, time-varying network of neighboring robots. The communication among robots in this set is bidirectional.

In order to establish flock-like behavior, a deviation energy is generated as a function depending on the distance between two robots. For the differentiability of this function at $z = 0$, mapping, the so called σ-norm, is introduced as follows [33]:

$$\|z\|_{\sigma} = \frac{1}{\epsilon}[\sqrt{1 + \epsilon\|z\|^2} - 1], \quad \text{modifizierte Entfernungsmessung, damit Steuerung bei sehr kleinen Abständen zw. Robotern stabil bleibt} \tag{4}$$

with a fixed parameter $\epsilon > 0$. For a smooth transition between 0 and 1 in the definition of *spatially-weighted* neighborhood matrix and interaction forces, we use the following function:

$$\rho_h(z) = \begin{cases} 1, & z \in [0, h) \\ \frac{1}{2}[1 + \cos(\pi \frac{(z-h)}{(1-h)})], & z \in [h, 1) \\ 0, & \text{otherwise} \end{cases} \quad \text{bestimmt wie stark Roboter untereinander je nach Entfernung interagieren. Nah dran, hohe Wirkung} \tag{5}$$

where $h \in (0, 1)$. With the help of (5), we can define the adjacency matrix $\mathbf{A} = (a_{ij})$ of the network $\mathcal{N}_i^{\alpha}$ as follows:

$$a_{ij}(\mathbf{p}) = \rho_h\left(\frac{\|\mathbf{p}_j - \mathbf{p}_i\|_{\sigma}}{r_{\alpha}}\right) \in [0, 1], \qquad j \neq i \quad \text{Wert zw 0 und 1, der angibt, wie stark zwei Roboter, je nach Abstand, miteinander verbunden sind} \tag{6}$$

with $r_{\alpha} = \|r_c\|_{\sigma}$. In this study, we aim to utilize the benefits of flocking behavior (cohesion, separation and alignment properties). Due to the ease of analysis and robustness in terms of

[1] In this paper, the holonomic "robots" are sometimes called "agents".

collective behavior, the dynamics of the multi-robot system are described similarly to those in the flocking algorithm in [33]. The control input is described by the following dynamics:

$$\mathbf{u}_i = \mathbf{u}_i^{\alpha} + \mathbf{u}_i^{\beta} + \mathbf{u}_i^{\gamma}, \quad \text{drei zusammen wirkende Steuerungen, eine für die Gruppe, eine für Hindernisse, eine für das Ziel} \tag{7}$$

where $\mathbf{u}_i^{\alpha}$ is the control input for *flock-like* behavior, $\mathbf{u}_i^{\beta}$ is for the *obstacle avoidance* and $\mathbf{u}_i^{\gamma}$ is for the *navigation*.

The control input $\mathbf{u}_i^{\alpha}$ consists of gradient-based and consensus terms as below: Einfluss der Nachbarn auf Roboter i

$$\mathbf{u}_i^{\alpha} = c_1^{\alpha} \underbrace{\sum_{j \in \mathcal{N}_i^{\alpha}} \phi_{\alpha}(\|\mathbf{p}_j - \mathbf{p}_i\|_{\sigma})\mathbf{n}_{ij}}_{\text{Gradient-based term}} + c_2^{\alpha} \underbrace{\sum_{j \in \mathcal{N}_i^{\alpha}} a_{ij}(\mathbf{p})(\mathbf{v}_j - \mathbf{v}_i)}_{\text{Consensus term}}, \tag{8}$$

where $\mathbf{n}_{ij} = \sigma_{\epsilon}(\mathbf{p}_j - \mathbf{p}_i) = \dfrac{\mathbf{p}_j - \mathbf{p}_i}{\sqrt{1 + \epsilon\|\mathbf{p}_j - \mathbf{p}_i\|^2}}$ a vector from $\mathbf{p}_j$ to $\mathbf{p}_i$ and $\epsilon \in (0, 1)$ a constant of the σ-norm. Using the control input $\mathbf{u}_i^{\alpha}$, agents try to keep a predefined distance d to each other. The necessary potential functions for the gradient-based term are formulated as

$$\phi_{\alpha}(z) = \rho_{h_{\alpha}}(z/r_{\alpha})\phi(z - d_{\alpha}) \tag{9}$$

with $d_{\alpha} = \|d\|_{\sigma}$ and

$$\phi(z) = \frac{1}{2}[(a+b)\sigma_1(z+c) + (a-b)], \quad \text{bestimmt wie stark zwei Roboter sich anziehen oder abstoßen sollen, je nach Abstand z} \tag{10}$$

where $\sigma_1 = \frac{z}{\sqrt{1+z^2}}$ and $\phi(z)$ is an uneven sigmoid function with parameters: $0 < a \leq b$, $c = \frac{|a-b|}{\sqrt{4ab}}$, which guarantee $\phi(0) = 0$.

Each agent can measure the relative position between the closest point on an obstacle. In this way, each agent can identify a set of detected obstacle points at a given time t_k,

$$\mathcal{N}_i^{\beta} = \{\hat{\mathbf{p}}_{i,k} \mid \|\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i\| < r_s\}, \quad \text{Menge von Hindernispunkten von Roboter i innerhalb seines Sensor-Radis rs} \tag{11}$$

where r_s is the sensor range for obstacle detection and $\hat{\mathbf{p}}_{i,k}$ is the detected point within the sensor, which has the shortest distance to the agent's current position. The control input for keeping a safe distance d_s from an obstacle is defined as

$$\mathbf{u}_i^{\beta} = c_1^{\beta} \sum_{k \in \mathcal{N}_i^{\beta}} \phi_{\beta}(\|\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i\|_{\sigma})\hat{\mathbf{n}}_{i,k} + c_2^{\beta} \sum_{k \in \mathcal{N}_i^{\beta}} b_{i,k}(\mathbf{p})(\hat{\mathbf{v}}_{i,k} - \mathbf{v}_i), \tag{12}$$

Berechnet die Kraft, mit der der Roboter einem Hindernis ausweicht, abhängig von der Nähe und der Bewegung des Hindernispunktes

with

$$\hat{\mathbf{n}}_{i,k} = \frac{\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i}{\sqrt{1 + \epsilon\|\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i\|^2}}, \quad b_{i,k}(\mathbf{p}) = \rho_{h_{\beta}}\left(\frac{\|\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i\|_{\sigma}}{d_{\beta}}\right),$$

Richtung von Roboter i zum gemessenen Hindernispunkt pik, als Einheitsvektor. In welche Richtung muss Roboter gucken, damit er weiß, wo das Hindernis ist?

where $\hat{\mathbf{v}}_{i,k}$ is the projection of the $\mathbf{v}_i$ on to the edge of the obstacle k. Moreover, $b_{i,k}$ is the weighted *adjacency* between agent i and obstacle k. In addition, a new repulsive potential force related to the obstacles is formulated as follows:

$$\phi_{\beta}(z) = \rho_{h_{\beta}}(z/d_{\beta})(\sigma_1(z - d_{\beta}) - 1), \quad \text{wie stark ein Roboter von Hindernis abgestoßen wird, je nach Abstand} \tag{13}$$

where $d_{\beta} < r_{\beta}$ with $d_{\beta} = \|d_s\|_{\sigma}$, $r_{\beta} = \|r_s\|_{\sigma}$. The control term for the *navigation* is defined as follows

$$\mathbf{u}_i^{\gamma} = -c_1^{\gamma}\sigma_1(\mathbf{p}_i - \mathbf{p}_d) - c_2^{\gamma}\mathbf{v}_i, \quad \text{virtuelle Kraft, die Roboter zum Ziel zieht} \tag{14}$$

where $\mathbf{p}_d \in \mathbb{R}^2$ represents the *desired* goal position. Here, c_{η}^{ν} are positive constant parameters for all $\eta = 1, 2$ and $\nu = \alpha, \beta, \gamma$. The priorities of the control objectives *flock-like behavior*, *obstacle avoidance* and *navigation* are weighted by these parameters.

The problem statement

In the following, we describe the problems with artificial potential field-based schemes in local path planning. There are many navigation approaches for multi-robot systems that are based on potential functions. Through artificial forces, robots are attracted from a target position and repulsed from obstacles and from other robots to avoid collisions. One common problem in potential-field method is the local minima, where repulsive and attractive forces are balanced. In addition to potential functions, camera-based approaches are also used for object detection and obstacle avoidance. However, high-intense sunlight or smoky environments can reduce the performance.

Various strategies have been investigated to overcome these problems especially for single robot navigation. The sensor-based approaches for collective motion planning include elimination of a local minimum by reconstructing potential functions through additional forces [27] and escape from local minima so that the balance between artificial forces are broken [29]. An arising issue in elimination-based methods is that the success of the escape heavily depends on particular obstacle configurations, especially its concaveness. Moreover, escape-based approaches do not always guarantee success in intricate passageways. However, they have a potential in accomplishing complex navigation tasks in swarms through communication, evaluation of environmental information and collective intelligence.

In many multi-agent control problems, multiple agents collaboratively work on a common task. One of the major issues in many multi-robot coordination tasks are the difficult operation areas with complex obstacles. In this study, we are concerned with the collective navigation of multiple robots in complex environments that are not known to robots. Each robot has a range sensor with a limited measuring range r_s and without having prior information about the workspace, it is challenging to navigate the group in an optimal way. The reason is that the obstacles or restrictions in the operation area cannot be perceived with relative short sensing ranges in their entirety. This requires an elaborated communication and through this efficient motion planning. Furthermore, each robot can only receive the relative position of its neighbors and exchange information only with other robots within its limited communication bandwidth r_c.

We assume that the measurements are without noise. For dealing with noisy measurements, we refer to distributed Kalman Filter for an improved collective state prediction [34]. In addition, the communication between agents occurs without package loss or time delay.

3. Single robot navigation

In this section, we briefly introduce a navigation schema for a single agent ($N = 1$), which we extend with a communication interface, collective motion planning framework and potential forces in later sections. The proposed approach mainly takes inspiration from the tangential navigation schema presented in [32]. In obstacle-free areas, agents use the control term (14) to move toward the desired goal position. However, the proposed approach in [32] is based on the reconstruction of the *navigation* term (14) by allocating temporary virtual goal positions considering the obstacle geometry to avoid local minima.

3.1. Tangential navigation

Once an obstacle is in the range of r_{tan} ($r_s > r_{tan} > d_s$), $\|\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i\| \leq r_{tan}$, the robot starts to perform the tangential navigation and it travels parallel to the obstacle surface. Firstly, the robot

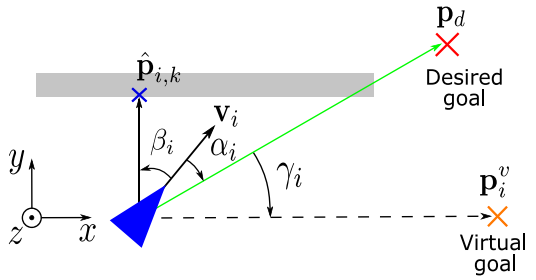
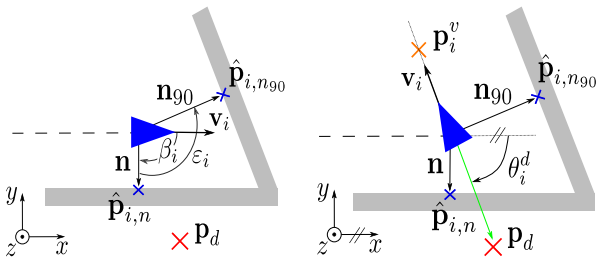


Fig. 1. Illustration of the tangential navigation schema.
Roboter erkennt Hindernis, fährt seitlich (tangential) daran entlang und legt neues Zwischenziel fest



(a) Corner detection. (b) Corner avoidance maneuver.

Fig. 2. Corner avoidance schema.
Roboter erkennt Ecke, weicht um bestimmten Winkel ab und legt neues Zwischenziel fest

determines the angle γ for the rotation matrix to project the desired goal position as follows

$$\gamma_i = \begin{cases} \beta_i - \alpha_i - 90°, & \text{if } \beta_i \geq 0° \\ \beta_i - \alpha_i + 90°, & \text{if } \beta_i < 0° \end{cases} \quad (15)$$

je nachdem ob Hindernis links oder rechts vom Roboter ist, wird entschieden, in welche Richtung der Roboter seitlich ausweicht

where the angles α_i and β_i are the orientation of the robot relative to the desired goal position and to the closest obstacle point, respectively, as shown in Fig. 1. The angle β_i is in the range $]-180, 180]$ and defines the relative position of an obstacle to the robot. If $\beta_i < 0°$, the obstacle is to the right of the robot. If $\beta_i > 0°$, the obstacle is on the left.

According to Eq. (15), the rotation direction of the agent depends on its orientation to the obstacle. For a better cohesive behavior in complex environments, we can predefine a primary direction of rotation using the following parameter:

$$c_r = \begin{cases} -1, & \text{for clockwise rotation} \\ 1, & \text{for counter-clockwise rotation} \end{cases} \quad (16)$$

In this case, the rotation angle should be calculated as follows:

$$\gamma_i = \beta_i - \alpha_i + c_r \cdot 90°. \quad (17)$$

The angle γ_i is used as a rotation angle for the calculation of the virtual goal position $\mathbf{p}_i^v$ defined as

$$\mathbf{p}_i^v = \mathbf{p}_i + \begin{bmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix} (\mathbf{p}_d - \mathbf{p}_i). \quad (18)$$

Wenn Hindernis, rechnet Roboter neues Zwischenziel aus (gedreht um einen Winkel yi)

The control input for the navigation is calculated similar to Eq. (14) as follows

$$\mathbf{u}_i^\gamma = -c_1^\gamma \left(c_n \cdot \frac{\mathbf{p}_i - \mathbf{p}_i^v}{\|\mathbf{p}_i - \mathbf{p}_i^v\|} \right) - c_2^\gamma \mathbf{v}_i, \quad (19)$$

where c_n is a positive constant for specifying a constant acceleration toward the virtual goal position.

3.2. Corner avoidance

The characteristic of a concave corner is that the robot simultaneously senses two different points on an obstacle. Each sampling time (each 0.02 s, for our simulation setup), the motion planner checks if two different points are sensed. If two obstacles are detected at the same time, the robot applies the corner avoidance maneuver.

Firstly, the robot classifies the detected obstacle points for the motion planning. Here, the normal vector to the obstacle, which lies in the direction of motion $\mathbf{v}_i$ in front of the agent, is defined as $\mathbf{n}_{90}$, the other one as $\mathbf{n}$. The corresponding sensed obstacle points are declared as $\hat{\mathbf{p}}_{i,n_{90}}$ and $\hat{\mathbf{p}}_{i,n}$. In order to calculate the rotation angle γ_i, we introduce the angle ε_i, which is defined between $\mathbf{n}$ to $\mathbf{n}_{90}$, measured from $\mathbf{n}$ (Fig. 2(a)). The rotation angle for projection of a new temporary, virtual goal position is defined as follows:

$$\gamma_i = \begin{cases} \beta_i - \alpha_i - \tau_i, & \text{if } \varepsilon_i \geq 0° \\ \beta_i - \alpha_i + \tau_i, & \text{if } \varepsilon_i < 0° \end{cases} \quad (20)$$

with

$$\tau_i = |\varepsilon_i| + 90°. \quad (21)$$

The angle ε_i defines the necessary direction of rotation in an indirect way, e.g., $\varepsilon_i \geq 0°$ results in a counter-clockwise rotation. Finally, the virtual goal position is calculated as

$$\mathbf{p}_i^v = \mathbf{p}_i + \begin{bmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix} \begin{bmatrix} 0.5 \cdot d_s \cdot \cos(\theta_i^d) \\ 0.5 \cdot d_s \cdot \sin(\theta_i^d) \end{bmatrix}, \quad (22)$$

where θ_i^d describes the desired orientation of the robot to the current, temporary goal position (Fig. 2(b)). The angle θ_i^d is defined as the angle between the x-axis of the inertial coordinate system and the line linking the agent's position and the desired goal position.[2] Note that the robot sets the virtual goal point to a closer area with the aim of generating a *weaker* attraction force compared to the tangential navigation. In this way, it performs more precise and safer maneuvers in the corners.

3.3. Motion planning at obstacle extremities

While a robot follows an obstacle tangentially, it loses the sensing contact shortly after reaching the obstacle endpoint. Without proper motion planning, it would move back to the desired goal position and might get into a local minimum instead of performing a suitable maneuver to turn around the obstacle extremity.

For the endpoint detection, we introduce a binary flag P_{tan} as in [32]. During the tangential navigation, the robot continuously senses the obstacle, $P_{tan} = 1$. The flag is only reset to 0 once there is no longer an obstacle inside the radius r_{tan}. If the flag P_{tan} is reset from 1 to 0, the robot recognizes the case *endpoint* and utilizes the detected obstacle point from the last time instant ($\hat{\mathbf{p}}_{i,k}(t_{k-1}) = \hat{\mathbf{p}}_{i,e}$), illustrated by the blue colored cross in Fig. 3, as a center point for its rotation. Furthermore, the calculated angle ($\beta_i(t_{k-1}) = \beta_{i,e}$) is also stored. Along a circular path, the robot iteratively calculates virtual goal positions rotating around $\hat{\mathbf{p}}_{i,e}$ by a defined angle of rotation δ. The angle of rotation for the maneuver is formulated as:

$$\gamma_i = \begin{cases} +\delta, & \text{if } \beta_{i,e} \geq 0° \\ -\delta, & \text{if } \beta_{i,e} < 0° \end{cases} \quad (23)$$

where $\delta > 0°$ represents the value of the predefined rotation angle. In the next step, the virtual goal position is determined on a circular path by the following equation:

$$\mathbf{p}_i^v = \hat{\mathbf{p}}_{i,e} + \begin{bmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix} \mathbf{n}_{i,e}, \quad (24)$$

Roboter legt neues virtuelles Ziel auf einer Kreisbahn um den Hindernispunkt pie fest, indem er seinen aktuellen Punkt um den Winkel yi dreht

with

$$\mathbf{n}_{i,c} = \frac{\mathbf{p}_i(t_k) - \hat{\mathbf{p}}_{i,e}}{\|\mathbf{p}_i(t_k) - \hat{\mathbf{p}}_{i,e}\|} \cdot r_{tan},$$

[2] The angle θ_i^d is measured from the positive x-axis.

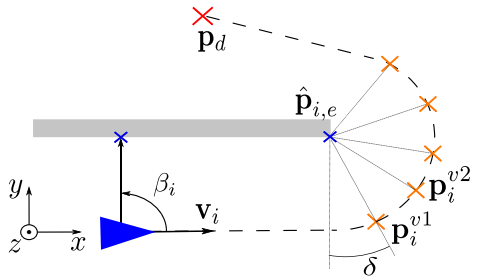


Fig. 3. Motion planning at the obstacle endpoint. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Roboter fährt Kurve, wenn Ende eines Hindernisses erreicht wurde

where the radius of the circular path is equal to r_{tan}. Once the agent reaches a virtual goal position on the circular path, e.g., $\|\mathbf{p}_i^{v1} - \mathbf{p}_i\| < 1$, it determines a new one, e.g., $\mathbf{p}_i^{v2}$ (see Fig. 3). In this way, the agent keeps setting new goal positions until the following condition is fulfilled:

$$|\alpha_i| \leq \delta. \tag{25}$$

This allows the robot an optimal orientation with respect to the desired goal position $\mathbf{p}_d$ to leave the circular path and move toward the desired goal position.

4. The proposed navigation approach for a multi-agent system

The proposed approach is based on the navigation algorithm for a single robot described in Section 3 and extends it by means of a novel communication interface for collective motion planning. With the help of inter-agent communication, agents are capable of perceiving and acting upon the environment by using the information from the communication network. In this section, we introduce the communication interface.

The basic principle of the navigation scheme presented in Section 3 is the tangential projection of a virtual goal position. In the case of multiple agents, the continuous creation of an individual, virtual goal position may yield a decomposition of the group or collisions. In order to prevent this, the information about the projection of a new, virtual goal should be communicated within the neighborhood of each agent. In addition, the critical points identified on an obstacle, such as corners and endpoints, should also be shared for a collective navigation strategy.

Fig. 4 shows the critical points for the communication scenarios. The first scenario depicts the tangential navigation along an obstacle. The second scenario is the detection of a corner and the third one represents the detection of an obstacle extremity. These scenarios are communicated independently from one other. In this way, the agents also acquire knowledge about the area and utilize the information for localization. The desired principles for the communication process in our approach are as follows:

- An agent should consider only the most relevant information for its next action.
- Only the contemporary information in the communication network should be considered.
- Detection before communication: A detected obstacle point has a higher priority than information received through communication in motion planning. Navigation based on the information from the communication network may result in a collision if an obstacle is detected. Hence, in such cases, the detected obstacle point is primarily taken into consideration for the next action of the agent. In this way, the agent can certainly avoid a possible collision with the obstacle.

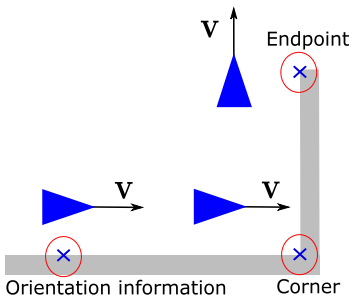


Fig. 4. The critical points for the communication scenarios.

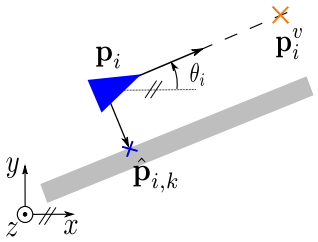


Fig. 5. Orientation information.

4.1. Structure of the communication interface

The communication network contains the information packages aggregated by means of own sensors and the information received from other agents via communication. Note that the information items received from the communication network will be identified later by the upper index c. In our concept, we define three types of information packages: *orientation information*, *information about endpoints* and *information about corners*. Each type of information package consists of several different items.

Orientation information

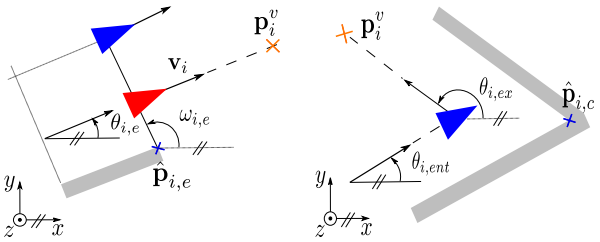
The information package consists of several items. The core item is the goal orientation θ_i and the detected point on the obstacle $\hat{\mathbf{p}}_{i,k}$. The angle goal orientation describes the angle between the x-axis of the inertial coordinate system and the line linking the agent's position with the current virtual goal position (see Fig. 5).[3] A further important item is $status_i$, which defines the current action of the agent and is included in the information package. We implemented several statuses that are explained later (see Section 5), as shown in Table 1. Through the assigned statuses, an agent receiving an information package can identify the actions of its neighboring agents. The last item is $\hat{t}_i$, which represents the detection time of an obstacle.

Information about endpoints

At the endpoint of an obstacle, the agent starts to move on a circular path in the direction of the desired goal position. This motion is planned by means of the last detected point $\hat{\mathbf{p}}_{i,e}$ and the angle $\omega_{i,e}$ between the normal vector from the agent to the obstacle and the x-axis of the inertial coordinate system.[4] Moreover, the goal orientation at the time instant of endpoint detection $\boldsymbol{\theta}_{i,e}$ is another important item in planning the motion of the agent. The agent that identifies an endpoint (shown in red, Fig. 6(a)) shares the items $\hat{\mathbf{p}}_{i,e}$, $\omega_{i,e}$, $\boldsymbol{\theta}_{i,e}$ within its neighborhood.

[3] The angle θ_i is measured from the positive x-axis.

[4] $\omega_{i,e}$ is defined from the positive x-axis.



(a) Information about an endpoint. (b) Information about a corner.

Fig. 6. Information about endpoints and corners.

Table 1
Definition of action statuses.

$status_i$	Definition
0	Motion toward the desired goal position
1	Obstacle detection and tangential navigation
2	Handling the endpoint of an obstacle
3	Corner avoidance maneuver
4	Orientation phase
5	Tangential navigation based on received information
6	Waiting mode

Information about corners

For collective motion planning, robots require information. An agent stores and communicates its goal orientation during the entry into the corner $\theta_{i,ent}$ and exit from the corner $\theta_{i,ex}$. Another important item in this information package is the corner point $\hat{\mathbf{p}}_{i,c}$ (blue colored cross in Fig. 6(b)), which is determined by calculating the point at which two lines intersect. The agents can calculate the intersection point easily using $\hat{\mathbf{p}}_{i,n}$, $\hat{\mathbf{p}}_{i,n_{90}}$, $\theta_{i,ent}$ and $\theta_{i,ex}$. Specifying these parameters allows the group of agents to make a proper collective decision for the corner avoidance maneuver.

4.2. Evaluation of information from the communication network

In contrast to a single robot navigation, robots in a multi-robot system are exposed to many information pieces at the same time through local communication. For the depth of the information processing, it is not sufficient to evaluate only the position, velocity and heading of neighboring robots. In order to prioritize the acquired information items, each agent evaluates these based on their registration time and sender-related information.

Each agent examines the relevance of the orientation information based on a weighted mean value function and assigns a relevance value *rel* defined in the interval $]-\infty, 10]$. Based on this, each agent makes a decision for its next action in the motion planning. For example, an agent considers an information item with $rel = 10$ as *very relevant*. Moreover, the agent ignores an information item with $rel \leq 0$. In the following, we will define the components of the relevance function with linear sub-functions and discrete expressions.

- **Age of the information:** In order to evaluate the temporal relevance of an information package, we define a linear relation between the current time t_k and the registration time of the information $\hat{t}^c$ by another agent in the network as follows:

$$rel_t = 10 - \frac{10 \cdot (t_k - \hat{t}^c)}{d_t}, \tag{26}$$

je älter eine empfangene Information, desto unwichtiger wird sie

where $d_t \in \mathbb{R}$ is a constant representing the duration time during which an information can have positive relevance.

- **Distance from the detected obstacle:** In order to evaluate the relevance regarding the distance of a communicated obstacle point, it is important to identify the location of the detected point with respect to the agent's motion. If it is located in front of the agent relative to its direction of motion, the distance-based relevance is defined as

$$rel_{dist} = 10 - \frac{10 \cdot \|\hat{\mathbf{p}}^c - \mathbf{p}_i\|}{d_x}, \tag{27}$$

where $d_x \in \mathbb{R}$ is the maximum distance required by $\|\hat{\mathbf{p}}^c - \mathbf{p}_i\|$ for a positive relevance. However, if the point $\hat{\mathbf{p}}^c$ is behind the agent with respect to the agents' motion perspective, the distance relevance obtains a negative value as

$$rel_{dist} = -\frac{10 \cdot \|\hat{\mathbf{p}}^c - \mathbf{p}_i\|}{d_x}. \tag{28}$$

If there is no orientation information in the agent's communication network, the distance relevance is defined as 0.

- **Relevance of the orientation based on the previous time step:** During the tangential navigation, each agent continuously expects only minor changes to its orientation. For this purpose, each agent compares its current goal orientation $\theta_i(t_k)$ with one of the existing, new orientations θ^c in its communication network. The relevance of expectation of orientation is expressed as

$$rel_{exp} = 10 - \frac{10 \cdot |\theta_i(t_k) - \theta^c|}{d_\theta}, \tag{29}$$

where d_θ is the maximum allowed difference between the angles for a positive relevance value.

- **Evaluation of the sender:** Agents also examine the sender of the information. To determine the relevance, it is important to know whether the sender is the owner of the information or just forwards received information from another agent.

$$rel_o = \begin{cases} 10, & \text{for information sent directly} \\ 0. & \text{for information sent indirectly} \end{cases} \tag{30}$$

- **Evaluation of information statuses:** Finally, $status^c$ is included in the evaluation. As shown in Table 1, an information package contains different statuses that express the sender's current action. The relevance of actions, which represent a reorientation of the agent, are evaluated with $rel_{type} = 10$. However, if an agent is following an obstacle tangentially, then $rel_{type} = 5$. Otherwise, the relevance is zero.

$$rel_{type} = \begin{cases} 10, & \text{if } status^c = 4 \vee status^c = 3 \\ 5, & \text{if } status^c = 1 \\ 0. & otherwise \end{cases} \tag{31}$$

We formulate the weighted arithmetic mean of the individual relevance values $\overline{rel}_n$ as follows

$$\overline{rel}_n = \frac{c_{type} \cdot rel_{type} + c_o \cdot rel_o + c_{exp} \cdot rel_{exp} + c_{dist} \cdot rel_{dist} + c_t \cdot rel_t}{c_{type} + c_o + c_{exp} + c_{dist} + c_t}, \tag{32}$$

Roboter rechnet sich zusammen, welche Info für ihn jetzt am wichtigsten ist

where $n \in \mathcal{S}_i$ and $\mathcal{S}_i$ is as set of all information packages in the communication network of agent i. The parameters c_z represent the weighting factor of the components of the relevance value. Various weighting factors can yield different behaviors in the swarm. Furthermore, the maximum relevance value of the existing information packages in the network is defined as

$$\overline{rel}_{max} = \underset{\overline{rel}_n \in \mathcal{R}_i}{\operatorname{argmax}}(\overline{rel}_n), \tag{33}$$

where $\mathcal{R}_i$ is the set of all the relevance values of the available information packages for the agent i. If the following condition holds

$$\overline{rel}_n \geq 0.95 \cdot \overline{rel}_{max}, \tag{34}$$

the agent considers an information package to be *relevant* and applies the corresponding goal orientation $\theta_i(t_{k+1}) = \theta^c$ in the next time step t_{k+1}. However, if more than one information package fulfills condition (34), the mean value of all corresponding goal orientations from the relevant information packages are determined.

In addition to introduced relevance criteria, the agent checks its distance to the communicated endpoints and corners as follows

$$\begin{aligned} d_s \leq \|\mathbf{p}_i - \hat{\mathbf{p}}_e^c\| \leq R_{rel}, \\ d_s \leq \|\mathbf{p}_i - \hat{\mathbf{p}}_c^c\| \leq R_{rel}, \end{aligned} \tag{35}$$

where $R_{rel} \in \mathbb{R}$ denotes the upper boundary of the relevance range. An endpoint or a corner is then considered if it is within a defined relevance range of the agent. Note: If two different critical points are within the specified range, the closer one is utilized for the next action.

5. Collective navigation using shared information

In the navigation approach for a single robot, we defined several scenarios: tangential navigation, corner avoidance, motion planning at obstacle extremities. In this section, we explain how to adapt these to a multi-agent system by systematically communicating the presented action statuses. Fig. 19 in Appendix A gives an overview about the relations among these statuses. A detailed explanation of the transitions between the action statuses are also given as pseudo codes in Appendix A. At the beginning of the collective navigation, each agent has the relevant information based on the evaluation scheme proposed in Section 4.2. The initial status of the agents is defined as **status 0**. An agent with this status follows the desired goal position $\mathbf{p}_d$ using the control term in Eq. (14) (see Algorithm 1).

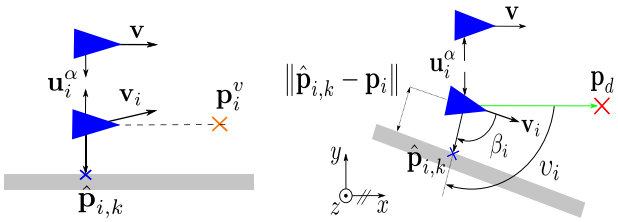
5.1. Collective tangential navigation

If an agent detects an obstacle in the range of r_{tan}, it starts to move toward a virtual goal, tangentially to the obstacle. The calculation of the virtual goal is analogous to the navigation algorithm for a single agent. This action is referred to as **status 1** in the communication network and the agent sends information with $status = 1$ to the neighboring agents (Algorithm 2).

However, the control input $\mathbf{u}_i^\alpha$ according to Eq. (8) intervenes if the ideal distance r between the agents is exceeded. As a result of this, the agent sensing an obstacle can move away from it through attractive forces and leave the tangential navigation range r_{tan} (Fig. 7(a)). This can distort the navigation. In order to prevent this, the agent sensing an obstacle applies only the horizontal component of $\mathbf{u}_i^\alpha$ along the vector $(\mathbf{p}_i^v - \mathbf{p}_i)$. In this way, it is not influenced by forces vertical to the obstacle. The input $\mathbf{u}_i^\alpha$ is reformulated for this case as $\mathbf{u}_i^{\alpha,t}$.

The input $\mathbf{u}_i^\alpha$ can also cause incorrect navigation through an obstacle. If the ideal distance between two adjacent agents is less than the desired distance, $\mathbf{u}_i^\alpha$ generates repulsive forces. This may allow an agent to detect an obstacle unnecessarily and start tracking it (**status 1**). In order to prevent an incorrect transition from **status 0** to **status 1**, we introduced the so-called ignorance condition as follows

$$\begin{aligned} (|\upsilon_i| > 90^\circ \wedge |\beta_i| > 91^\circ) \quad \vee \\ \|\hat{\mathbf{p}}_{i,k} - \mathbf{p}_i\| > (0.3 \cdot (r_{tan} - d_s) + d_s), \end{aligned} \tag{36}$$



(a) Loss of contact. (b) Incorrect navigation.

Fig. 7. Illustration of problems due to $\mathbf{u}_i^\alpha$.

where υ_i denotes the angle between the vector $(\mathbf{p}_d - \mathbf{p}_i)$ and the normal vector to the obstacle. The first part of Eq. (36) ensures that the agent initially moves away from the obstacle and the second part represents an ignorance zone. If condition (36) is satisfied, the agent ignores the obstacle for the motion planning and applies the modified input $\mathbf{u}_i^{\alpha,t}$.

Due to previous information received from the communication network, an agent might be inconveniently oriented once it senses an obstacle. In this case, it changes its goal orientation based on the obstacle detected. This can take some time and the orientation phase is described by **status 4** (Algorithm 5). An agent may have $status = 4$ if it receives new information or detects an obstacle while the following condition is satisfied:

$$|\psi_i - \theta_i| > 45^\circ, \tag{37}$$

where ψ_i is the orientation of the agent.[5]

The communication interface allows agents to calculate a virtual goal position based on information received from the communication network. This case is defined by **status 5** (Algorithm 6). Based on the communicated goal orientation θ^c, the virtual goal position can be determined by the agent according to the following equation:

$$\mathbf{p}_i^v = \mathbf{p}_i + \begin{bmatrix} \cos(\theta^c) & -\sin(\theta^c) \\ \sin(\theta^c) & \cos(\theta^c) \end{bmatrix} \cdot \mathbf{e}_x \cdot s, \tag{38}$$

where $\mathbf{e}_x$ is the unit vector in direction of the x-axis. Agents have a constant tracking acceleration through the positive gain s. In addition, each agent follows an individual temporary goal. The vertical distances among agents are ensured by $\mathbf{u}_i^{\alpha,t}$.

The disadvantage of this approach is that the swarm may perceive the endpoint of an obstacle belatedly. The problem is illustrated in Fig. 8. The red colored agent detects the obstacle and shares the information items θ^c and $\hat{\mathbf{p}}_k^c$ within the network.[6] The first agent follows the obstacle just above the sensing radius r_s, but cannot detect the obstacle itself. Thus, the end of the obstacle can only be perceived belatedly by the red colored agent. We introduce the following conditions to optimize this behavior.

$$\theta_i(t_{k+1}) = \begin{cases} \theta^c - 20^\circ, & \|\tilde{\mathbf{p}}_{i,k} - \mathbf{p}_i\| < 1.5 \cdot r \ \wedge \ \beta_i < 0^\circ \\ \theta^c + 20^\circ, & \|\tilde{\mathbf{p}}_{i,k} - \mathbf{p}_i\| < 1.5 \cdot r \ \wedge \ \beta_i \geq 0^\circ \end{cases} \tag{39}$$

where $\tilde{\mathbf{p}}_{i,k}$ is the orthogonally projected position of the agent i onto the obstacle k, which is described by $\hat{\mathbf{p}}_k^c$ and θ^c as a virtual line. In this way, the agent can estimate its distance to the obstacle. The first condition ensures that there is no other agent between the agent and the obstacle. If it is fulfilled, θ_i is manipulated depending on β_i in the next time step.

[5] ψ_i is the angle between the vector $\mathbf{v}_i$ and the x-axis and measured from the positive x-axis.

[6] The red colored parameters in figures are shared among the neighborhood of agents. Thus, they are known to all agents via communication after some iterations.

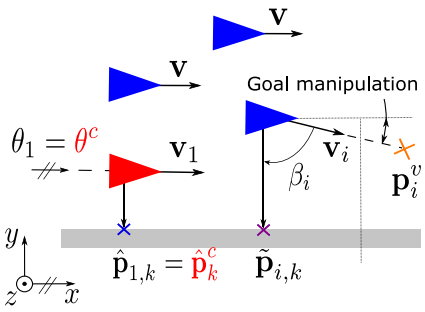
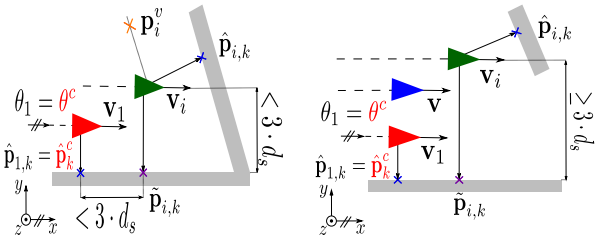


Fig. 8. Distance manipulation of an agent toward an obstacle.

Roboter schätzen, wie weit sie noch vom Hindernis weg sind und passen ihre Fahrtrichtung an, damit sie sich nicht unnötig verheddern



(a) Estimation of a corner via communication.

(b) Waiting for decision-making.

Fig. 9. Collective corner avoidance.

5.2. Collective corner avoidance

The scenario for avoiding corners described in Section 3.2 is represented by **status 3** in the collective navigation (Algorithm 4).

In status 3, an agent detects two obstacle points at the same time. Another scenario is the combination of two information items: a self-detected obstacle point with the range sensor $\hat{\mathbf{p}}_{i,k}$ and a point projected onto an obstacle $\tilde{\mathbf{p}}_{i,k}$ by using shared information items $(\hat{\mathbf{p}}_k^c, \theta^c)$. With the help of these items, an agent builds two intersecting virtual lines and estimates a corner's position (see Fig. 9). However, there might be a passage through which the swarm can proceed. In order to prevent such a misestimation, the following condition is introduced:

$$\|\tilde{\mathbf{p}}_{i,k} - \mathbf{p}_i\| < 3 \cdot d_s \ \wedge \ \|\tilde{\mathbf{p}}_{i,k} - \hat{\mathbf{p}}_k^c\| < 3 \cdot d_s. \tag{40}$$

If the above condition is fulfilled, the agent assumes that a corner exists (see the green colored agent in Fig. 9(a)). Thus, similarly to Section 3.2, it determines $\hat{\mathbf{p}}_{i,n_{90}}$ and $\hat{\mathbf{p}}_{i,n}$ by considering $\tilde{\mathbf{p}}_{i,k}$ and $\hat{\mathbf{p}}_{i,k}$. Then, a new virtual goal position is determined by applying Eqs. (20)–(22).

In this case, the agent changes its status to 4 and it additionally sends information about the corner. If the conditions (40) are not satisfied, the agent stores its current location $\mathbf{p}_i(t_k)$ as $\mathbf{p}_i^{wait}$ and waits at this position until it makes a decision for an orientation by using items from the communication network (Fig. 9(b)). The state of waiting is represented by **status 6** (Algorithm 7). The virtual target is defined in this case as

$$\mathbf{p}_i^v = \mathbf{p}_i^{wait}. \tag{41}$$

At *status 6*, the agent evaluates the relevance of information in the network of his neighborhood. If orientation information with *status* $= 3$ or *status* $= 4$ is declared as relevant, the agent applies it and leaves the status of waiting. If, however, information with another status is classified as relevant, the agent examines whether an information about a nearby endpoint is available in the communication network. If it is, the agent applies the direction of rotation of the circular motion and tracks the obstacle. If there is information with *status* $= 1$ in its network and if the information corresponds to the same obstacle, the agent uses the orientation information and tracks the obstacle.

5.3. Collective motion at obstacle extremities

If an agent detects the endpoint of an obstacle, it follows the virtual target positions defined on a circular path as described in Section 3.3. This navigation approach is represented by **status 2** in the multi-agent setting (Algorithm 3). Here, each agent individually sets virtual goals along a circular path.

Fig. 10 illustrates the described scenario with *status* $= 2$. The group tracks the obstacle tangentially. The red colored agent detects the endpoint of the obstacle and shares the information items $(\hat{\mathbf{p}}_{i,e}, \omega_{i,e}, \boldsymbol{\theta}_{i,e})$ (red colored variables) within its network. Once an agent receives the information about the endpoint of an obstacle via the communication network, it examines its current location to start a circular motion. For this purpose, firstly, it defines a virtual straight line by using the items received $(\hat{\mathbf{p}}_e^c, \omega_e^c)$ from the communication network, the *start line* for the circular motion. By means of orthogonal projection, the front agent, represented in green, can project its current position $\mathbf{p}_i$ onto this line and obtains the point $\mathbf{p}_i^*$ (see Fig. 11(a)). The agent evaluates the following condition before it starts a circular motion:

$$(\mathbf{p}_i^* - \mathbf{p}_i) = -k \cdot (\mathbf{p}_i^v - \mathbf{p}_i), \tag{42}$$

Prüfung ob Roboter Startpunkt für Kreisbewegung erreicht hat

where $k \in \mathbb{R}_{\geq 0}$ is a constant. The agent uses the condition (42) to examine whether it has reached the start line. If the condition is **not** fulfilled, the agent continues to set tangential goal positions and adjusts its velocity by replacing $\mathbf{u}_i^\gamma$ with $\mathbf{u}_{i,e}^\gamma$ as follows:

$$\mathbf{v}_{i,ref} = \frac{\mathbf{p}_i^v - \mathbf{p}_i}{\|\mathbf{p}_i^v - \mathbf{p}_i\|} \cdot \frac{d_s}{r_{i,e}^{max}} \cdot v_{max}, \tag{43}$$

$$\mathbf{u}_{i,e}^\gamma = -c_e^\gamma (\mathbf{v}_i - \mathbf{v}_{i,ref}), \tag{44}$$

where v_{max} represents the predefined maximum allowed speed at the endpoint. In addition, $r_{i,e}^{max}$ denotes the maximum distance of an agent to an obstacle communicated in the neighborhood of agent i. In this way, the speed of the agent is adapted to the minimum possible speed required for the circular motion.

If condition (42) holds, the agent starts to perform a circular motion similar to the presented schema in Section 3.3. During the circular motion, $\mathbf{u}_i^{\alpha,t}$, the component of $\mathbf{u}_i^\alpha$ along the vector $(\mathbf{p}_i^v - \mathbf{p}_i)$, acts again to ensure that the agent reaches its current goal position on the circular path. In addition, each agent examines the angle between the line linking $(\mathbf{p}_i^v, \hat{\mathbf{p}}_{i,e})$ and the line $(\mathbf{p}_i, \hat{\mathbf{p}}_{i,e})$, defined from the vector $(\mathbf{p}_i^v, \hat{\mathbf{p}}_{i,e})$. If the angle has a positive sign (Fig. 11(b), green colored agent), it means the agent has passed its virtual goal and should set a new goal position by applying Eqs. (23)–(24).

Fig. 10 shows that the agents follow an individual circular path with a radius $r_{i,e}$, depending on their distance from the obstacle. As a result, the agents on an inner circular path travel a shorter path than the agents on an outer circular path. The second term in Eq. (8) ensures the velocity consensus for a translational motion. For the case of a circular motion, the consensus is based on the equality of the angular velocities. Thus, in the state of consensus, the minimum desired angular velocity of agents is defined as

$$\omega_{min} = \frac{v_{max}}{r_{i,e}^{max}}, \tag{45}$$

where $r_{i,e}^{max}$ represents the maximum radius for the circular motion. The navigation input $\mathbf{u}_i^\gamma$ is replaced by $\mathbf{u}_{i,e}^\gamma$ described in the following steps

$$\mathbf{v}_{i,ref} = \frac{\mathbf{p}_i^v - \mathbf{p}_i}{\|\mathbf{p}_i^v - \mathbf{p}_i\|} \cdot \omega_{min} \cdot r_{i,e}, \tag{46}$$

Damit alle Roboter in der Runde gleichmäßig drehen, rechnet jeder seine ideale Geschwindigkeit aus

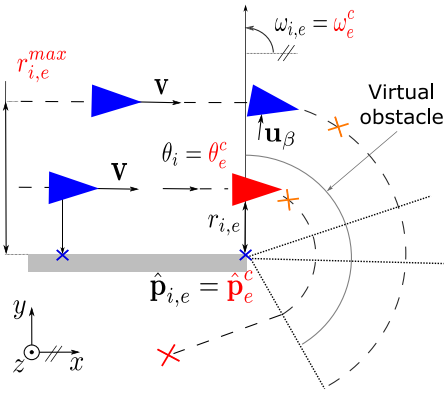


Fig. 10. Collective maneuver at the endpoint of an obstacle.
Roboter fahren gemeinsam um Hindernisende herum und versuchen dabei möglichst eng zusammen zu bleiben

$$\mathbf{u}_{i,e}^{\gamma} = -c_e^{\gamma}(\mathbf{v}_i - \mathbf{v}_{i,ref}), \tag{47}$$

where $\mathbf{v}_{i,ref}$ is the velocity required for the consensus of the angular velocities.

Proposition 1. *A group of agents $G(\mathbf{p}(t))$ with $status_i = 2$ applies (8) and (47). Assume that $G(\mathbf{p}(t))$ is a connected, undirected graph based on positions $\mathbf{A} = (a_{ij}(\mathbf{p}))$ and each agent moves tangentially to its circular path so that the angular velocity of each agent can be described approximately as*

$$\omega_i = \frac{\|\mathbf{v}_i\|}{r_{i,e}}, \tag{48}$$

then

$$\lim_{t\to\infty} \omega_i(t) = \omega_{min}, \quad \forall i \in G, \tag{49}$$

alle Roboter auf einer Kreisbahn passen ihre Drehgeschwindigkeit so an, dass sie gleich schnell im Kreis fahren

i.e., consensus of angular velocities is achieved.

Proof. For the stability analysis of the system, analogous to the proof of [33, Theorem 2], we use moving reference states defined as:

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}_c \quad \text{and} \quad \tilde{\omega}_i = \omega_i - \omega_c,$$

where $\mathbf{p}_c = \frac{1}{N}\sum_{i=1}^{N} \mathbf{p}_i$ is the average position and ω_c denotes the angular velocity of the flock center point. The Hamiltonian representing the structural energy of the system with $status_i = 2$ can be defined as

$$H(\tilde{\mathbf{p}}, \tilde{\omega}) = U(\tilde{\mathbf{p}}) + K(\tilde{\omega}), \tag{50}$$

where $K(\tilde{\omega})$ is the kinetic energy and $U(\tilde{\mathbf{p}})$ denotes the aggregate potential function (cf. [33]). Since we consider a rotational motion, we can define the kinetic energy as

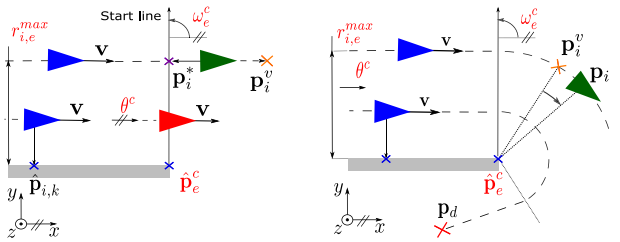
$$K(\omega) = (1/2)\sum_i \omega_i^2. \tag{51}$$

Assume that $\mathbf{v}_{i,ref}$ and $\mathbf{v}_i$ are tangential to the circular path if the angle $\delta/2$ is small enough to apply a small-angle approximation. The structural energy of the system should be monotonically decreasing. The derivative of structural energy of the system can be defined as

$$\dot{H} = -c_e^{\gamma}(\tilde{\omega}^T\tilde{\omega}) - \tilde{\omega}^T\hat{\mathbf{L}}\tilde{\omega} < 0, \tag{52}$$

with $\tilde{\omega} = (\tilde{\omega}_1, \tilde{\omega}_1, \tilde{\omega}_2, \tilde{\omega}_2 \ldots \tilde{\omega}_N, \tilde{\omega}_N)^T \in \mathbb{R}^{2N}$ and $\hat{\mathbf{L}} \in \mathbb{R}^{(2N\times 2N)}$ being a positive semidefinite Laplacian matrix.

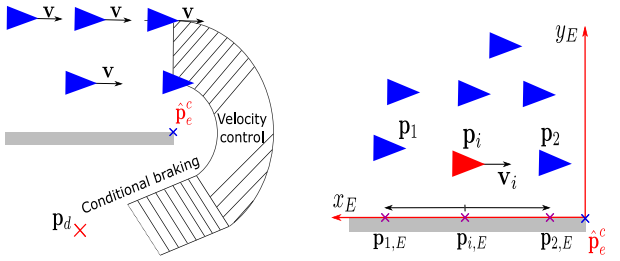
From LaSalle's invariance principle, $\dot{H} = 0$ implies that $\tilde{\omega} = \mathbf{0}$. Therefore, the angular velocity of all agents asymptotically matches $\tilde{\omega}_i = 0 \rightarrow \omega_i = \omega_{min}$.



(a) Crossing the virtual start line.

(b) Inspection during the circular motion.

Fig. 11. Motion planning on a circular path.



(a) Illustration of the velocity control and the conditional braking regions.

(b) Identification of the backmost agent of a group.

Fig. 12. Conditional braking mechanism.

Conditional braking

Each agent leaves the circular path by satisfying condition (25) at different time instants. The acceleration toward $\mathbf{p}_d$ after the circular motion with controlled angular velocity might yield a separation of the system. In order to optimize this behavior, we integrated a conditional braking mechanism that supports the cohesive motion of the group. This is illustrated in Fig. 12(a).

The agents use d_s to calculate the necessary speed that is the lowest possible speed during the circular motion of the swarm. However, they have to determine the *back-most agent* in the group, which is identified by the index $\bar{b}$. The conditional braking is only canceled by the *backmost agent* once it has completed its circular motion. For this purpose, the communication interface is extended by the variables $(\bar{b}, c_{reset})$. The variable c_{reset} denotes the signal for the end of the reformation phase, ($c_{reset} = 0$: velocity control for the reformation, $c_{reset} = 1$: end of the reformation). This signal is communicated by the backmost agent.

In order to identify the *backmost agent*, first, we define a new coordinate system E with the origin at $\hat{\mathbf{p}}_{i,e} = \mathbf{0}_E$, as shown in Fig. 12(b). The orientation of the axis of the new coordinate system is given by the two communicated angles $\theta_{i,e}$ and $\omega_{i,e}$. For further analysis, the current positions of the agents are projected onto the x-axis of the new coordinate system E. The projected position of the agent onto x_E is referred to as $\mathbf{p}_{i,E}$, and the neighboring agents, referred to as $\mathbf{p}_{j,E}$, $j \in \mathcal{N}_i^{\alpha}$ (see Fig. 12(b)).

During the whole circular navigation, each agent examines its neighborhood. In order to determine the backmost member, all agents utilize the index-dependent function given as

$$d(i) = \begin{cases} \|\mathbf{p}_{i,E} - \mathbf{0}_E\|, & \text{if } (\mathbf{p}_{i,E} - \mathbf{0}_E) = k \cdot \mathbf{e}_{x_E} \\ -\|\mathbf{p}_{i,E} - \mathbf{0}_E\|, & \text{if } (\mathbf{p}_{i,E} - \mathbf{0}_E) = -k \cdot \mathbf{e}_{x_E} \end{cases} \tag{53}$$

schaut, welcher Roboter in der aktuellen Bewegungsrichtung am weitesten hinten ist

where $\mathbf{e}_{x_E}$ denotes the unit vector in the positive direction of x_E and $k \in \mathbb{R}_{>0}$ is a positive constant.

Proposition 2. *Each agent can determine the index of the global backmost member $\bar{b}$ with respect to the motion of direction in*

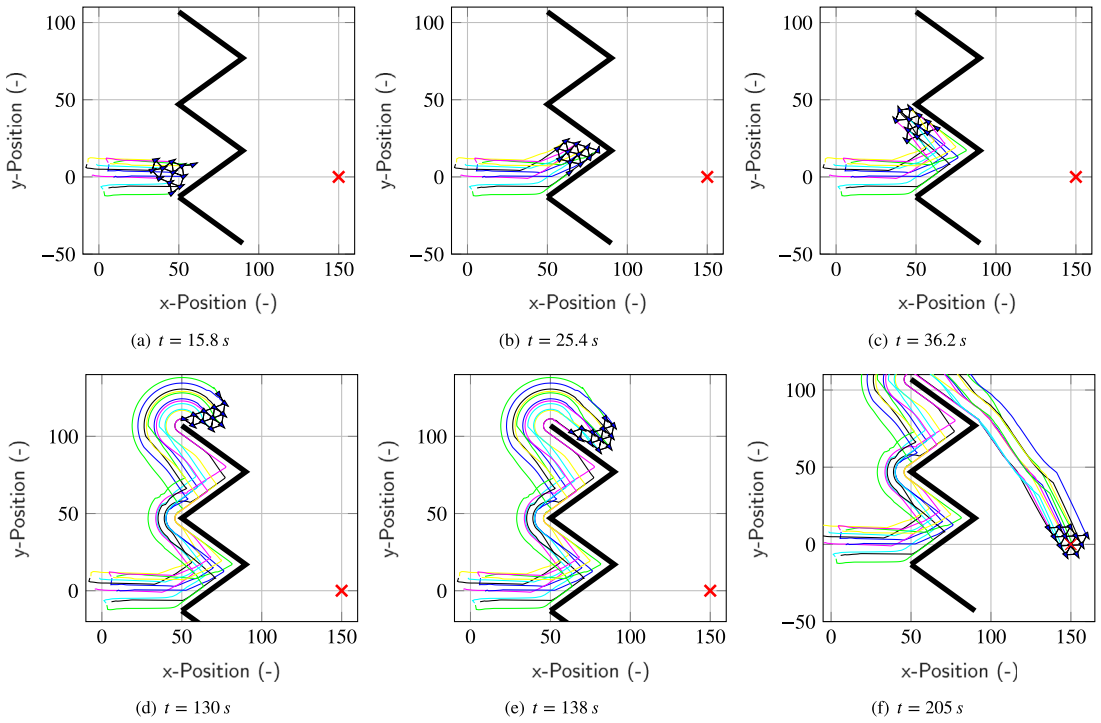


Fig. 13. Consecutive snapshots of the collective navigation with $N = 12$ agents escaping from a zigzag obstacle.

the whole network $G(\mathbf{p}(t))$ if the graph $G(\mathbf{p}(t))$ is connected and undirected.

Proof. Let $b_i(t_k)$ be the index of the determined backmost agent in the communication network of agent i at the time instant t_k and $\mathcal{B}_i$ be the set of all indices in the neighborhood of agent i. For $t_k = 0$ at which the first information about an endpoint is registered, $b_i(0) = i$. Each agent applies a gossip-like communication:

$$b_i(t_{k+1}) = \underset{b_j \in \mathcal{B}_i}{\operatorname{argmax}}\, d(b_j(t_k)), \quad j \in \mathcal{N}_i^{\alpha}, \tag{54}$$

so that the index of the determined backmost member with the greatest d in the neighborhood is adopted. By iterating (54), each agent determines the index of the global backmost member in the whole network,

$$\lim_{t_k \to \infty} b_i(t_k + 1) = \bar{b}, \quad \forall i \in G.$$

Once the global backmost agent identified completes its circular motion, it sends a reset signal $c_{reset} = 1$. The rest of the agents break the conditional braking once they receive the reset signal through the communication.

Watchdog timer

During the collective navigation, some agents can grind to a halt due to communication errors. In this case, the agent hardly moves and cannot reach the target. To recover such malfunctions, each agent is equipped with a self-monitoring mechanism, a so-called watchdog timer. It is activated if the speed of the agent falls below a threshold, e.g., $\|\mathbf{v}_i\| < 0.3$.

When the watchdog timer is activated, the agent saves its current position $\mathbf{p}_{i,wd} = \mathbf{p}_i(t_{i,wd})$ with the time instant $t_{i,wd}$. If the agent moves 1.5 further than the saved location,

$$\|\mathbf{p}_{i,wd} - \mathbf{p}_i(t_k)\| \geq 1.5,$$

and the watchdog timer is still activated, it is deactivated again because the agent assumes that a possible deadlock has broken out.

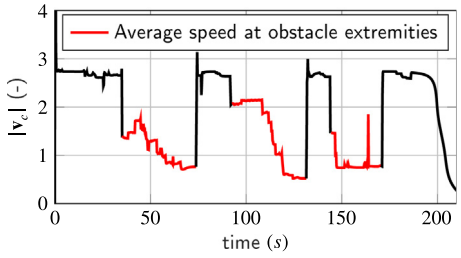


Fig. 14. Average speed of the swarm $|\mathbf{v}_c|$ during the escape from the zigzag obstacle.

If no deactivation signal is registered after 15 s and the agent has moved very little compared to its position at the watchdog activation time,

$$\|\mathbf{p}_{i,wd} - \mathbf{p}_i(t_k)\| < 1.5,$$

status 0 is activated and the agent moves toward the desired goal position.

One drawback of this feature is the fragmentation of the swarm. However, the watchdog timer ensures that each agent reaches the desired goal in finite time despite an environment or communication-based deadlock.

6. Simulation results

In this section, we present the simulation results of the proposed collective navigation approach. We consider two scenarios for demonstrating the effectiveness of the proposed method. The simulation parameters for all scenarios are given in Table 2 and the step size is 0.02 s. There is no predefined primary direction of rotation for the tangential navigation. The navigation task involves $N = 12$ holonomic agents. They are randomly placed in the domain $[-10, 10]^2$ and the initial velocity $\mathbf{v}_i(0) \in \mathbb{R}^2$ of each agent is $(0, 0)^T$.

Table 2
Parameters for the simulation.

Collective behavior and navigation				Communication	
d	7	δ	16°	d_t	0.6 s
r_c	$1.2 \cdot d$	c_1^α	20	d_x	15
d_s	$0.6 \cdot d$	c_1^β	80	d_θ	90°
r_s	$5 \cdot d_s$	c_1^γ	30	c_{type}	5
r_{tan}	$1.2 \cdot d_s$	c_2^α	$2 \cdot \sqrt{c_1^\alpha}$	c_o	1
ϵ	0.1	c_2^β	$2 \cdot \sqrt{c_1^\beta}$	c_{exp}	1
a	5	c_2^γ	$2 \cdot \sqrt{c_1^\gamma}$	c_{dist}	1
b	5	c_e^γ	50	c_t	5
h_α	0.2	v_{max}	4	R_{rel}	30
h_β	0.3	s	100		

In the first scenario, a zigzag obstacle is considered, which might cause the problem of local minimum with potential field-based approaches. The desired goal position is defined at $\mathbf{p}_d = (150, 0)^T$. Fig. 13 shows the consecutive snapshots and trajectories the robots followed during navigation for important time instants. The blue triangles represent the position of robots, and the heading of each triangle specifies the direction of motion of each robot. In addition, the illustrated lines linking robots denote the proximity of agents. At $t = 15.8$ s, an obstacle is detected by the front agents and the group starts the tangential navigation along the border of the obstacle. At $t = 25.4$ s, an agent identifies a corner and through communication, the rest of the group changes its orientation. Then, an endpoint is perceived and the group performs a circular motion (Fig. 13(c) and 13(d)). At such obstacle extremities, robots reduce their speeds to achieve a reliable circular motion by preserving cohesion and collision-avoidance properties. The average speed curve of the swarm is given in Fig. 14. Red curves demonstrate the behavior of the group when encountering obstacle extremities where the robots slow down. Furthermore, at $t = 138$ s, it can be observed that the front agents have approached the obstacle systematically by applying Eq. (39) to optimize the navigation.

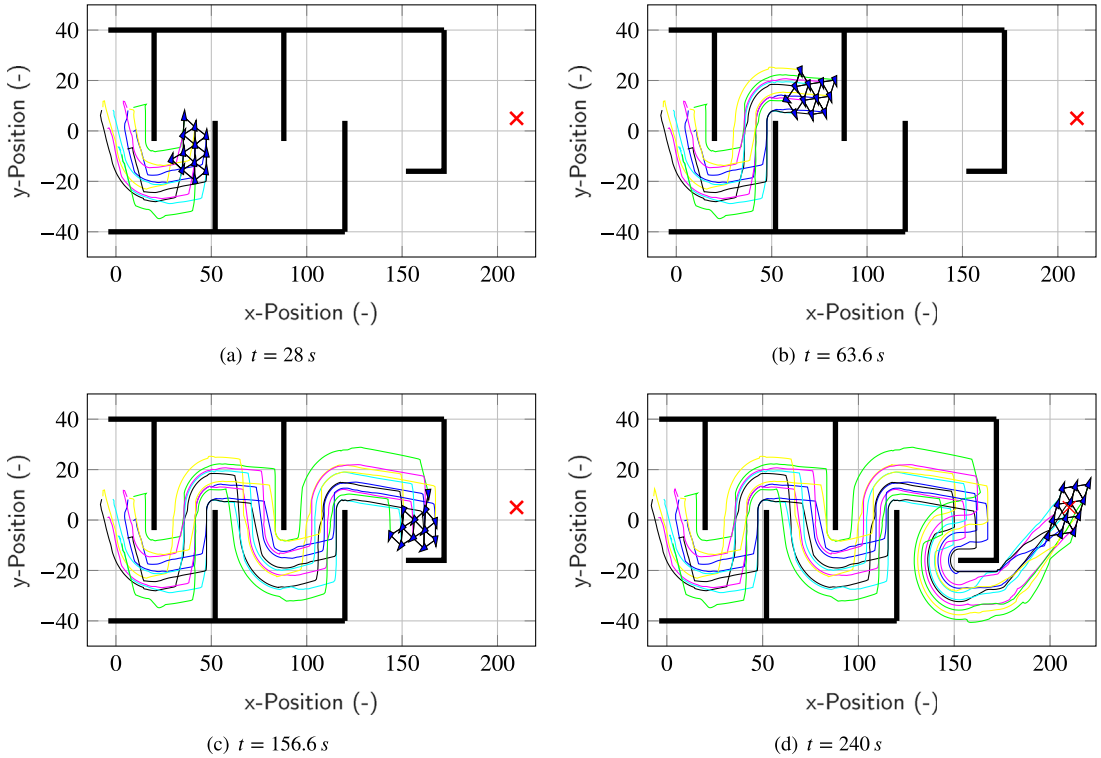
Fig. 15 shows the second selected scenario, in which the same group of robots navigates through a corridor with obstacles. The initial state of robots is the same as in the first scenario. The coordinates of the desired goal position are $\mathbf{p}_d = (210, 5)^T$. The capability of robots to rotate to any direction at obstacles enables agility and flexibility. It is clear that the robots can successfully avoid the obstacles and find a proper path toward the desired goal position applying the collective navigation approach.

Guideline for parameter choice

There are further remarks on the proposed approach that might be useful for the application of our framework. In this approach, the agents share only critical points for motion planning instead of constructing a full map of the workspace by storing all utilized sensing points. This releases the memory of agents. One can easily integrate a memory function to the approach to store all sensed points, or critical points (corners, endpoints) as proposed in [32]. Storing and utilizing critical points would improve motion planning in labyrinths. For more detail, the reader is referred to [32].

Our simulation analyses have revealed that the parameters of the relevance function (d_t, d_x, d_θ) and the weighting factors $(c_{type}, c_o, c_{exp}, c_{dist}, c_t)$ (see Eq. (32)) have a significant influence on the collective behavior of the multi-agent system.

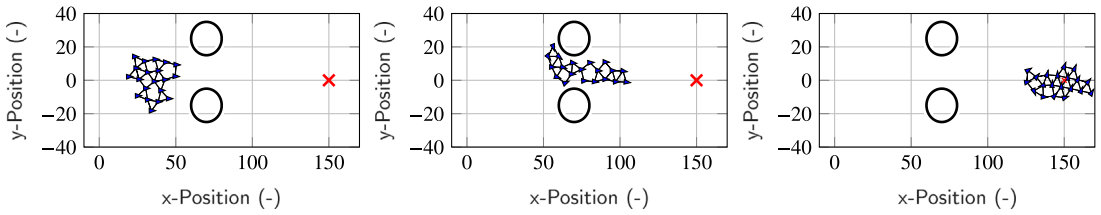
The parameters chosen for our simulations are suitable for small-sized multi-agent systems. In this study, a system with 6–12 agents is defined as small-sized depending on the space between obstacles. The parameter set in this case can be selected so that the MAS exhibits a **global** behavior. This means that all agents perform identical actions approximately at the same time. If, for example, an agent in the group detects a corner, all agents make similar navigation maneuvers. This global behavior is suitable if the distribution of agents in the workspace is small with respect to the space between the obstacles. In order to have a global system behavior, the evaluation of the temporal



(a) $t = 28\ s$ (b) $t = 63.6\ s$

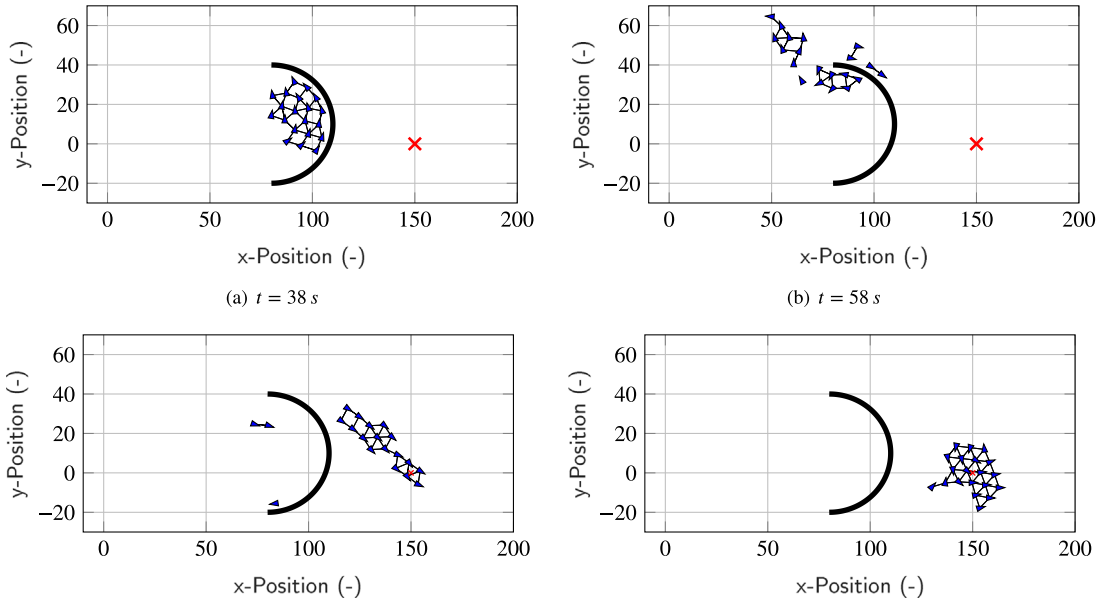
(c) $t = 156.6\ s$ (d) $t = 240\ s$

Fig. 15. Consecutive snapshots of the collective navigation with $N = 12$ agents navigating through a corridor.



(a) $t = 12\,s$ (b) $t = 38\,s$ (c) $t = 70\,s$

Fig. 16. Consecutive snapshots of the collective navigation with $N = 20$ agents — two circular obstacles.



(a) $t = 38\,s$ (b) $t = 58\,s$

(c) $t = 96\,s$ (d) $t = 150\,s$

Fig. 17. Consecutive snapshots of the collective navigation with $N = 20$ agents — semi circular obstacle.

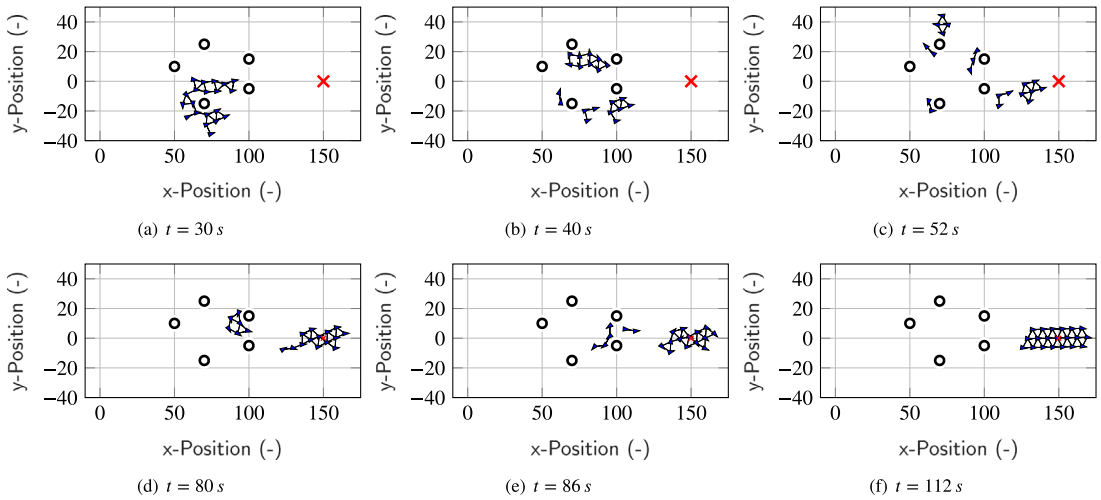
relevance and status of the information is essential for the next action. Hence, the weighting factors of these relevance values, c_t and c_{type}, should be chosen higher than the rest. Highly prioritizing the temporal relevance of an information package (age of information — rel_t) ensures that an agent will consider mainly current information packages. Moreover, assigning a high priority to the evaluation of the actions of the neighboring agents rel_{type} ensures that all agents in the group will consider information packages, which impose a reorientation. In this way, a group of agents can avoid obstacles in a collective manner.

However, agents in large groups, e.g., with 20 agents for our obstacle configuration, often have many relevant information packages at the same time. This means that a global decision might yield suboptimal behaviors. Hence, the agents have to make proper, **local** decisions. In this case, the evaluation of the information relevance should be predominantly based on the individual location of an agent. For this purpose, the relevance of the distance to the obstacle rel_{dist}, the relevance of the expected orientation for the next action rel_{exp} and the relevance of the sender rel_o should be weighted more than rel_{type} and rel_t. This yields the example parameter set given in Table 3: $c_{dist} = c_{exp} = c_o > c_t > c_{type}$. Taking this into account would allow an agent to give greater consideration to information packages received from agents close by and yield only a small deviation from the expected orientation of motion.

Using the communication parameters in Table 3, we perform tests with $N = 20$ agents considering further scenarios, which include a narrow passage, a semi-circular obstacle and many small obstacles. The initial state of robots are chosen in an identical way as in the previous simulations and the desired goal position is defined at $\mathbf{p}_d = (150, 0)^T$. Fig. 16 shows an example for a squeezing maneuver. In this configuration, agents perceive both obstacles via communication and the group can avoid them without splitting up. In Fig. 17, we deal with a semi-circular obstacle. Until the agents arrive at the endpoint, they apply the regular maneuver of tangential navigation. However, apparently at $t = 58\,s$, a communication error occurs because of the complex communication structure and the group splits up. This can also be seen in the video at https://youtu.be/EYEl35pD4H0. By checking action statuses, agents that ignore the endpoint correct their

Table 3
Parameters for large systems.

Communication	
d_t	0.6 s
d_x	15
d_θ	90°
c_{type}	5
c_o	1
c_{exp}	1
c_{dist}	1
c_t	5
R_{rel}	30



(a) $t = 30\,s$ (b) $t = 40\,s$ (c) $t = 52\,s$

(d) $t = 80\,s$ (e) $t = 86\,s$ (f) $t = 112\,s$

Fig. 18. Consecutive snapshots of the collective navigation with $N = 20$ agents — small circular obstacles.

direction of motion and return to the right direction. In this way, they can nevertheless complete navigation as a small group. In Fig. 18, we are concerned with path planning through small circular obstacles, which are narrowly spread in the workspace. Hereby, reacting to different obstacles yields fragmentation of the group. This is a natural behavior because agents are exposed to many different information pieces about the environment and their priority is to avoid collisions with these obstacles. Finally, they still manage to reach the desired goal.

For a further qualitative impression, videos of all simulation scenarios are available at https://youtu.be/EYEl35pD4H0. Additional simulation videos containing scenarios with $N = 100$ agents using the communication parameters in Table 3 can be found at https://www.youtube.com/watch?v=AAynKLMV750.

7. Conclusion

In this paper, we presented our collective navigation strategy for multiple, autonomous robots without a priori knowledge about the environment. The proposed framework is based on a tangential escape schema and information sharing via a communication network. The communication protocol allows multiple robots to efficiently explore an unknown area by exchanging local information about critical points and actions of neighboring robots. Moreover, artificial forces generated through potential fields allow the robots to perform collision-free, cooperative maneuvers and a flock-like behavior.

The groups with a large dispersion relative to the navigation area can tend to fragmentation. In order to optimize this behavior, another parameter set for constants in relevance function and weighting factors can be chosen. Furthermore, in some cases a single agent might lose the connection to the group or to a fragment thereof. However, it can still navigate as a single agent and reach the defined goal position on its own. The proposed navigation approach can also be used for the motion planning of holonomic robots. However, it is important to select a large enough safety distance from obstacles and other robots, and the robots should be operated at a low speed to prevent possible collisions. Since sensitivity of the approach to dimensional changes and communication intensity were not within the scope of this study, it is useful to investigate these when applying the strategy to large-scale groups ($N > 1000$). In the future work, we will focus on the potential to implement this in real time, as well as realize efficient wireless communication. In addition, we will investigate alternative network topologies for efficient interaction among the agents.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research was partially funded by the German Research Foundation (DFG), SFB 768.

Appendix A

See Algorithms 1–7 and Fig. 19.

Algorithm 1 Status 0: Motion toward the desired goal position

Input: $status_i$, Memory buffer
Output: $status_i(t_{k+1})$

while $status_i(t_k) == 0$ **do**
  if $N_i^\beta == 0$ **and** Orientation information is available **then**
    $status_i(t_{k+1}) = 5$
  else if $N_i^\beta == 1$ **and** Eq. (36)$== 0$ **then**
    if Eq. (37)$== 1$ **then**
      $status_i(t_{k+1}) = 4$
    else if Eq. (37)$== 0$ **then**
      $status_i(t_{k+1}) = 1$
    end if
  else if $N_i^\beta == 2$ **then**
    $status_i(t_{k+1}) = 3$
  else
    $status_i(t_{k+1}) = 0$
  end if
end while

einfach:
- solange nix los ist -> fahr zum Ziel
- wenn Hindernis oder wichtige Info -> Situation checken und Status wechseln

Jeder Roboter hat immer genau einen Status. Je nachdem was er sieht und was er per Kommunikation erfährt, wechselt er zw. sieben Zuständen und führt dann den passenden Algorithmus aus.
=> Das macht die Navigation dynamisch, dezentral und robust gegen Hindernisse und schwierige Ecken.

Algorithm 2 Status 1: Obstacle detection and tangential navigation

Input: $status_i$, Memory buffer
Output: $status_i(t_{k+1})$

while $status_i(t_k) == 1$ **do**
  if $N_i^\beta == 0$ **then**
    if Eq. (25)== 1 **then**
      $status_i(t_{k+1}) = 0$
    else if Eq. (25)== 0 **then**
      $status_i(t_{k+1}) = 2$
    end if
  else if $N_i^\beta == 1$ **and** Eq. (36)== 0 **and** Eq. (37)== 1 **then**
    $status_i(t_{k+1}) = 4$
  else if $N_i^\beta == 2$ **then**
    $status_i(t_{k+1}) = 3$
  else if Relevant information about a corner is available (see Eq. (35)) **then**
    $status_i(t_{k+1}) = 3$
  else
    $status_i(t_{k+1}) = 1$
  end if
end while

einfach:
- wenn du am Hindernis bist -> fahr an der Kante entlang
- wenn nix mehr im Weg -> weiter Richtung Ziel

Algorithm 3 Status 2: Motion at the endpoint of an obstacle

Input: $status_i$, Memory buffer
Output: $status_i(t_{k+1})$

while $status_i(t_k) == 2$ **do**
  if $N_i^\beta == 0$ **and** Eq. (25)== 0 **then**
    $status_i(t_{k+1}) = 0$
  else if $N_i^\beta == 1$ **and** Eq. (36)== 0 **then**
    if Eq. (37)== 1 **then**
      $status_i(t_{k+1}) = 4$
    else if Eq. (37)== 0 **then**
      $status_i(t_{k+1}) = 1$
    end if
  else if $N_i^\beta == 2$ **then**
    $status_i(t_{k+1}) = 3$
  else if $c_{reset} == 1$ **then**
    $status_i(t_{k+1}) = 0$
  else
    $status_i(t_{k+1}) = 2$
  end if
end while

einfach:
- am Hindernisende: mach eine Runde drum herum
- wenn fertig -> zurück zum Zielkurs

Algorithm 4 Status 3: Corner avoidance maneuver

Input: $status_i$, Memory buffer
Output: $status_i(t_{k+1})$

while $status_i(t_k) == 3$ **do**
  if $N_i^\beta == 0$ **then**
    $status_i(t_{k+1}) = 0$
  else if $N_i^\beta == 1$ **and** Eq. (36)== 0 **then**
    if Eq. (37)== 1 **then**
      $status_i(t_{k+1}) = 4$
    else if Eq. (37)== 0 **then**
      $status_i(t_{k+1}) = 1$
    end if
  else if $N_i^\beta == 2$ **then**
    $status_i(t_{k+1}) = 3$
  else
    $status_i(t_{k+1}) = 3$
  end if
end while

einfach:
- Ecke erkannt? scharf abbiegen
- wenn erledigt -> Ziel ansteuern oder nochmal ausrichten

Algorithm 5 Status 4: Orientation phase

Input: $status_i$, Memory buffer
Output: $status_i(t_{k+1})$

while $status_i(t_k) == 4$ **do**
  if $N_i^\beta == 0$ **then**
    if Relevant orientation information is available **then**
      $status_i(t_{k+1}) = 5$
    else if No relevant orientation information is available **then**
      $status_i(t_{k+1}) = 0$
    end if
  else if $N_i^\beta == 1$ **and** Eq. (37)== 0 **then**
    $status_i(t_{k+1}) = 1$
  else
    $status_i(t_{k+1}) = 4$
  end if
end while

einfach:
- ich bin grad am Umorientieren; wenn alles klar ist -> weiter zum Ziel oder Hindernis ausweichen

Algorithm 6 Status 5: Tangential navigation based on received information

Input: $status_i$, Memory buffer
Output: $status_i(t_{k+1})$

while $status_i(t_k) == 5$ **do**
  if $N_i^\beta == 0$ **then**
    if Eq. (25)== 1 **then**
      $status_i(t_{k+1}) = 0$
    else if No relevant information is available **then**
      $status_i(t_{k+1}) = 0$
    else if Relevant information about an endpoint is available **and** Eq. (25)== 0 **and** Eq. (42)== 1 **then**
      $status_i(t_{k+1}) = 2$
      Apply Eq. (47)
    else if Relevant information about an endpoint is available (Eq. (35)) **and** Eq. (42)== 0 **then**
      $status_i(t_{k+1}) = 5$
      Apply Eq. (44)
    end if
  else if $N_i^\beta == 1$ **then**
    if Eq. (36)== 0 **and** Eq. (37)== 1 **then**
      $status_i(t_{k+1}) = 4$
    else if A virtual corner is determined (see Fig. 9) **and** Eq. ((40)== 1) **then**
      $status_i(t_{k+1}) = 3$
    else if A virtual corner is determined (Fig. 9) **and** Eq. ((40)== 0) **then**
      $status_i(t_{k+1}) = 6$
    else
      $status_i(t_{k+1}) = 1$
    end if
  else if $N_i^\beta == 2$ **then**
    $status_i(t_{k+1}) = 3$
  else
    $status_i(t_{k+1}) = 5$
  end if
end while

einfach:
- wenn ein anderer was entdeckt hat und teilt, folg ich ihm, aber bleib wachsam für Hindernisse

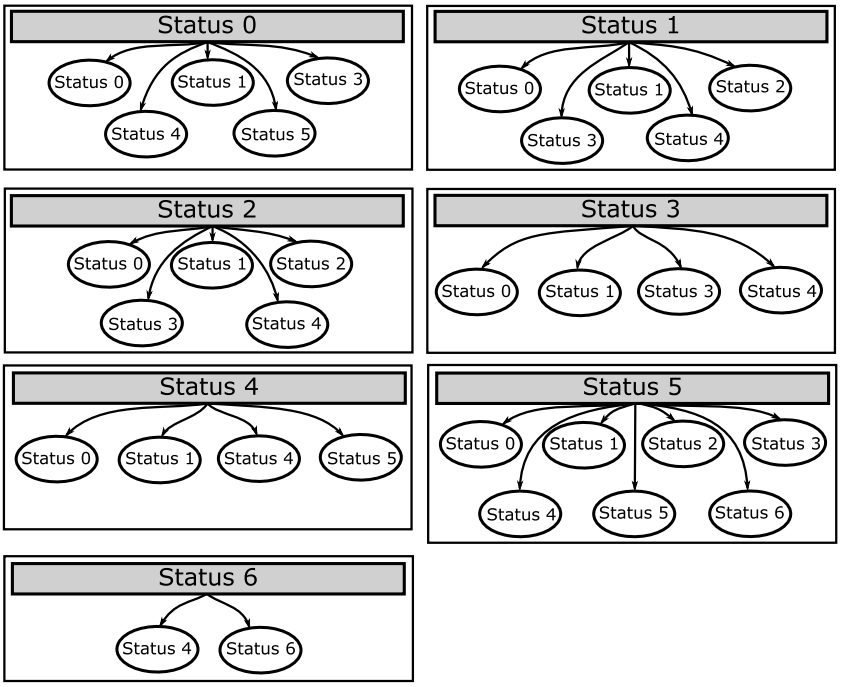


Fig. 19. Relations among the action statuses.

Algorithm 7 Status 6: Waiting mode

Input: $status_i$, Memory buffer
Output: $status_i(t_{k+1})$

while $status_i(t_k) == 6$ **do**
  if Reorientation information ($status = 3$ **or** $status = 4$) is available **then**
    $status_i(t_{k+1}) = 4$
    Apply the corresponding θ^c
  else if Relevant information about an endpoint is available (Eq. (35)) **then**
    $status_i(t_{k+1}) = 4$
    Adopt the direction of rotation of the circular motion for the next time step
  else if Information with $status = 1$ is available **then**
    $status_i(t_{k+1}) = 4$
    Apply the corresponding θ^c
  else
    $status_i(t_{k+1}) = 6$
  end if
end while

einfach:
- unsicher? Bleib stehen, hör dich um; wenn einer was schlaues macht, schließ dich an

References

[1] D.P. Stormont, Autonomous rescue robot swarms for first responders, in: CIHSPS 2005. Proceedings of the IEEE International Conference on Computational Intelligence for Homeland Security and Personal Safety, IEEE, 2005, pp. 151–157.

[2] K. Whitehead, C.H. Hugenholtz, S. Myshak, O. Brown, A. LeClair, A. Tamminga, T.E. Barchyn, B. Moorman, B. Eaton, Remote sensing of the environment with small unmanned aircraft systems (UASs), part 2: Scientific and commercial applications, J. Unmanned Veh. Syst. 2 (3) (2014) 86–102.

[3] F. Cucker, J.-G. Dong, A conditional, collision-avoiding, model for swarming, Discrete Contin. Dyn. Syst. 34 (3) (2014) 1009–1020.

[4] C.W. Reynolds, Flocks, herds and schools: A distributed behavioral model, ACM SIGGRAPH Comput. Graph. 21 (4) (1987).

[5] S. Mou, M. Cao, A. Morse, Target-point formation control, Automatica 61 (2015) 113–118.

[6] M. Cao, B.D. Anderson, A.S. Morse, C. Yu, Control of acyclic formations of mobile autonomous agents, in: 2008 47th IEEE Conference on Decision and Control, IEEE, 2008, pp. 1187–1192.

[7] N. Ganganath, W. Yuan, C. Cheng, T. Fernando, H.H.C. Iu, Territorial marking for improved area coverage in anti-flocking-controlled mobile sensor networks, in: IEEE International Symposium on Circuits and Systems, ISCAS, 2018.

[8] N. Ganganath, C.-T. Cheng, K.T. Chi, Distributed antiflocking algorithms for dynamic coverage of mobile sensor networks, IEEE Trans. Ind. Inf. 12 (5) (2016) 1795–1805.

[9] S.H. Semnani, O.A. Basir, Semi-flocking algorithm for motion control of mobile sensors in large-scale surveillance system, IEEE Trans. Cybern. 45 (1) (2015) 129–137.

[10] S. Knorn, Z. Chen, R.H. Middleton, Overview: Collective control of multiagent systems, IEEE Trans. Control Netw. Syst. 3 (4) (2015) 334–347.

[11] E. Olcay, K. Gabrich, B. Lohmann, Optimal control of a swarming multi-agent system through guidance of a leader-agent, IFAC-PapersOnLine 52 (20) (2019) 1–6.

[12] A. Borzi, S. Wongkaew, Modeling and control through leadership of a refined flocking system, Math. Models Methods Appl. Sci. 25 (02) (2015) 255–282.

[13] D. Mellinger, A. Kushleyev, V. Kumar, Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams, in: 2012 IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 477–483.

[14] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107.

[15] M. Milford, G. Wyeth, Hybrid robot control and SLAM for persistent navigation and mapping, Robot. Auton. Syst. 58 (9) (2010) 1096–1104.

[16] E. Olcay, B. Lohmann, Extension of the cucker-dong flocking with a virtual leader and a reactive control law, in: 18th European Control Conference, ECC, IEEE, 2019, pp. 101–106.

[17] M. Guerra, D. Efimov, G. Zheng, W. Perruquetti, Avoiding local minima in the potential field method using input-to-state stability, Control Eng. Pract. 55 (2016) 174–184.

[18] M. Okamoto, M. Akella, Novel potential-function-based control scheme for non-holonomic multi-agent systems to prevent the local minimum problem, Internat. J. Systems Sci. 46 (12) (2013) 2150–2164.

[19] J. Lee, Y. Nam, S. Hong, W. Cho, New potential functions with random force algorithms using potential field method, J. Intell. Robot. Syst. 66 (3) (2012) 303–319.

[20] N.E. Leonard, E. Fiorelli, Virtual leaders, artificial potentials and coordinated control of groups, in: Proceedings of the 40th IEEE Conference on Decision and Control, Vol. 3, IEEE, 2001, pp. 2968–2973.

[21] J. Borenstein, Y. Koren, et al., The vector field histogram-fast obstacle avoidance for mobile robots, IEEE Trans. Robot. Autom. 7 (3) (1991) 278–288.
[22] I. Ulrich, J. Borenstein, VFH+: Reliable obstacle avoidance for fast mobile robots, in: Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), Vol. 2, IEEE, 1998, pp. 1572–1577.
[23] C. Lum, J. Vagners, M. Vavrina, J. Vian, Formation flight of swarms of autonomous vehicles in obstructed environments using vector field navigation, in: Proceedings of the International Conference on Unmanned Aircraft Systems, 2012.
[24] T. Mercy, R.V. Parys, G. Pipeleers, Spline-based motion planning for autonomous guided vehicles in a dynamic environment, IEEE Trans. Control Syst. Technol. 26 (6) (2018) 2182–2189.
[25] M. Futterlieb, V. Cadenat, T. Sentenac, A navigational framework combining visual servoing and spiral obstacle avoidance techniques, in: 2014 11th International Conference on Informatics in Control, Automation and Robotics, ICINCO, Vol. 2, IEEE, 2014, pp. 57–64.
[26] J. Vilca, L. Adouane, Y. Mezouar, Reactive navigation of a mobile robot using elliptic trajectories and effective online obstacle detection, Gyroscopy Navig. 4 (1) (2013) 14–25.
[27] A. Dang, H.M. La, T. Nguyen, J. Horn, Distributed formation control for autonomous robots in dynamic environments, 2017, URL https://arxiv.org/abs/1705.02017.
[28] J. Fax, R.M. Murray, Information flow and cooperative control of vehicle formations, IEEE Trans. Automat. Control 49 (9) (2004) 1465–1476.
[29] E. Olcay, B. Lohmann, M.R. Akella, An information-driven algorithm in flocking systems for an improved obstacle avoidance, in: 45th Annual Conference of the IEEE Industrial Electronics Society, IECON, Vol. 1, IEEE, 2019, pp. 298–304.
[30] Z. Qu, Cooperative Control of Dynamical Systems: Applications to Autonomous Vehicles, Springer Science & Business Media, 2009.
[31] J. Alonso-Mora, T. Naegeli, R. Siegwart, P. Beardsley, Collision avoidance for aerial vehicles in multi-agent scenarios, Auton. Robots 39 (1) (2015) 101–121.
[32] A.S. Brandão, M. Sarcinelli-Filho, R. Carelli, An analytical approach to avoid obstacles in mobile robot navigation, Int. J. Adv. Robot. Syst. 10 (6) (2013) 278.
[33] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, IEEE Trans. Automat. Control 51 (3) (2006) 401–420.
[34] R. Olfati-Saber, Distributed Kalman filtering for sensor networks, in: 2007 46th IEEE Conference on Decision and Control, IEEE, 2007, pp. 5492–5498.

Ertug Olcay received the B.Sc. and M.Sc. degrees in Mechanical Engineering from Technical University of Munich, Germany, in 2013 and in 2016. He is currently a research assistant at the Chair of Automatic Control, Technical University of Munich, where he works toward his Ph.D. degree. His research interests are swarm intelligence, self-organization, cooperative control and motion planning.

Fabian Schuhmann is currently pursuing his Master's degree in Mechanical Engineering at the Technical University of Munich. His fields of interest are centered around automation and control.

Boris Lohmann is a professor and the head of the Chair of Automatic Control, Department of Mechanical Engineering, Technical University of Munich. His research interests are linear and non-linear control theory, modeling and model reduction, and control engineering applications in mechatronics, automotive and plant automation.