PADERBORN UNIVERSITY
The University for the Information Society

Forming Large Patterns from Widespread Swarms of Oblivious Robots with Limited Visibility

Dissertation

In partial fulfillment of the requirements for the degree of

Doctor rerum naturalium (Dr. rer. nat.)

at the Faculty of Computer Science,
Electrical Engineering and Mathematics
at Paderborn University

submitted by

JONAS HARBIG

Reviewers

- Prof. Dr. Friedhelm Meyer auf der Heide,
  Paderborn University

- Prof. Dr. Christian Scheideler,
  Paderborn University

The world is changing:
I feel it in the water,
I feel it in the earth,
I smell it in the air.

Tolkien [65]

Abstract

Within the last years, exploration of hazardous environment (e.g. the Mars) by robot swarms became more and more relevant and theoretical research on robot swarms with restricted abilities increased. However, fundamental problems such as ARBITRARY-PATTERN-FORMATION, where the swarm should self-organize to a predefined geometric pattern, remained difficult under strong restrictions. Therefore, this thesis considers disoriented robots with limited visibility and presents a novel solution to the ARBITRARY-PATTERN-FORMATION problem. The well established $\mathcal{O}$BLOT model is assumed; robots are all identical, compute their decisions locally, and cannot communicate or store persistent information (oblivious). They act according to the $\mathcal{F}$SYNC scheduler, where robots execute fully-synchronously LOOK-COMPUTE-MOVE cycles. Disoriented means that each robot observes its surrounding in a self-centered, arbitrarily rotated coordinate-system. With limited visibility, only robots within a constant radius can be observed.

ARBITRARY-PATTERN-FORMATION (APF) considers a swarm, where all n robots know a pattern in advance and must obtain it in arbitrary rotation and translation. The problem is already solved for oblivious robots with global visibility [69]. The possibility of forming a pattern depends on its *symmetricity*, a measurement of the rotational symmetries that counts how often a pattern matches itself while rotating it around itself. The symmetricity of the pattern must be an integer multiple of the swarms symmetricity, otherwise APF is not solvable in general; this is called *symmetry condition* in the following. [72] presents a solution of APF for robots with (arbitrary large) memory and limited visibility, but the swarm can only form a pattern with diameter ≤ 1: First, the swarm is contracted to a diameter ≤ 1 (this is called NEAR-GATHERING) without changing its symmetricity, and then the formation of the pattern from [69] is applied to form a small pattern that satisfies the symmetry condition. Note that limited visibility naturally restricts the results to *connected* swarms, i.e. where the unit-disc-graph is connected. None of the mentioned results holds for oblivious robots with limited visibility.

This thesis extents the results from [69, 72] to oblivious robots with limited visibility. First, a class called λ-contracting NEAR-GATHERING protocols is presented, which are to our knowledge the first NEAR-GATHERING protocols under this model. A NEAR-GATHERING protocol transforms a *widespread* swarm (diameter can be much larger than 1) into a *contracted* swarm (diameter ≤ 1). Secondly, as far as we know the first method for analyzing the symmetricity change induced by a protocol is presented, and two concrete symmetricity preserving protocols are given. One protocol always preserves symmetricity, but leads only for certain swarms to NEAR-GATHERING. The other one preserves symmetricity only when the Connectivity-Boundary (the set of robots surrounding the swarm) is convex and the swarm does not contain a 1-hole (circle of diameter 1 without a robot) but always leads to a NEAR-GATHERING. Thirdly, to the best knowledge the first APF protocol for oblivious robots with limited visibility is presented that forms any *large* connected pattern (diameter can be much larger than 1) from any contracted initial configuration that fulfills the symmetry condition above.

To round up the results, they are generalized for APF with widespread patterns – a significantly more difficult problem as robots have to coordinate themselves outside their viewing range. The three results above do not require memory. However, it is shown in this thesis that the formation of a large pattern is not in general possible for oblivious robots with limited visibility if the swarm is widespread (even though both are connected and the symmetry condition is fulfilled). But by introducing just one single bit of memory it is possible to combine the results such that a large connected pattern can be formed from a widespread swarm. The combined result only holds for swarms that fulfill the symmetry condition and where NEAR-GATHERING preserves symmetricity, i.e. that have a convex Connectivity-Boundary and contain no 1-hole.

Zusammenfassung

In den letzten Jahren ist die Erkundung von gefahrenreichen Umgebungen (z.B. dem Mars) durch Roboterschwärme immer relevanter gewordern und die Forschung an Roboterschwärmen mit eingeschränkten Fähighkeiten hat zugenommen. Allerdings sind fundamentale Probleme wie ARBITRARY-PATTERN-FORMATION (zu deutsch: Beliebige-Muster-Bildung), in dem ein Schwarm selbst organisiert ein vorab definiertes geometrisches Muster bilden soll, nach wie vor schwer zu lösen für stark eingeschränkt Roboter. Daher betrachten diese Arbeit orientierungslose Roboter mit eingeschränkter Sichtweite und präsentiert eine neue Lösung des ARBITRARY-PATTERN-FORMATION Problems. Das $\mathcal{O}$BLOT Modell wird angenommen, in dem Roboter identisch sind, ihre Entscheidungen lokal treffen und weder miteinander kommunizieren noch Informationen dauerhaft speichern können (erinnerungslos). Roboter agieren in vollkommen synchronen Zyklen ($\mathcal{F}$SYNC), in denen zunächst alle Roboter ihre Umgebung beobachten, dann eine Bewegung auf Basis der Beobachtung berechnen und diese schließlich ausführen. Orientierungslos heißt, dass jeder Roboter seine Umgebung in einem selbstzentrierten und beliebig gedrehtem Koordinatensystem wahrnimmt. Mit eingeschränkter Sichtweite können nur Roboter in einem konstanten Radius beobachtet werden.

Das ARBITRARY-PATTERN-FORMATION (APF) Problem betrachtet einen Schwarm, in dem alle n Roboter ein Muster $P \in \mathbb{R}^{2n}$ kennen und dieses in beliebiger Rotation und Translation bilden müssen. Das Problem ist bereits für erinnerungslose Roboter mit globaler Sicht gelöst [69]. Es wurde gezeigt, dass es von der *Symmetricity* des Musters abhängt, ob dieses gebildet werden kann. Symmetricity ist eine Einheit für die Rotationssymmetrien einer Punktmenge; sie zählt, wie häufig die Menge mit sich selbst übereinstimmt, während sie einmal um sich selbst gedreht wird. Die Symmetricity des Musters muss ein ganzzahliges Vielfaches der Symmetricity des Schwarmes haben, damit das Muster gebildet werden kann; dies wird im Folgenden als *Symmetriebedingung* bezeichnet. [72] präsentiert eine Lösung des APF Problems für Roboter mit (beliebig großem) Speicher und begrenzter Sichtweite, allerdings muss das Muster einen Durchmesser von ≤ 1 haben. Zunächst wird der Schwarm zusammengezogen zu einem Durchmesser ≤ 1 (das wird NEAR-GATHERING genannt, zu deutsch: Nahes-Versammeln) ohne dass sich seine Symmetricity ändert. Dann wird das Protokoll aus [69] angewendet, um ein kleines Muster zu bilden, welches die Symmetriebedingung erfüllt. Die begrenzte Sichtweite schränkt die Ergebnisse auf Schwärme ein, deren Unit-Disc-Graph zusammenhängt. Keines der genannten Ergebnisse gilt für erinnerungslose Roboter mit eingeschränkter Sichtweite.

Diese Dissertation erweitert die Ergebnisse von [69, 72] für orientierungslose $\mathcal{O}$BLOT Roboter mit eingeschränkter Sichtweite. Erstens: Eine Klasse, genannt λ-kontrahierende NEAR-GATHERING Protokolle, wird präsentiert, welche nach bestem Wissen die ersten bekannten NEAR-GATHERING Protokolle für dieses Modell sind. Ein NEAR-GATHERING Protokoll verwandelt einen *weitverteilten* Schwarm (der Durchmesser kann viel größer als 1 sein) in einen *kontrahierten* Schwarm (Durchmesser ≤ 1). Zweitens: Die, soweit wir wissen, erste Methode zur Untersuchung der Symmetricity-Änderung aufgrund eines Protokolls wird präsentiert. Außerdem werden zwei konkrete Symmetricity-erhaltende Protokolle angegeben. Das eine erhält in allen Fällen die Symmetricity, führt aber nur in einigen Fällen zu NEAR-GATHERING. Das andere erhält die Symmetricity nur, wenn die Connectivity-Boundary (die Menge der Roboter, die den Schwarm umschließen) konvex ist und der Schwarm kein 1-Loch (ein Kreis mit Durchmesser 1 ohne Roboter) beinhaltet, führt aber in jedem Fall zu NEAR-GATHERING. Drittens: Das, unseres Wissens nach, erste ARBITRARY-PATTERN-FORMATION Protokoll für dieses Modell wird präsentiert, welches jedes *große* und zusammenhängende Muster (der Durchesser kann viel größer als 1 sein) von jedem kontrahierten Schwarm bilden kann, wenn die Symmetriebedingung erfüllt ist.

Um die Ergebnisse abzurunden werden sie generalisiert für APF mit weitverteilten Schwärmen – ein signifikant schwereres Problem, da Roboter sich koordinieren müssen ohne sich gegenseitig sehen zu können. Die drei oben genannten Resultate benötigen keinen Speicher. Allerdings wird in dieser Arbeit gezeigt, dass das Bilden eines großen Musters nicht im Generellen für erinnerungslose Roboter mit eingeschränkter Sichtweite möglich ist, wenn der Schwarm weitverteilt ist (auch dann nicht, wenn beide zusammenhängend sind und die Symmetriebedingung erfüllt ist). Mit einem Bit Speicher ist es jedoch möglich, die Ergebnisse zu kombinieren, sodass ein großes zusammenhängendes Muster von einem weitverteiltem Schwarm gebildet werden kann. Das kombinierte Ergebnis gilt nur für Schwärme, welche die Symmetriebedingung erfüllen und in denen NEAR-GATHERING die Symmetricity erhält, d.h. die eine konvexe Connectivity-Boundary haben und kein 1-Loch enthalten.

Preface

Writing this thesis was a long way. I would never have gotten this far without the help of my friends, family, and colleagues. First of all, I would like to thank my supervisor Friedhelm Meyer auf der Heide, who guided me through the Bachelor and Master and gave me the opportunity to research at his chair afterwards. I am thankful that you always pushed me towards my research goals and helped me find new problems when I was standing on the spot. You supported me even after your retirement and made sure that I could seamlessly move to the chair of Christian Scheideler.

Next, I would like to thank Christian Scheideler and his whole chair for the warm welcome. After many of my colleges and my supervisor left the university at the same time, you integrated me into your group. You gave me excellent research conditions and a great social environment such that I could finish long-awaited results. Sincerely, I also would like to thank Peter Kling, who supported me during this time as a co-author.

Apart from Peter and Friedhelm, I had several great co-authors: Jannik Castenow, Raphael Gerlach, Sören von der Gracht, Christopher Hahn, Daniel Jung, Till Knollmann. Especially, I would like to thank Daniel, who supervised my Bachelor thesis, and Jannik, who supervised my Master thesis, both in the field of swarm algorithms. Swarm algorithms fascinated me from this point on, which ultimately led me to pursue a Ph.D. in this field. Jannik, you showed me the art of writing scientific papers and helped me in the beginning of my Ph.D. to find the right research direction. Jannik and Till, it was a great time in the shared office with you!

Most of the time during my Ph.D. I was part of the DFG-project “Algorithms for Swarm Robotics: Distributed Computing meets Dynamical Systems”. I would like to thank my project partners Prof. Michael Dellnitz, Raphael Gerlach and Sören von der Gracht for the great and productive research meetings that finally lead to some of my results.

Many thanks also to Andre Graute, my dear coffee fellow, and Jan-Luca Hansel, who moved together with me from Friedhelm to Christian and shared the office with me for the last two years. I had a lot of fun with you!

Sincerely, I would like to thank all people who helped me in writing down this thesis by providing me with a layout class, helping me out with LaTeX tips, as well as proofreading everything in the end. Without you, it would not look so good.

In addition, I would like to thank my family and, especially, Klara my girlfriend. Klara, you accompanied me the longest time during the Ph.D., not only as my partner but as a

fellow Ph.D.-student as well. You were a role model for me, pursuing your Ph.D. with the greatest patience and determination imaginable. You were always there for me when I needed you most, and you supported me in writing this thesis, especially during the last weeks when I was more a ghost, writing and haunting our home, than a loving partner.

Jonas Harbig
June 2025

Contents

1. Introduction

Swarm robots are small robots that cooperate in swarms of large numbers to perform a single task. There are various areas where such robots would be useful. One example is rescue and exploration applications in hazardous environments like the surface of planets [45] or earthquake regions. These robots should be small enough to fit into cracks or rockets and cheap enough so that the loss of a few robots is not devastating. Another area is medicine in which small robots could be injected into the bloodstream for precise surgery or drug administration [63]. One central task for a swarm is the deployment of individual robots where they are needed. In an exploration scenario the robots should for example spread evenly around the whole area to survey it. For precise surgery robots must for example cover the parts of the body that are to be removed. In general one can say the desired outcome of the deployment can be expressed as a set of coordinates called *pattern*. However, due to space constraints or practicability, the robots cannot always be placed as desired by an external force (like a drone). Therefore, they should be able to reach the set of coordinates on their own after being released in an arbitrary arrangement (e.g. dumped out of a bag). The process of reaching a pattern from an arbitrary arrangement in the field of swarm algorithms this is called PATTERN-FORMATION.

There exist several small robots developed for applications in larger swarms. Most famous are the NASA Bee Bots [45], which are designed to survey large areas of Mars. Examples developed for research on swarm behavior are "Jasmine" [48] from the University of Stuttgart and "Kilobots" [61] from Harvard University. Although the examples mentioned above are small robots (around the size of a coin), they are still large enough to include powerful communication equipment. Currently, there are commercially available UMTS and GPS chips smaller than a coin [66, 67]. UMTS allows all robots in a swarm, to communicate with a central server which says each robot, what to do, and where to move. Together with a common coordinate system obtained from GPS, this makes the formation of a pattern trivial. However, there are several scenarios in which the robots cannot be equipped with GPS and UMTS (or similar technologies). First, those technologies require a high degree of infrastructure in form of transmission masts or satellites; these are not necessarily available in certain situations, e.g. on the Mars, in a deep cave or after an earthquake. Secondly, those technologies require a non-negligible amount of energy. Small robots can only carry equally small batteries; therefore, the operation of a swarm can be extended without GPS and UMTS. Last but not least, the size of communication chips limits the size of the robots. There exist research on nanobots [40, 42], robots in the scale of nanometers; if nanobots will eventually be developed, they will likely be too small to contain technologies like UMTS and GPS.

This thesis analyzes the capabilities of swarms of robots that are not able to communicate or determine their positions in a global coordinate system. It uses mathematical and geometrical methods and provides a theoretical answer to the question *Can a swarm of such robots form any in advanced specified pattern?* In the following, the pattern formation problem is formally introduced and motivated on the basis of literature and known results. This contains a short overview of the considered model (a comprehensive description of the model is given later in Section 1.3). Afterwards in Section 1.2, an outline of this thesis is presented that contains a summary of its results.

1.1 Introduction into Pattern Formation Problems

Pattern formation for a swarm of autonomous robots is a widely studied field in theoretical computer science. Initially, the swarm is in an arbitrary configuration (i.e., an unorganized arrangement of robots). The goal is that the swarm reaches a specified final configuration. An often considered example for a pattern formation problem is GATHERING where all robots have to move onto the same not predefined position [24]. There are various results that consider one *specific pattern* as a problem. Examples include LINE-FORMATION [9], where robots must form a straight line, and CIRCLE-FORMATION [32], where a regular n-gon is formed by n robots (see Figure 1.1). To follow the above described applications of swarm robots, GATHERING can be used to collect the swarm (for example to recharge the robots) and CIRCLE-FORMATION can be used to enclose an area such that nothing enters or leaves unnoticed. In the following a brief description of the model is given before pattern formation is formally introduced (see Section 1.3 for a comprehensive description of the model).

Model Summary This thesis considers *point shaped* (no mass or area) mobile robots in the Euclidean plane according the $\mathcal{O}$BLOT model [31]. The $\mathcal{O}$BLOT model (short for "$\mathcal{O}$BLivious robOT") assumes that robots are moving entities that are *oblivious* (have no persistent memory), *identical* (perform the same protocol $\mathcal{P}$ and are indistinguishable by any means beside their positions) and *autonomous* (compute the protocol $\mathcal{P}$ locally). Robots are *disoriented*, they observe their surroundings as relative positions in an arbitrarily rotated self-centered coordinate system where the rotation can change arbitrarily over time. Robots have a common understanding on the unit distance and sense of rotation (*chirality*). In addition, they are *deterministic* (cannot use randomness) and *synchronous* (act according to the $\mathcal{F}$SYNC scheduler). Robots act in discrete time steps where every robot performs an LCM-cycle consisting of LOOK (taking a snapshot of the surrounding), COMPUTE (computing an action based on the snapshot specified by the protocol $\mathcal{P}$) and MOVE (executing the computed action). A robot can move up to distance 1 during the MOVE phase. The $\mathcal{F}$SYNC scheduler specifies that the phases of the LCM-cycle are completely synchronous; the cycles are counted as *rounds*. Oblivious means that the robot's memory is erased after each MOVE; while most results in this thesis consider oblivious robots, Theorem 9 assumes one bit of persistent memory.

A *configuration* $\mathbf{z}^t \in \mathbb{R}^{2n}$ describes the positions of n swarm robots at time t. We call

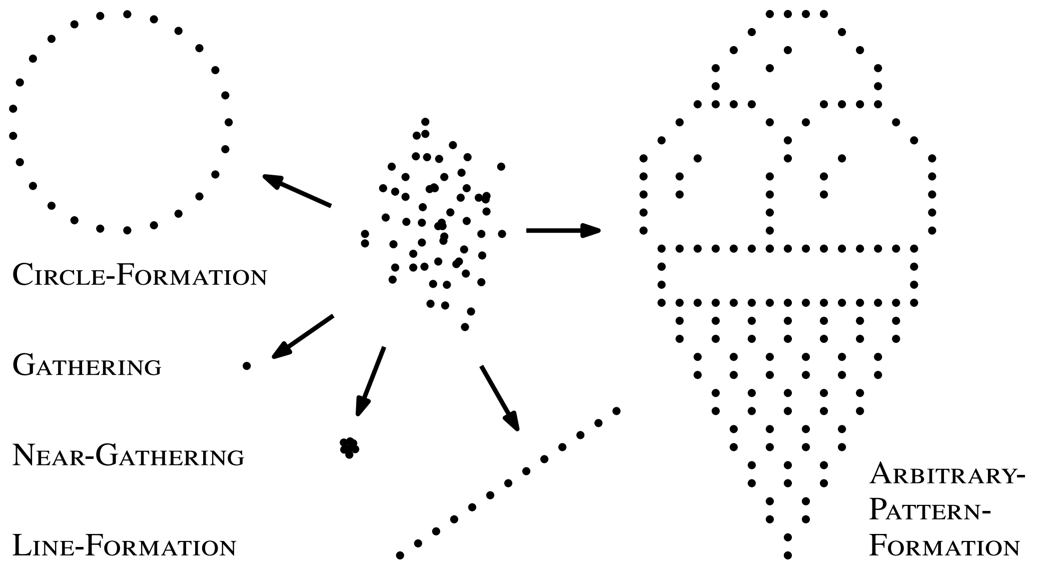


Figure 1.1: Examples of CIRCLE-FORMATION, GATHERING, NEAR-GATHERING, LINE-FORMATION, and ARBITRARY-PATTERN-FORMATION.

a configuration *connected* if the unit disc graph (UDG) with distance 1 is connected. Robots have a *limited visibility*, the snapshot taken during LOOK contains other robots only within a constant distance called *viewing range* around itself. The viewing range varies throughout the thesis between 1 and $2+\sqrt{2}$.

A comprehensive description of the model and its implication can be found in Section 1.3, further notations are defined in Section 1.4.

A *pattern* $P \in \mathbb{R}^{2n}$ in the context of pattern formation problems is the desired final configuration of the swarm. The model specifies that each robot computes its movement locally. Combining all local movements leads to a change from configuration $\mathbf{z}^t$ to $\mathbf{z}^{t+1}$. This is defined as the evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ with $F(\mathbf{z}^t) = \mathbf{z}^{t+1}$ for $t \geq 0$. For the GATHERING and LINE-FORMATION problem above we can define the pattern *point* and *line* as follows.

- $point := \big((0,0),\cdots,(0,0)\big)$
- $line := \big((0,0),(0,1),\cdots,(0,n-1)\big)$

The mentioned results on GATHERING, LINE- and CIRCLE-FORMATION have in common that the given protocols form only one specific pattern. A more general approach is the ARBITRARY-PATTERN-FORMATION, introduced in [64]. There, the protocol of the robots is designed agnostic of the pattern. The pattern P is known by all robots only at execution, one may assume P as an additional static argument passed every COMPUTE to the local computation of each robot (respectively, the evolution function $F(\mathbf{z}^t,P)$). The robots have to form P in arbitrary rotation and translation. The problem is formally defined as follows.

Definition 1.1 — ARBITRARY-PATTERN-FORMATION (APF). A local protocol with evolution function F solves the ARBITRARY-PATTERN-FORMATION if for every initial configuration $\mathbf{z}^0 \in \mathbb{R}^{2n}$ and every pattern $P \in \mathbb{R}^{2n}$ there exist a rotation matrix M_ρ, translation matrix M_τ, permutation matrix M_κ and round $t' \in \mathbb{N}$ such that $M_\rho M_\tau M_\kappa P = \mathbf{z}^t$ for all $t \geq t'$.

R Note that in the considered model not all patterns can be formed by all initial configurations. Instead, it is characterized for every pair of pattern and initial configuration, whether a formation is possible or not.

ARBITRARY-PATTERN-FORMATION under $\mathcal{O}$BLOT. The first result for ARBITRARY-PATTERN-FORMATION considering the $\mathcal{O}$BLOT model with global visibility was presented in [34, 69]. A key aspect that determines whether a pattern can be formed is its symmetry. The authors introduced a measure called *symmetricity* that describes the symmetry of a pattern or swarm configuration.

Definition 1.2 — Symmetricity [34, 64]. Consider $P \in \mathbb{R}^{2n}$ whose smallest enclosing circle is centered at $c \in \mathbb{R}^2$. An *m-regular* partition of P is a partition of P into $k = |P|/m$ regular m-gons with common center c. The *symmetricity* of P is defined as $\mathrm{sym}(P) := \max\{m \in \mathbb{N} \mid \text{there is a } m\text{-regular partition of } P\}$.

In this definition, a single point is considered as a 1-gon with arbitrary center. Therefore, any P has a 1-regular partition. Loosely spoken, the symmetricity describes the number of times a pattern/configuration matches itself while turning it a full rotation around itself.

Symmetricity allows characterizing patterns that can be formed by synchronous oblivious robots with an unlimited viewing range.

Theorem 1 — Symmetry Condition, [34, Theorem 1]. A pattern $P \in \mathbb{R}^{2n}$ can be formed by $\mathcal{O}$BLOT robots with unlimited viewing range in the $\mathcal{F}$SYNC model from configuration $\mathbf{z}^0 \in \mathbb{R}^{2n}$ if and only if $\mathrm{sym}(\mathbf{z}^0)$ divides $\mathrm{sym}(P)$.

The impossibility of forming a pattern with incompatible symmetricity follows directly from our robot model. If you have disoriented robots that are identical and deterministic, it is not possible to decrease the symmetricity. Two robots that are in symmetric positions obtain essentially the same snapshot during the LOOK-phase. Therefore, they compute the same (simply rotated) movement and maintain the symmetricity. Symmetricity can be increased in such a model, but only by integer factors. Even for robots with global visibility and persistent memory the limitation based on symmetricity cannot be circumvented [64].

APF with limited visibility. A similar result exists for robots with limited visibility and persistent memory [72]. There, the swarm first performs NEAR-GATHERING (move the robots close together) such that all robots have mutual visibility. Then, the protocol for oblivious robots with global visibility [69] is used to form the pattern with a diameter smaller than the viewing range. The paper leaves two questions unanswered.

Can a pattern be formed on a large scale if robots have limited visibility?

Is it possible to form (small) patterns with oblivious robots that have limited visibility?

The persistent memory in [72] is only used during NEAR-GATHERING. Therefore, the formation of small patterns with oblivious robots can be done if NEAR-GATHERING is possible without memory.

1.1.1 Near-Gathering

NEAR-GATHERING is related to the GATHERING problem, both move robots closer together. In contrast to GATHERING, robots must in the end obtain unique positions (i.e., do not collide with other robots). The goal is to make the swarm configuration have a constant diameter. This is, strictly speaking, not a pattern formation problem because the target configuration cannot be explicitly stated as a list of coordinates. The target configuration is a Diam-c-Configuration defined as follows.

Definition 1.3 — Diam-c-Configuration. A Diam-c-Configuration is a configuration of robots with diameter $\leq c$ without collisions (i.e., each robot obtains a unique position).

Note, that a swarm must avoid collisions throughout the entire execution to end up in a configuration without collisions. This follows directly from the robot model, where robots are defined to be oblivious, identical, disoriented and deterministic. If robots are on the same position, they would always compute the exact same target position in such a model and therefore never end up on different positions in the future. Using the definition above one can define NEAR-GATHERING as follows.

Definition 1.4 — NEAR-GATHERING. The NEAR-GATHERING problem is solved, if there exists a constant c and a protocol that reaches from every collision-free configuration a Diam-c-Configuration and all robots terminate simultaneously.

It is defined, that the protocol must terminate in an instant, i.e., all robots decide simultaneously (in the same round) with local knowledge that a Diam-1-Configuration is reached. On one hand, this distinguishes NEAR-GATHERING from the related CONVERGENCE problem, where robots converge towards the same position (but neither necessarily reach it not detect it). On the other hand, the simultaneous termination makes NEAR-GATHERING especially useful, because afterwards APF (or more generally speaking: another protocol) can start. Otherwise, it is possible that some robots are still contracting, while others have already started the next protocol.

Because the symmetricity is of importance for what patterns can be formed, the symmetricity of the initial configuration must not change during the NEAR-GATHERING. The authors of [34] use the memory of the robots to ensure that the symmetricity is not changed. Robots are only indistinguishable when they are disoriented and have the same current memory. The above NEAR-GATHERING protocol takes advantage of memory by storing all snapshots. If the initial configuration has a different symmetricity than the final Diam-c-Configuration, at least two robots must end up at on symmetric positions (i.e., on the same m-gon, see symmetricity Definition 1.2) that have not originated at symmetric positions. Those must have seen at least one different snapshot, which is stored in their memory. This difference can be used to move those robots away from the m-gon in a way that restores the original symmetricity.

In [57] a NEAR-GATHERING protocol is presented for oblivious robots with limited visibility and compass. The protocol works only for configurations that are *well-connected*, which is defined as follows.

Definition 1.5 — Well-Connectedness. Let v be the viewing range of the robots and $\sigma > 0$ be an arbitrary small constant. A configuration is called *well-connected* if its unit-disc-graph with unit-distance $v - \sigma$ is connected.

The presence of a compass (agreement on a common direction "north") immediately eliminates all rotational symmetries. This makes it possible to form all patterns (not only patterns with fitting symmetricity).

Gathering. There exist results solving gathering for oblivious robots with limited visibility that have the potential to be complimented with a collision-avoidance method. Most notably is the GO-TO-THE-CENTER (GTC) protocol [24]. There, robots move towards the center of the smallest enclosing circle (SEC) of the locally observed robots. Robots usually do not move exactly onto this center but only half way to keep the unit-disc-graph (UDG) connected. Only if the SEC has a small diameter, the robots move exactly to the center. This property is called *collapsing*. It makes sure that eventually a GATHERING is reached because otherwise robots would in general only converge towards one point but not necessarily reach it. One could assume that it is sufficient to remove the collapsing property from GTC to obtain a NEAR-GATHERING protocol. However, the correctness proof from [24] requires the protocol to be collapsing, therefore the correctness of a non-collapsing GTC is not yet known.

There exists also a notable result in a continuous-time model. It considers robots as points with a movement vector. The movement vector changes continuously, depending on the positions of the robots within local visibility. [52] presents a class of protocols called *contracting*. It consists of all protocols where the movement vectors point inside the convex hull of the swarm. The authors proved that all *contracting protocols* solve CONVERGENCE if robots move with constant speed. They additional provide a NEAR-GATHERING protocol in this class. The *contracting* CONVERGENCE protocol known as "n-bug problem", that is further analyzed in [35], is of particular interest when trying to translate this result to $\mathcal{F}$SYNC. There, n robots are placed in a regular n-gon (or more general: a circular graph), and each robot moves in the direction of its clockwise neighbor. Intuitively, one may think that this results in a circular motion. But instead, the robots move in spirals towards the center. In a discrete-time model (e.g. $\mathcal{F}$SYNC) however, the robots will in fact move in circles without ever coming nearer to the center. Therefore, this class of *contracting protocols* cannot be simply translated from a continuous time model into $\mathcal{F}$SYNC.

Problem 1 Is it possible to perform NEAR-GATHERING with oblivious, disoriented robots under limited visibility?

1.1.2 Symmetry Preservation

We motivated above that the possibility to solve ARBITRARY-PATTERN-FORMATION is highly dependent on the symmetricity of the initial configuration. Hence, any NEAR-GATHERING protocol that is used as a first step towards APF must not change the symmetricity of the swarm. If NEAR-GATHERING is possible in general, the next question is about symmetricity preservation. [72] proves, that it is not possible to preserve the symmetricity in general with limited visibility when the robots are oblivious and have *non-rigid* movement (can be stopped by an adversary during the movement). The argument is the following. Let us consider the configuration depicted in Figure 1.2 that has a symmetricity of 1. Only the black robots can move without violating transitive visibility. Robots a and b are the only two robots, that are not in symmetric positions to another robot. They have to move towards their only visible neighbor, the adversary can stop them such that they end up on positions symmetric to each other and the swarm increased its symmetricity to 2. Note that this argument does not hold for swarms with *rigid* movement (robots always reach their computed target positions).

Problem 2 Does a NEAR-GATHERING protocol for oblivious, disoriented robots with limited visibility and rigid movement exist that does not change the symmetricity of the swarm?

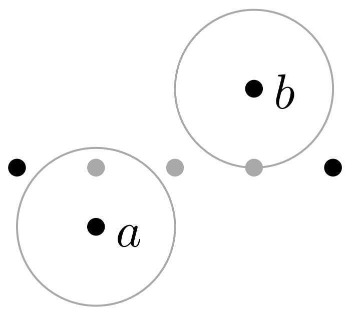


Figure 1.2: A counter example why robots with non-rigid movement and local visibility cannot preserve symmetricity in general. The viewing ranges of a and b are drawn as gray circles.

There is very little research on symmetry preservation considering models with disorientation and limited visibility. In fact, to the best of our knowledge, there are no known non-trivial protocols[1] under these models that preserve symmetry.

1.1.3 Large Pattern Formation

The second question at the beginning was about the size of the pattern. It is easy to see that the formation of a large pattern by disoriented, oblivious robots with limited visibility is not possible. This is shown briefly in the following. Assume a sufficiently large cross as the target pattern P and a similar cross as the initial configuration $\mathbf{z}$. If both crosses have different ratios between the two beams as in Figure 1.3, it is not locally observable for a robot with limited visibility whether it is in P or $\mathbf{z}$. This leads directly to the following observation.

Observation 1.1 ARBITRARY-PATTERN-FORMATION for disorientated $\mathcal{O}$BLOT robots with limited visibility is not always solvable, even if the symmetry condition from Theorem 1 is matched.

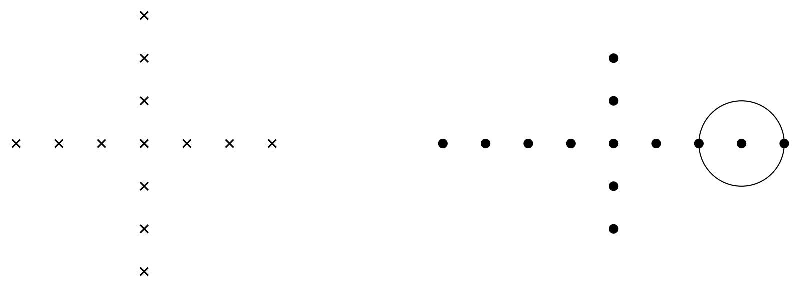


Figure 1.3: An example of configuration $\mathbf{z}$ (right), where robots cannot distinguish locally whether the formation of pattern P (left) already finished. The viewing range is exemplary marked.

However, one can use one bit of memory to overcome the sketched problem above. If all robots know in the beginning that the protocol is in its "initial phase" they can start with the pattern formation. At some point during the execution they can flip this bit, to remember that the initial phase has ended and continue with the formation of the pattern.

[1]A trivial protocol is DO-NOTHING where each robot remain at its initial position forever.

It is reasonable to use NEAR-GATHERING as the initial phase, because in a Diam-1-Configuration all robots can observe the global state of the swarm without (larger amounts of) memory and communication.

Let us assume that the initial phase terminated in a Diam-1-Configuration and all robots set their bit accordingly. All robots can use the methods from [69] to compute the position that they must obtain in the pattern (in their local coordinate system). The pattern may have a diameter larger than the visibility range. So, eventually, robots need to move towards these positions such that the "global visibility" ends (i.e., the swarm reaches a diameter larger than the viewing range). Without memory and without a fixed compass, the computed position in the pattern as well as a direction towards it cannot be maintained. [55] provides a method to use three robots (called TuringMobile) in a way that the relative positions between them encode the current state and the belt content of a Turing machine. They further describe a protocol that lets the TuringMobile move around in a way that collects all other robots (eventually, solving NEAR-GATHERING) and places them elsewhere to solve ARBITRARY-PATTERN-FORMATION. However, they assume that one TuringMobile can be identified in the initial configuration. The TuringMobile is designed so that it has no rotational symmetries (symmetricity of 1). If it is unambiguously identifiable, the entire configuration must have a symmetricity of 1. Because symmetries are one of the key obstacles in forming patterns, this is a huge restriction for a general result on ARBITRARY-PATTERN-FORMATION.

Problem 3 Is it possible to form any large pattern from any Diam-1-Configuration where the general limitation of Theorem 1 holds with oblivious disoriented robots and limited visibility?

1.2 Outline of the Thesis and Main Results

In this thesis, we consider the ARBITRARY-PATTERN-FORMATION problem for *disoriented, oblivious* robots according to the $\mathcal{O}$BLOT model acting *fully-synchronous* with *limited visibility*. It fills the gaps of [72] by answering the two questions.

Can a pattern be formed on a large scale if robots have limited visibility?

Is it possible to form (small) patterns with oblivious robots that have limited visibility?

The technique for APF with limited visibility from [72] is adapted to oblivious robots. The robots first performed a NEAR-GATHERING to reach a Diam-1-Configuration and formed a *small pattern* (diameter ≤ 1) afterwards. Remember, that the NEAR-GATHERING utilizes large persistent memory by storing all observed snapshots to preserve the symmetricity. The formation of a small pattern afterwards exploits the global visibility in a Diam-1-Configuration to apply the APF-protocol from [69] (without using memory). This thesis adapts their results, by presenting a APF protocol that first performs NEAR-GATHERING, but without any memory, and afterwards forms a large patter (with a diameter up to n), again without persistent memory. As already stated in Observation 1.1 the whole process of forming a large pattern from an arbitrary connected swarm is not in general possible in the here considered model (especially with obliviousness, disorientation and limited visibility). This is circumvented by the introduction of one bit of persistent memory. While both stages of the APF protocol do not need any memory, the robots need this bit to remember in what stage they currently are.

The results are split into three chapters, each introducing novel solutions to the Problems 1–3. First, a class of NEAR-GATHERING strategies for oblivious robots with limited visibility is presented (Chapter 2 answering Problem 1). Remember that a NEAR-GATHERING must not change the symmetricity of the configuration to be used as a first stage for an APF protocol because the possibility of forming a pattern is highly dependent on the symmetricity of the swarm (see Theorem 1). The second result of this thesis introduces methods on how to analyze the impact of protocols on symmetricity. Two concrete strategies are given that preserve symmetricity. One preserves symmetry in general but not always leads to a Diam-1-Configuration; the other preserves symmetry only for a restricted class of initial configurations but solves NEAR-GATHERING in general (Chapter 3 answering Problem 2). Thirdly, a protocol that forms large patterns but requires the initial configuration to be a *contracted swarm* (i.e., a Diam-1-Configuration) is presented (Chapter 4 answering Problem 3).

A detailed summary of the three results is given in the following (Sections 1.2.1–1.2.3), each subsection restates the problem it is about in the beginning. These summaries give overviews of central parts of the proofs. The three results are combined in the end of this section (Section 1.2.4) to follow the main theorem of this thesis, that states under what conditions a *large pattern* (diameter $>> 1$) can be formed by a *widespread swarm* (diameter $>> 1$). A comprehensive description of the model is given in Section 1.3 and specific notations used in this thesis is defined in Section 1.4. Afterwards an overview on related work can be found in Section 1.5.

1.2.1 Near-Gathering (Chapter 2)

Problem 1 Is it possible to perform NEAR-GATHERING with oblivious, disoriented robots under limited visibility?

The NEAR-GATHERING problem is solved by introducing the class of λ-contracting NEAR-GATHERING protocols (Definition 2.3) that lead to a Diam-1-Configuration. They have three important properties; they are λ-contracting (Definition 2.2) and collision-free. λ-contracting protocols are similar to the continuous protocols in [23]. The target position of a robot (where it moves) must not only lie inside the convex hull of the swarm, but must be sufficiently far from the corners of the convex hull (to prevent circular movement as described above). The following result is shown.

Theorem 2 Consider a swarm of robots in $\mathbb{R}^2$ with diameter Δ that is *well-connected* Every λ-contracting NEAR-GATHERING protocol solves NEAR-GATHERING after $\mathcal{O}\left(\Delta^2\right)$ rounds.

We also show that well-connectedness is a natural requirement for any NEAR-GATHERING protocol in this model (Observation 2.1). While NEAR-GATHERING is a relevant problem of its own, it can be used as an intermediate step to solve GATHERING-WITHOUT-EARLY-COLLISION and CIRCLE-FORMATION. GATHERING-WITHOUT-EARLY-COLLISION is a variant of GATHERING where collisions are only allowed in the very last round [52]. Simultaneous termination allows the robots to move onto one position in the round after reaching a Diam-1-Configuration.

Theorem 3 Consider a swarm of robots in $\mathbb{R}^2$ with diameter Δ that is *well-connected* Every λ-contracting NEAR-GATHERING protocol solves GATHERING-WITHOUT-EARLY-COLLISION in $\mathcal{O}\left(\Delta^2\right)$ fully-synchronous rounds.

Similarly, they could form a small circle in the next round, to solve CIRCLE-FORMATION. Other patterns are not formable in general as a next step, because the protocol class makes no guaranties on symmetricity changes during the execution.

There exist several λ-contracting protocols, that are presented in the doctoral thesis of Jannik Castenow [7]. Most notable, the GO-TO-THE-CENTER protocol [2, 24] is one of them. In this thesis a collision avoidance method is introduced, such that any λ-contracting protocol can be transformed into a λ-contracting NEAR-GATHERING protocol.

Theorem 4 For every λ'-contracting protocol $\mathcal{P}'$ there exist a λ-contracting NEAR-GATHERING protocol $\mathcal{P}$ with $\lambda \in \mathcal{O}(\lambda')$.

In Theorem 12 is stated, that GO-TO-THE-CENTER is a concrete example of a λ-contracting protocol that can be transformed into a λ-contracting NEAR-GATHERING protocol. The results on λ-contracting NEAR-GATHERING protocols were first published in

A Unifying Approach to Efficient (Near)-Gathering of Disoriented Robots with Limited Visibility [10]
Jannik Castenow, Jonas Harbig, Daniel Jung, Peter Kling, Till Knollmann and Friedhelm Meyer auf der Heide
Conference on Priciples of Distributed Systems (OPODIS) 2022

The paper also contains results on GATHERING (with early collisions) that are part of Jannik Castenows doctoral thesis [7]. Chapter 2 includes more details on the shared authorship.

1.2.2 Symmetricity Preservation (Chapter 3)

Problem 2 Does a NEAR-GATHERING protocol for oblivious, disoriented robots with limited visibility and rigid movement exist that does not change the symmetricity of the swarm?

As argued above, NEAR-GATHERING is a useful step towards ARBITRARY-PATTERN-FORMATION. However, to obtain a result comparable to Theorem 1, it is necessary that the symmetricity during NEAR-GATHERING does not increase. To the best of our knowledge, there exists no research on symmetry changes induced by local protocols under the $\mathcal{O}$BLOT model. The second result of this thesis introduces to the best of our knowledge the first method to analyze the impact of a protocol on the symmetricity of a swarm. It makes use of invertibility of the protocol's evolution function. The evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ defines the behavior of a swarm from the perspective of a global observer. If F is invertible, a global observer can compute where the robots were a round before. The following theorem is shown.

Theorem 5 Consider an arbitrary swarm protocol with evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$. Assume that F is (locally) invertible (Definition 3.3). Then, any configuration $\mathbf{z} \in \mathbb{R}^{2n}$ and its successor configuration $\mathbf{z}^+ := F(\mathbf{z})$ have the same symmetricity $\mathrm{sym}(\mathbf{z}^+) = \mathrm{sym}(\mathbf{z})$.

The intuition for this is the following. Symmetricity can never decrease for disoriented oblivious robots as mentioned above to explain Theorem 1. However, it can increase by an integer factor i. Assume $\mathbf{z}^+$ has the predecessor configuration $\mathbf{z}$ with $\mathrm{sym}(\mathbf{z}^+) = i \cdot \mathrm{sym}(\mathbf{z})$ for $i > 0$. When you rotate $\mathbf{z}$ by $^{360°}/_{i}$ its successor configuration is a rotation of $\mathbf{z}^+$ by $^{360°}/_{i}$ and therefore equivalent to $\mathbf{z}^+$ because $\mathrm{sym}(\mathbf{z}^+)$ is devisible by i. F is not invertible for $\mathbf{z}^+$ if it has multiple predecessor configurations. Therefore, (local) invertibility is a sufficient condition for the preservation of symmetricity.

The theorem is applied to create two symmetricity preserving protocols. One is based on the GO-TO-THE-AVERAGE protocol where all robots move towards the average of the locally observed robots (Algorithm 4). GO-TO-THE-AVERAGE is known to not maintain connectivity, the swarm will split into multiple clusters that each converge towards one position (in case the swarm keeps connectivity a Diam-1-Configuration is reached, see Lemma 3.7). However, the protocol is a demonstration that there exist non-trivial protocols that preserve symmetricity for all initial configurations.

Theorem 6 The execution of ε-GO-TO-THE-AVERAGE (Algorithm 4) protocol for $\varepsilon < \frac{1}{29}$ does not change the symmetricity of the swarm.

Note, that ε-GO-TO-THE-AVERAGE contain n, the number of robots in a swarm, as a parameter. This global knowledge is implicitly available for NEAR-GATHERING that is used as part of the ARBITRARY-PATTERN-FORMATION problem, because the pattern consist of n positions as defined in Definition 1.1.

The second protocol, called WAVE-PROTOCOL, solves NEAR-GATHERING for all initially connected configurations. It is based on GO-TO-THE-MIDDLE, a protocol where each robot moves towards the midpoint between its neighbors. The Connectivity-Boundary of a configuration consists of all robots that are near the outside of a swarm (see Definition 1.6 for a formal definition and Figure 1.4 for an example). Only robots near to the swarm's Connectivity-Boundary move in WAVE-PROTOCOL. The robots on the Connectivity-Boundary perform GO-TO-THE-MIDDLE and other robots nearby move slightly to prevents collisions. A hole inside the configuration is a circle without robots (see Definition 1.7 for a formal definition and Figure 1.4 for an example). A larger hole makes it impossible for robots to locally detect whether they are on the Connectivity-Boundary or not. However, to proof the invertibility of WAVE-PROTOCOL the invariant is required that only robots on the Connectivity-Boundary perform GO-TO-THE-MIDDLE. This leads to the following result on symmetricity preservation.

Theorem 7 We assume robots according the $\mathcal{O}$BLOT model and $\mathcal{F}$SYNC scheduler with a viewing range of $2+\sqrt{2}$ in a swarm with a convex Connectivity-Boundary (Definition 1.6) and no 1-holes (Definition 1.7). The WAVE-PROTOCOL (Algorithm 7) leads to NEAR-GATHERING and does not change the symmetry.

The restriction for the initial configurations are based on the symmetricity analysis.

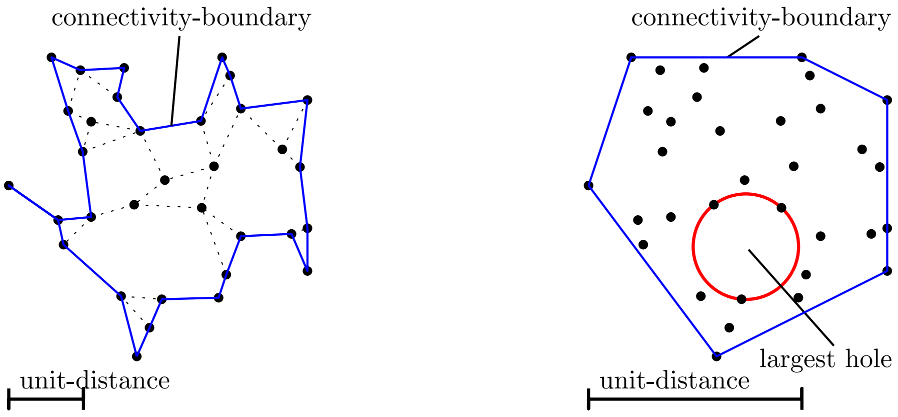


Figure 1.4: Left: a configuration with a non-convex Connectivity-Boundary and its unit-disc-graph. Right: A configuration with a convex Connectivity-Boundary and its largest hole.

The evolution-function of GO-TO-THE-MIDDLE is linear and invertible if it is applied to a closed chain of robots. In a swarm with a convex Connectivity-Boundary and no 1-holes robots can locally decide whether they are on the Connectivity-Boundary and the protocol can be designed in a way that the chain of robots performing GO-TO-THE-MIDDLE does not change. This allows to follow from the invertibility of GO-TO-THE-MIDDLE that WAVE-PROTOCOL is invertible. The large viewing range of the protocol is necessary such that robots near the Connectivity-Boundary can detect this move as well (to stay inside the Connectivity-Boundary and to prevent collisions). For arbitrary connected configurations (with larger holes and without a convex Connectivity-Boundary) it is shown in Lemma 3.17, that the protocol still solves NEAR-GATHERING (but not necessarily preserves symmetricity).

In total, GO-TO-THE-AVERAGE and WAVE-PROTOCOL complement each other to give novel results on symmetricity preserving NEAR-GATHERING protocols. The first protocol is in general symmetricity preserving but leads to a Diam-1-Configuration only for certain configurations. The latter protocol is only analyzed for a subset of configurations to preserve symmetricity, but it solves NEAR-GATHERING in general[2]. It is still open whether a protocol exists, that solves NEAR-GATHERING while preserving symmetry for any initially connected configuration. The results were first published in

Symmetry Preservation in Swarms of Oblivious Robots with Limited Visibility [36]
Raphael Gerlach, Sören von der Gracht, Christopher Hahn, Jonas Harbig and Peter Kling
Conference on Priciples of Distributed Systems (OPODIS) 2024

For the results on symmetry preservation methods from the mathematical theory of dynamical systems are used. The application of these methods is largely contributed by the co-authors of the paper.

[2]Note that the λ-contracting NEAR-GATHERING protocol from Chapter 2 are in general superior in therms of viewing range and running time if a NEAR-GATHERING protocol is needed without requirements on symmetry preservation.

1.2.3 Large Pattern Formation from Contracted Swarms (Chapter 3)

Problem 3 Is it possible to form any large pattern from any Diam-1-Configuration where the general limitation of Theorem 1 holds with oblivious disoriented robots and limited visibility?

The results on ARBITRARY-PATTERN-FORMATION with global visibility and oblivious robots [69] and on limited visibility and remembering robots [72] are further generalized to limited visibility and oblivious robots. Complementing the results on NEAR-GATHERING discussed above that lead to a Diam-1-Configuration the following result is shown.

Theorem 8 A connected pattern P can be formed by $|P|$ disoriented $\mathcal{O}$BLOT robots with limited viewing range in the $\mathcal{F}$SYNC model from a Diam-1-Configuration $\mathbf{z}$ if and only if $\mathrm{sym}(\mathbf{z}) \mid \mathrm{sym}(P)$. The formation takes $\mathrm{O}(|P|)$ rounds, which is worst-case optimal.

The DRAWING-PROTOCOL, a protocol for ARBITRARY-PATTERN-FORMATION, is provided for this model. It uses the protocol for global visibility [69] in the first round to form $\mathrm{sym}(\mathbf{z})$ many *drawing formations*, small configurations of a subset of robots that encode an enumeration in $\mathcal{O}(|\mathbf{z}|)$, as well as the orientation of a common coordinate system. The drawing formations then each form one part of the pattern by traversing a path dependent on P and leaving robot at P's positions, the enumeration being used to remember the step where it currently is. In contrast to *TuringMobile* [55] that encodes arbitrary large numbers with three robots by allowing infinitesimally small differences between their distances, the *drawing formation* places robots in a grid with distance $\Omega(\sqrt{n}^{-1})$ between robots. DRAWING-PROTOCOL has a viewing range of 1 and can only form patterns where the unit-disc-graph is connected. This is a natural limitation for robots with limited visibility. However, our technique can be adapted to form disconnected patterns, as long as they contain a connected component of size ≥ 3. This work was first published in

Forming Large Patterns with Local Robots in the OBLOT Model [39]
Christopher Hahn, Jonas Harbig and Peter Kling
Symposium on Algorithmic Foundations of Dynamic Networks (SAND) 2024

1.2.4 APF for Widespread Swarms (Combined Main Result)

Combining WAVE-PROTOCOL and DRAWING-PROTOCOL leads to an ARBITRARY-PATTERN-FORMATION protocol that works for all initial configurations that have a convex Connectivity-Boundary, no 1-hole and which fulfills the symmetry condition from Theorem 1. As shown above, robots are not always able to distinguish locally whether the pattern is already formed or not (see Observation 1.1). This is solved by introducing one bit of persistent memory in each robot that is used to remember whether it is currently executing WAVE-PROTOCOL or DRAWING-PROTOCOL. Initially all bits are set on 'executing WAVE-PROTOCOL'. WAVE-PROTOCOL is a NEAR-GATHERING protocol that ends with simultaneous termination. At this moment each robot flips its bit; it now indicates 'executing DRAWING-PROTOCOL'. The combination of Theorems 7 and 8 immediately yields following main theorem.

Theorem 9 — Main Theorem. We assume robots according the $\mathcal{O}$BLOT model with the exception of a single bit of persistent memory. We further assume the $\mathcal{F}$SYNC scheduler, a viewing range of $2+\sqrt{2}$ and a configuration $\mathbf{z}$ with a convex Connectivity-Boundary and no 1-holes. A connected pattern P can be formed by $|P|$ robots in configuration $\mathbf{z}$ if and only if $\mathrm{sym}(\mathbf{z}) \mid \mathrm{sym}(P)$.

1.3 Model

In the following, at first the $\mathcal{O}$BLOT model is defined. Afterwards the schedulers ($\mathcal{F}$SYNC, $\mathcal{S}$SYNC and $\mathcal{A}$SYNC) are defined. The $\mathcal{O}$BLOT model does not define, how robots are oriented or when they observe each other; this is described in two separate subsections below. In the end of the model section, the usage of determinism is depicted. At parts, multiple models and different model parameters are described. In the end of each subsection it is explicitly remarked what the concrete model or model parameter of this thesis is.

1.3.1 $\mathcal{O}$BLOT Model

The $\mathcal{O}$BLOT model is described in [31]. $\mathcal{O}$BLOT is an abbreviation of "$\mathcal{O}$BLivious roboT". *Robots* in this model are autonomously moving entities that act in discrete LOOK-COMPUTE-MOVE (LCM) cycles. An LCM-cycle is split into three phases.

LOOK. The robot observes its surrounding and stores it in a temporary snapshot $\mathcal{S}$. All information, e.g., the positions of other robots, is stored in a *local coordinate system* that is self-centered. Depending on the orientation capabilities of a robot, the rotation, scaling, or mirroring of the local coordinate system may vary between robots and from cycle to cycle, but it is consistent until the end the current LCM-cycle. The snapshot includes all information a robot has about itself, other robots, or its environment and task (e.g. the pattern).
COMPUTE. The robot uses a predefined protocol $\mathcal{P}$ to compute an action based solely on the snapshot $\mathcal{S}$. An action is usually a movement but can, dependent on the robot's capabilities, also be the storage of information in persistent memory or the communication of a message. A movement is computed as the *target position* in the local coordinate system of the snapshot. The target position is defined as the function $\mathrm{target}^{\mathcal{P}}(\mathcal{S})$ dependent on the protocol $\mathcal{P}$ and the snapshot $\mathcal{S}$. Usually, a robot is only allowed to move a limited distance (e.g., 1) in one cycle. The target position must be within this limit.
MOVE. The robot executes the computed action and moves to $\mathrm{target}^{\mathcal{P}}(\mathcal{S})$. If the movement is *rigid*, the target position is reached exactly. Otherwise, the robot moves on a straight line towards the target position but may be stopped by an adversary at an arbitrary point. Afterwards, the snapshot and the knowledge of the target position is deleted.

The $\mathcal{O}$BLOT model defines that the robots are homogeneous, anonymous, identical, silent, and oblivious.

homogenieous. All robots perform the same predefined protocol $\mathcal{P}$. *identical*. A robot has no way to distinguish the robots it observes during LOOK beside their relative position (and possibly messages). Even if it can store the snapshot from a past cycle in persistent memory, it cannot reliably map the robots from the past to the current snapshot. *anonymous*. Robots do not have a unique identifier (or similar attribute) to solve leader election (or similar problems) with internal information. *silent*. Robots do not communicate. No messages are sent during MOVE or received during LOOK. *oblivious*. Robots do not have persistent memory. They forget everything after MOVE.

From these properties it follows that the snapshot made during LOOK is equivalent to a set of positions representing the observed robots.

Some results in this thesis use a slightly changed model that considers robots with persistent memory. The related work also considers models where robots can communicate. The adaption of the $\mathcal{O}$BLOT model is defined as follows. During the COMPUTE phase a robot decides, what information I is to be stored in persistent memory or what message M is to be communicated. In the MOVE phase, I is stored and/or M is communicated. The next snapshot in LOOK contains the information I stored by the robot in the last MOVE and the message each observed robot communicated during its respective last MOVE phases. When the stored information is also communicated (therefore $I = M$) and has only a constant size (i.e., can be modeled with constant many states), the model is call $\mathcal{L}$UMI [31] (for "luminous" because it can be modeled with a light that can shine in different colors).

R In this thesis, the $\mathcal{O}$BLOT model is assumed. Only Theorem 9 assumes one bit of persistent memory in addition. The robots' movement is rigid and limited by distance 1 per cycle.

1.3.2 Scheduler

[31] defines three schedulers that specify how the LCM-cycles of different robots are synchronized. The fully-synchronous $\mathcal{F}$SYNC scheduler is the strictest as all phases have to be synchronized. All robots must have finished their LOOK phase before the first robot can start its COMPUTE phase, and so on. Because no robot is observed during MOVE, the movement can be assumed to be instant. The cycles executed since initialization are counted as *rounds*.

The semi-synchronous $\mathcal{S}$SYNC scheduler loosens these restrictions. Robots can pause entire rounds. Those robots are called *inactive* and the others are called *active*. The phases of the active robots must be synchronized as defined for $\mathcal{F}$SYNC. An *epoch* is the time frame in which each robot was active at least once.

The asynchronous $\mathcal{A}$SYNC scheduler does not give restrictions on the timing of cycles or phases. Each phase can take an arbitrary (but final) amount of time with arbitrary pauses in between. During movement, a robot can pause for an arbitrary period of time. Similarly to $\mathcal{S}$SYNC, an epoch is the time frame where each robot executed at least one LCM-cycle. All three schedulers guarantee that the duration of a round/epoch is final.

R The results in this thesis mostly assume $\mathcal{F}$SYNC; some results in Chapter 2 are stated for the more general $\mathcal{S}$SYNC scheduler.

1.3.3 Orientation

Robots are placed in Euclidean hyperspace (usually only two- or three-dimensional). As described above, a robot observes its surrounding in a self-centered coordinate-system during LOOK. It is possible to have multiple degrees of agreement between local coordinate-systems of different robots possible. A *compass* is an agreement of a common direction of the X-axis. If robots have no compass we call them *disoriented*. Disoriented robots each have an arbitrarily rotated local-coordinate system. If robots agree on *chirality* they have a common understanding of left and right; otherwise, the coordinate-system can be arbitrarily mirrored.

When considering the protocol executed locally on a disoriented robot, the protocol can make use of the local orientation of the robot[3]. However, most literature considering those models designs protocols invariant under the local orientation.

R This thesis assumes robots in the Euclidean plane. They are disoriented, but agree on a unit-distance. For Chapter 2 an agreement on chirality is not necessary while the other results assume a common chirality. The protocols in this thesis are invariant under the local orientation.

1.3.4 Visibility and Connectivity

$\mathcal{O}$BLOT does not specify what robots can be observed during LOOK. Some results (e.g. [69]) consider a *global* visibility where all robots can mutually observe each other. Certain problems like GATHERING become trivial[4] with this assumption. Other results (e.g. [72]) consider *limited visibility* where robots can only observe other robots within a defined *visibility range*. Robots with limited visibility are sometimes called *local*.

With limited visibility, the question of *connectivity* is raised. If there is no transitive visibility between a set of robots, one can hardly assume a unified swarm behavior. Most of the results in this thesis distinguish between visibility and connectivity; for this case, a second distance called *connectivity range* is introduced. A swarm is considered to be *connected* if the *unit disc graph (UDG)*, where the unit distance is the connectivity-range is connected; it is called *connectivity-graph*. We call robots that are connected *neighbors*. We define the *boundary* of the connectivity-graph (called Connectivity-Boundary) as follows.

Definition 1.6 — Connectivity-Boundary. The connectivity-graph is a graph in the Euclidean plane. Initially, we define the whole plane as "outside". Every cycle in the connectivity graph surrounds an area, we define the union of these areas as "inside". The Connectivity-Boundary consists of all nodes (robots) that are adjacent to "outside". In the Connectivity-Boundary all induced edges remain that are adjacent to "outside" and do not cross other nodes.

The Connectivity-Boundary is a graph with exactly one Euler cycle. If the Connectivity-Boundary is convex, the Euler cycle is a Hamilton cycle as well. Sometimes it is referred to neighbors of a robot in a convex Connectivity-Boundary. This always means the two neighbors in the Hamilton cycle.

A swarm is called *well-connected* if a connectivity range is assumed that is smaller than the visibility range by a constant (see Definition 1.5).

R In this thesis, the distances are always normalized so that the connectivity-range is 1. Chapter 2 assumes a well-connected swarm, i.e. the viewing range is $1+\sigma$ for an arbitrary constant $\sigma > 0$. In Chapter 3 the viewing range is $2+\sqrt{2}$ for Theorem 7 and 1 for the other results. In Chapter 4, a viewing rage of 1 is assumed.

1.3.5 Determinism

Access to a true source of randomness is a powerful capability for distributed robot swarms. Most problems considered in this thesis become trivial or much simpler with

[3]An example would be the protocol where robots move distance 1 in the direction of their local X-axis
[4]All robots can move to the center of the swarms smallest enclosing circle.

randomness. For example, Near-Gathering can be reduced to Gathering. If the Gathering is finished each robot can use the randomness to move a very small random distance in a random direction. With high probability, this solves Near-Gathering in a few extra steps. Similarly, any configuration with symmetricity > 1 can reduce its symmetricity to 1 by perturbing the position of each robot. This removes the necessity of symmetry preservation and even allows forming patterns that do not satisfy the symmetry condition of Theorem 1. However, the exact analysis of swarm formation problems considers randomness to be controlled by an adversary. This allows to present solutions to problems that always works (and not only with high probability) but limits the advantages of randomness severely. Much of the related work incorporates this into the model by allowing only deterministic protocols (e.g. [24, 68, 69, 72]).

R This thesis considers deterministic protocols.

1.4 Notation

In the following, special notations used in this thesis are described.

1.4.1 Configuration

Usually, the number of swarm robots is denoted by $n \in \mathbb{N}$. Let $\{r_1, \cdots, r_n\}$ be the set of robots in the swarm (the ordering is arbitrary and is not known by the robots). We denote the positions of robots in an arbitrary global coordinate system. $\text{pos}_t(r_i) = z_i^t = (x_i^t, y_1 i^t) \in \mathbb{R}^2$ is defined to be the position of r_i. The *configuration* $\mathbf{z}^t = (z_i^t)_{i=1}^n \in (\mathbb{R}^2)^n \equiv \mathbb{R}^{2n}$ describes the positions of all robots in round t. $\mathbf{z}^0$ is also called *initial configuration*. The notation $\mathbf{z}$ is sometimes abused to represent the set of points in the configuration rather than a vector, e.g. $p \in \mathbf{z}$.

1.4.2 Target Position

As described above, a robot takes a snapshot $\mathcal{S}$ during the Look phase and computes the *target-position* $\text{target}^{\mathcal{P}}(\mathcal{S})$ during Compute. Because the model specified for this thesis (identical and disoriented robots) contains no additional information to the relative positions in the snapshot, it is equivalent to a translated and rotated subvector of the configuration $\mathbf{z}^t$. Because the protocol is invariant under the local orientation, the rotation of the subvector can be ignored. To simplify the notation, $\text{target}^{\mathcal{P}}(p;\mathbf{z})$ is used instead to describe the target position of a robot observing the configuration $\mathbf{z}$ from the position p. For robot r_i in round t, this is $\text{target}^{\mathcal{P}}(z_i^t, \mathbf{z}^t)$ and this case is usually abbreviated further to $\text{target}_i^{\mathcal{P}}(t)$.

1.4.3 Characterizing Protocols via their Evolution Function

Chapter 3 uses methods from *dynamical systems*. There, the evolution of the entire configuration is considered rather than the movement of individual robots. The evolution of a configuration can be described as

$$\mathbf{z}^{t+1} = F^{\mathcal{P}}(\mathbf{z}^t) = \begin{pmatrix} \text{target}^{\mathcal{P}}(z_1^t;\mathbf{z}^t) \\ \vdots \\ \text{target}^{\mathcal{P}}(z_n^t;\mathbf{z}^t) \end{pmatrix}. \tag{1.1}$$

$F^{\mathcal{P}}$ is called the *evolution function* of the protocol $\mathcal{P}$. In the context of the evolution function, $\mathcal{P}$ is usually omitted and $f(z_i^t;\mathbf{z}^t)$ used instead of $\text{target}^{\mathcal{P}}(z_i^t;\mathbf{z}^t)$. This leads to the notation

$$\mathbf{z}^{t+1} = F(\mathbf{z}^t) = \begin{pmatrix} f(z_1^t;\mathbf{z}^t) \\ \vdots \\ f(z_n^t;\mathbf{z}^t) \end{pmatrix}. \tag{1.2}$$

Whenever the specific value of t is irrelevant (e.g., when an arbitrary round is investigated) the superscript is dropped and $z_i^+ = f(z_i;\mathbf{z})$ abbreviated where $z_i^+ \in \mathbb{R}^2$ indicates the “next” position of robot i. Similarly, the notation $\mathbf{z}^+ = F(\mathbf{z})$ is used to indicate the “next” configuration if the evolution function F is applied to some configuration $\mathbf{z}$.

1.4.4 Pattern

Remember that the ARBITRARY-PATTERN-FORMATION problem is considered for the main result (Theorem 9) as well as the results in Chapter 4: a swarm of n robots must form a target pattern $P \subseteq \mathbb{R}^2$ of $|P| = n$ coordinates. Since robots are oblivious, the standard assumption is used that, each round, they receive P as part of the snapshot during LOOK in an arbitrary but fixed coordinate system (i.e., robots receive the exact same numerical values). Without loss of generality, it is assumed that P’s smallest enclosing circle is centered at the origin (otherwise robots translate P accordingly). P’s symmetry is measured via its *symmetricity* (see Definition 1.2).

Let P be a pattern with symmetricity s. The $n = |P|$ positions can be split into n/s regular s-gons with the same center. We call a set that contains exactly one position from each s-gon a *symmetric-component* of P.

1.4.5 Geometrical Notations

A configuration $\mathbf{z}$ has several geometric properties that are relevant in multiple definitions. The convex hull of $\mathbf{z}$ is referred to as *global convex hull*, the diameter of $\mathbf{z}$ is the *global diameter*. The convex hull and diameter of the subvector of $\mathbf{z}$ observed by the robot r_i is referred to as the *local convex hull* and *local diameter* of r_i. This works analogous for the *global/local smallest enclosing circle* (usually abbreviated with *SEC*). $\text{diam}(t)$, respectively $\text{hull}(t)$, is defined as the global diameter, respectively convex hull, in round t. $\text{diam}_i(t)$, respectively $\text{hull}_i(t)$, is the local diameter, respectively convex hull, of r_i in round t. The unit-distc-graph for the configuration in round t is denoted with $\text{UDG}(t)$.

We define a hole inside a configuration as follows.

Definition 1.7 — δ-hole. A *δ-hole* of a configuration is a circular area inside the Connectivity-Boundary with a diameter of δ that contains no robot.

For two points $p,q \in \mathbb{R}^2$ we define $\text{dist}(p,q) = \|p-q\|_2$ as their Euclidean distance. We extend this notation in the natural way to sets $S \subseteq \mathbb{R}^2$, such that, e.g., $\text{dist}(p,S) = \min\{\,dist(p,q) \mid q \in S\,\}$. We use a set-like notation for sequences $S = (s_i)_{i=1}^n$, like $p_1 \in S$, $S \subseteq \mathbb{R}^2$, or $\text{dist}(p,S)$. The minimal distance between two points in a set (or sequence) $S \subseteq \mathbb{R}^2$ is $\text{mindist}(S) := \min\{\,\text{dist}(p,q) \mid p,q \in S, p \neq q\,\}$. For $p \in \mathbb{R}^2$ and $r \in \mathbb{R}$ the set $\mathcal{B}(p,r) = \{\,q \in \mathbb{R}^2 \mid \text{dist}(p,q) < r\,\}$ denotes the open ball around p with radius r. For a set $S \subseteq \mathbb{R}^2$ we write its power set as $\text{Pow}(S)$, its closure as $\overline{S}$, and its boundary as $\partial S = \overline{S} \cap \overline{\mathbb{R}^2 \setminus S}$.

1.5 Further Related Work

The section is divided into a section on GATHERING, a section on ARBITRARY-PATTERN-FORMATION and related work on dynamic systems. See [70] for a survey on pattern formation. A more recent and general overview of results and open problems in swarm robotics and related areas can be found in [30].

1.5.1 Gathering and Convergence

GATHERING is the formation of the specific pattern “point”. A highly researched gathering protocol is GO-TO-THE-CENTER that is already mentioned above while motivating the NEAR-GATHERING results. It assumes the $\mathcal{O}$BLOT model for disoriented robots with a limited viewing range and was first introduced in [2]. The running time of $\Theta(n+\Delta^2)$ rounds until a gathering is reached is shown in [6, 24] (n is the number of robots, Δ the largest distance between two robots). [6, 7] additionally adapt GTC for the high-dimensional Euclidean space. There exists no known protocol with an asymptomatic running time faster than GO-TO-THE-CENTER in the same model. It is conjectured that the running time of GTC is worst-case optimal in the model [6, 7, 11]. With slight changes of the model, a faster running time is reached up to the trivial lower bound of $\Omega(\Delta)$. [59] considers a one-axis agreement (robots have a common compass but not chirality) and provides a gathering protocol in $\mathcal{O}(\Delta)$.

There exist comparable results of gathering for robots in the two-dimensional grid ($\mathbb{N}^2$). A gathering is possible in $\mathcal{O}(n^2)$ rounds for oblivious, disoriented robots with limited visibility [8]. An improvement of the running time to $\mathcal{O}(n)$ can be achieved by giving disoriented robots a constant-sized memory [20].

In intense research was conducted on what assumptions and initial configuration gathering are possible or impossible to solve. Especially the impossibility results are relevant for this thesis, because they can directly be translated to the ARBITRARY-PATTERN-FORMATION problem. In the following, robots with global visibility are assumed. It is shown that oblivious, disoriented robots without multiplicity detection can solve the gathering under $\mathcal{F}$SYNC [18] but not under $\mathcal{S}$SYNC or $\mathcal{A}$SYNC [60]. With multiplicity detection (robots can observe whether a position is occupied by 1 or multiple robots), a swarm can be gathered when it contains at least 3 robots, even under the weaker assumption of the $\mathcal{A}$SYNC scheduler [17]. However, a swarm with only 2 robots cannot be gathered under the same assumption [64]. Similarly to the case with limited visibility, a one-axis agreement is also an advantage with global visibility as well; it enables gathering in general under $\mathcal{A}$SYNC scheduler [3].

CONVERGENCE is a problem related to GATHERING where robots reach a configuration of arbitrary small diameter. In contrast to gathering, CONVERGENCE does not require multiplicity detection for disoriented oblivious robots with global visibility under $\mathcal{A}$SYNC [18]. [19] showed that convergence is possible in $\mathcal{O}(n \cdot \log \Delta/\delta)$ epochs (Δ is the diameter of the initial configuration, δ the desired diameter of the final configuration). With limited visibility and otherwise, the same model under $\mathcal{S}$SYNC [2] provides a protocol that solves CONVERGENCE. Under $\mathcal{A}$SYNC scheduler, convergence is not possible with limited visibility [47]. However, for $O(1)$-bounded asynchronicity (i.e. the k-$\mathcal{A}$SYNC scheduler where one robot is activated at most k times during the LCM-cycle of another robot) CONVERGENCE is possible with limited visibility and multiplicity detection [46, 47].

GATHERING and CONVERGENCE is considered for connected chains of robots in the

Euclidean plane as well. A chain connects each robot to a right and left neighbor, the ends of the chain can have only a right or left neighbor (in a closed chain the relation is circular). Visibility is measured in distance along the chain, not in Euclidean distance. An often considered protocol is GO-TO-THE-MIDDLE which solves convergence [27]. It has a running time of $\mathcal{O}(n^2 \cdot \log \Delta/\delta)$ to reach a configuration of diameter δ [18]. Similarly to the grid, the running time is much faster ($\mathcal{O}(n)$) when robots have a constant memory [12].

There exist also research that considers GATHERING and CONVERGENCE with more uncertainty in the behavior of robots, a survey can be found in [22]. On one hand, the swarm may include a few robots that crash (remain stationary from this time on) or are Byzantine (operates an adversarial protocol), on the other hand, the measurement or movement may be inaccurate.

1.5.2 Arbitrary Pattern Formation

The ARBITRARY-PATTERN-FORMATION problem has been considered in numerous settings and variants. The works closely related to this thesis [69, 72] are already described in Section 1.1. They consider disoriented robots according to the $\mathcal{O}$BLOT model under $\mathcal{F}$SYNC scheduler. A central result there is the general limitation which initial configuration can result in which patterns based on the symmetricity (Theorem 1). [33] continues these results by showing under what conditions the formation of *any pattern* (no matter the symmetricity) with global visibility is possible. They prove that an agreement on both axes is sufficient to from any pattern and with a one-axis agreement (common compass, no chirality) any pattern with an odd number of positions can be formed. [16, 25, 34] researched APF under the $\mathcal{A}$SYNC scheduler for disoriented robots without common chirality. They proved that there the ARBITRARY-PATTERN-FORMATION is equivalent to LEADER-ELECTION, i.e. APF is solvable with initial configuration $\mathbf{z}_0$ if and only if the robots in $\mathbf{z}_0$ can agree on one robot as a leader. This is equivalent with a symmetricity of 1 in $\mathbf{z}_0$ in the model used for this thesis. Under the *sequential scheduler* (robots are activated in a Round-Robin fashion) APF is solvable for all initial configuration unrestricted by the symmetry condition (Theorem 1) [28].

[64] shows a restriction similar to Theorem 1 for robots with global visibility and arbitrarily large memory. The memory enables robots to explore the local orientation of all robots, therefore, the symmetricity definition is adjusted to include the local orientations of all robots (i.e. a robot is represented by a unit vector instead of a point in Definition 1.2). [4] considers pattern formation for oblivious robots with unlimited visibility on an infinite grid, showing that an initially asymmetric swarm can form any pattern. The authors of [5] assume unlimited but obstructed visibility (i.e., robots can obstruct each others view) with partial axis agreement and luminous robots (that can communicate via a constant number of lights). In [21], the authors considered when oblivious robots with unlimited visibility can form (cyclic) sequences of patterns. Pattern formation for robots in three-dimensional space is considered in [71].

ARBITRARY-PATTERN-FORMATION designs an algorithm that can form any at runtime given pattern. A protocol can be designed much more dedicated if the pattern is known in advance. One example of such a *specific* pattern formation is the GATHERING problem above. Other patterns often considered are “circle” and “line”. The CIRCLE-FORMATION problem (forming a n-gon of given radius) is solved for disoriented $\mathcal{O}$BLOT under $\mathcal{A}$SYNC with global visibility and non-rigid movement [32, 56, 68]. The MAX-LINE-FORMATION problem (forming a line with distance 1 between robots) is solved for $\mathcal{O}$BLOT robots with limited visibility and one-axis agreement [9]. However, robots must have a *square* connectivity and viewing range (robots are connected/observed within a square of a given side length; the square is aligned with the common axis direction), otherwise MAX-LINE-FORMATION is in general impossible. The problem of forming a line is also studied for connected chains of robots, there it is called (MAX-)CHAIN-FORMATION. The GO-TO-THE-MIDDLE protocol solves it [27] by reaching the shortest line between two static endpoints of the chain. [13] presents a protocol that reaches a long line by maximizing the distances between the two endpoints of the chain.

There exists also research on pattern-formation considering robots with identifiers. Each robot has a color, each group of robots with the same color has to form the pattern. This problem is solved for GATHERING and CIRCLE-FORMATION [29, 62].

1.5.3 Dynamical System

In dynamical systems theory, much of the related literature considers *consensus* or *synchronization* problems that share characteristics with GATHERING or NEAR-GATHERING, e.g., by identifying robot positions with “opinions” the agents want to find a consensus on. A good overview over this research branch can be found in [58] or in the slightly more recent surveys [14, 26]. Extensions of these methods to pattern formation have been proposed as well (see e.g., [1]). However, in this area the focus is typically on time-continuous systems — in the form of differential equations — and the models are not as strictly restricted as necessary in our context, e.g., allowing global communication range.

There are several results in the field of swarm robots applying techniques from dynamical systems theory. [49, 50] use it, to prove that a swarm of robots can form a grid. [35] analyzes the convergence speed of different circular structures on the basis of dynamical systems theory.

2. A Class of Near-Gathering Strategies

The contribution of this chapter is to provide a class of protocols that solves NEAR-GATHERING in $\mathcal{O}(\Delta^2)$ rounds, where the diameter Δ denotes the initial maximal distance between two robots. It is based on a class called *λ-contracting protocols*. Such protocols restrict the allowed target points to a specific subset of a robot's local convex hull (formed by the positions of all visible robots, including itself) in the following way. Let *diam* denote the diameter of a robot's local convex hull. Then, a target point p is an allowed target point if it is the center of a line segment of length $\lambda \cdot \mathit{diam}$, completely contained in the local convex hull. This guarantees that the target point lies far enough inside the local convex hull (at least along one dimension) to decrease the swarm's diameter sufficiently. See Figure 2.1 for an illustration.

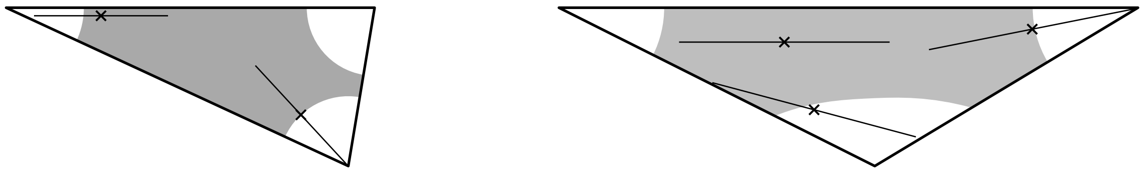


Figure 2.1: Two local convex hulls, each formed by 3 robots. The gray area marks valid target points of λ-contracting protocols. The exemplary line segments all have length $\lambda \cdot \mathit{diam}$, where *diam* is the diameter of the respecting convex hull. On the left $\lambda = 4/7$, on the right $\lambda = 4/11$.

We believe these λ-contracting protocols encapsulate the core property of fast CONVERGENCE protocols. Note that λ-contracting protocols do neither lead directly to GATHERING, which requires robots to eventually reach the same position, nor to NEAR-GATHERING, which requires simultaneous termination and collision avoidance.

In this chapter, the class of λ-contracting NEAR-GATHERING protocols is presented that solves NEAR-GATHERING. Naturally, those protocols must ensure collision-freeness. As in previous work on this problem [57], our protocols require that the initial swarm is *well-connected*, that is, that the swarm is connected with respect to *connectivity range* of 1 and the robots have *viewing range* of $1+\tau$, for the constant $\tau > 0$ (Definition 1.5). The protocols ensure that the swarm stays connected concerning the connectivity range. These are, to our knowledge, the first NEAR-GATHERING protocols for oblivious disoriented robots with limited visibility.

Well-connectedness serves two purposes. First, it allows a robot to compute its target point under the given λ-contracting protocol and the target points of nearby robots to prevent collisions. Its second purpose is to enable termination. Once there is a robot whose local convex hull has a diameter at most τ, *all* robots must have a distance at most τ, as otherwise the swarm would not be connected in terms of the connectivity range 1. Thus, all robots can *simultaneously* decide (in the same round in $\mathcal{F}$SYNC) whether NEAR-GATHERING is solved. If the swarm is not well-connected, it is easy to see that such a simultaneous decision is impossible. Consider a protocol that solves NEAR-GATHERING for a swarm of two robots and terminates in the $\mathcal{F}$SYNC model. Fix the last round before termination and add a new robot visible to only one robot (the resulting swarm is connected). One of the original two robots still sees the same situation as before and will terminate, although NEAR-GATHERING is not solved.

Observation 2.1 NEAR-GATHERING with simultanous termination is not possible in a swarm of oblivious robots with limited visibility if the swarm is not well-connected.

Simultaneous termination also allows us to derive the first protocol to solve UNIFORM-CIRCLE-FORMATION for disoriented robots with limited visibility as well as GATHERING without early collisions. Once the robots' local diameter (and hence also the global diameter) is less than τ, they essentially have a global view. As the UNIFORM CIRCLE protocol from [32] maintains a small diameter, it can be used after the termination of our NEAR-GATHERING protocol without any modification. For GATHERING all robots simply move in one step onto the center of the SEC (any deterministically computable point invariant of rotation and translation inside the convex hull is sufficient). Because the protocol is otherwise collision-free, collisions only occur in this last step.

ARBITRARY-PATTERN-FORMATION cannot be solved directly after the NEAR-GATHERING, even though there are protocols that consider a similar model at first glance ([69] and Chapter 4). These protocols restrict the set of formable patterns to have rotational symmetry similar to the initial configuration, i.e., the patterns *symmetricity* (Definition 1.2) must be an integer multiple of the symmetricity of the initial configuration. However, the class of protocols presented here does not guarantee symmetry preservation. Therefore, we can only make trivial assumptions about which patterns are formable by which initial configuration *before* the NEAR-GATHERING. Symmetricity always divides the number of robots n. Therefore, patterns with symmetricity n can always be formed. The only two patterns that match this definition have already been mentioned above: circle and point. In Chapter 3, a method is presented for preserving the symmetricity.

The results of this chapter were first published in

A Unifying Approach to Efficient (Near)-Gathering of Disoriented Robots with Limited Visibility [10]
Jannik Castenow, Jonas Harbig, Daniel Jung, Peter Kling, Till Knollmann and Friedhelm Meyer auf der Heide
Conference on Priciples of Distributed Systems (OPODIS) 2022

Joint Work. The design of λ-contracting protocols (Sections 2.1 and 2.2) was a joint work and is also published in the doctoral thesis of Jannik Castenow [7]. Castenow focused on the GATHERING problem and developed a class of λ-contracting gathering protocols while this thesis presents the related λ-contracting NEAR-GATHERING protocols with results on NEAR-GATHERING. The correctness and running proofs for both protocol classes are based on the same high-level idea that is presented in Section 2.2.1. However, they are different on a technical level because the proof for λ-contracting NEAR-GATHERING protocols utilizes that the swarm is *well-connected* (Definition 1.5). A λ-contracting gathering protocol does not require the swarm to be well-connected, instead the proof in [7] uses that the protocol is *collapsing*, i.e. robots move regularly onto the same position. The developed λ-contracting NEAR-GATHERING protocols (Section 2.3) and the related proofs (Section 2.4) are mainly written by the author of this thesis (Jonas Harbig) and not published in any other doctoral thesis.

Outline. First, the classes of λ-contracting and λ-contracting NEAR-GATHERING protocols are formally introduced (Section 2.1); a proof is given in Section 2.2 that λ-contracting NEAR-GATHERING protocols solve NEAR-GATHERING. In Section 2.3, it is presented how to create a λ-contracting NEAR-GATHERING protocol from any λ-contracting protocol; the proofs can be found in the next section (Section 2.4).

2.1 A Class of Contracting Protocols

The first intuition to define a class of protocols that contracts would be to transfer the class of continuous contracting protocols from [51] (cf. Section 1.5) to the discrete LCM case. Robots that are part of the global convex hull move with constant speed toward the inside or along the boundary of the global convex hull. A translation to the discrete (LCM) case might be to demand that each robot moves a constant distance inward (away from the boundary) of the global convex hull, cf. Figure 2.2.

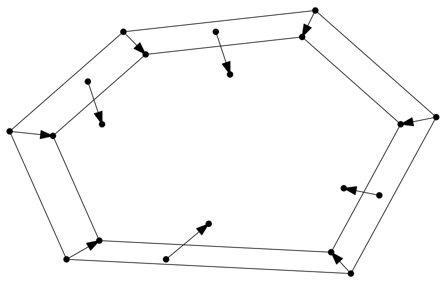


Figure 2.2: Ideally, every robot that is close to the boundary of the global convex hull would move a constant distance inwards.

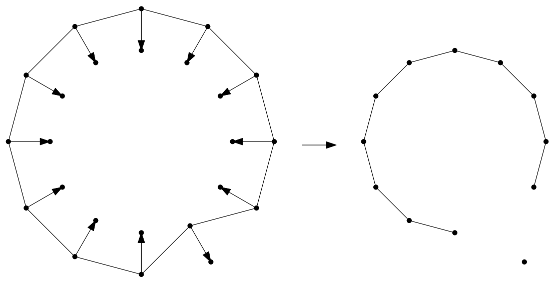


Figure 2.3: Visualization of the example to emphasize that continuous protocols cannot be directly translated to the LCM case.

However, such a protocol cannot exist in the discrete LCM setting. Consider n robots placed on the vertices of a regular polygon with side length 1. Now take one robot and mirror its position along the line segment connecting its two neighbors (cf. Figure 2.3). Next, we assume that all robots would move a constant distance along the angle bisector between their direct neighbors in the given gathering protocol. Other movements would lead to the same effect since the robots are disoriented. In the given configuration, $n-1$ robots would move a constant distance inside the global convex hull while one robot even leaves the global convex hull. Not only does the global convex hull not decrease as desired, but also the connectivity of the UDG is not maintained as the robot moving outside loses connectivity to its direct neighbors. Consequently, discrete gathering protocols have to move the robots more carefully to maintain connectivity.

2.1.1 λ-contracting Protocols

Initially, we emphasize two core features of the protocols. A protocol is *connectivity preserving* if it always maintains the connectivity of $\mathrm{UDG}(t)$. Due to the limited visibility and disorientation, every protocol to solve NEAR-GATHERING must be connectivity preserving since it is deterministically impossible to reconnect lost robots to the remaining swarm. Moreover, we study protocols that are *invariant*, i.e., the movement of a robot does not change no matter how its local coordinate system is oriented. This is a natural assumption since the robots have variable disorientation and thus cannot rely on their local coordinate system to synchronize their movement with nearby robots. Moreover, many known protocols under the given robot capabilities are invariant, e.g., [2, 6, 53, 54].

Definition 2.1 — λ-Centered. Let Q be a convex polygon with diameter *diam* and $0 < \lambda \leq 1$ a constant. A point $p \in Q$ is called to be $\underline{\lambda\text{-centered}}$ if it is the midpoint of a line segment that is completely contained in Q and has a length of $\lambda \cdot diam$.

Definition 2.2 — λ-Contracting. A connectivity preserving and invariant robot formation protocol $\mathcal{P}$ is called $\underline{\lambda\text{-contracting}}$ if $\mathrm{target}_i^{\mathcal{P}}(t)$ is a λ-centered point of $\mathrm{hull}_i(t)$ for every robot r_i and every $t \in \mathbb{N}_0$.

Two examples of λ-centered points are depicted in Figure 2.1. Observe that Definition 2.2 does not necessarily enforce a final GATHERING or NEAR-GATHERING of the protocols. Consider, for instance, two robots. A protocol that demands the two robots to move halfway towards the midpoint between themselves would be $1/4$-contracting, but the robots would only *converge* towards the same position. However, for GATHERING, the robots must compute the same target point eventually. That robots converge (even without collisions) does also not lead directly to NEAR-GATHERING, because this requires robots to *simultaneously terminate*. Without memory or communication, a swarm must be *well-connected* (Definition 1.5) to do that, i.e., let the viewing range be $1+\tau$ for a constant $\tau > 0$ and let UDG be connected. Consider a protocol that solves NEAR-GATHERING for a swarm of two robots and terminates in the $\mathcal{F}$SYNC model. Fix the last round before termination and add a new robot visible to only one robot (the resulting swarm is not connected). One of the original two robots still sees the same situation as before and will terminate, although NEAR-GATHERING is not solved. We state this in the following observation.

Observation 2.2 Any swarm of robots according to the $\mathcal{O}$BLOT model must be *well-connected* to solve NEAR-GATHERING.

Additionally, a NEAR-GATHERING protocol must avoid collisions. This leads to the following class of protocols.

Definition 2.3 — λ-Contracting Near-Gathering Protocol. A robot formation protocol $\mathcal{P}$ is a $\underline{\lambda\text{-contracting near-gathering protocol}}$ if $\mathcal{P}$ is λ-contracting, *collision-free* and keeps the swarm *well-connected*.

2.2 Analysis of λ-contracting Near-Gathering Protocols

We state the following upper bound for λ-contracting NEAR-GATHERING protocols.

Theorem 2 — restated. Consider a swarm of robots in $\mathbb{R}^2$ with diameter Δ that is *well-connected* Every λ-contracting NEAR-GATHERING protocol solves NEAR-GATHERING after $\mathcal{O}\left(\Delta^2\right)$ rounds.

Note, that the protocol also works under $\mathcal{S}$SYNC with the same asymptotic running time, see Observation 2.3.

The running time proof for the upper bound can be found in the following sections and is split in a high-level description and a detailed analysis, that contains the proofs.

The following matching lower bound for λ-contracting protocols can be shown. The proof of the lower bound is, in most parts, identical to the lower bound of the GTC protocol [24].

Theorem 10 For a swarm of n robots in $\mathbb{R}^2$ with a viewing range of $2-\delta$ there exists a connected configuration $\mathbf{z}^0$ such that every λ-contracting protocol $\mathcal{P}$ requires $\Omega\left(\Delta^2\right)$ rounds to reach a Diam-1-Configuration if δ is constant.

Proof. A worst case configuration to show the theorem is a regular n-gon with side length 1. Because the robots are disoriented and the protocols deterministic, the robots can be rotated in a way that the configuration always remains a regular n-gon. With a viewing range of $2-\delta$, a robots only see its two direct neighbor on the n-gon as long the side-length $> 1-\delta/2$. Because the protocols is λ-contracting a robot must stay inside the triangle of itself and its two neighbors. Therefore, the radius of the n-gon can shrink by at most height of this triangle. Analogous to the proof of Lemma 3.18 can be shown, that it needs at least $\Omega(\Delta^2)$ rounds such that the side length of the n-gon becomes $\leq 1-\delta/2$ and a robot can see more neighbors. ■

Gathering without early collision. This chapter is about NEAR-GATHERING, which is in some way the opposite of GATHERING because the latter requires collisions. A stronger variant of the GATHERING problem considers it without early collision, i.e. only in the very last step do all robots move onto the same position while having no collisions beforehand. Our λ-contracting NEAR-GATHERING protocol can easily be adapted to solve this problem in a fully-synchronous setting. NEAR-GATHERING requires that the robots terminate simultaneously in the end. At this moment, each robot can observe the whole swarm, and no collision has happened. Instead of termination, the robots move onto the SEC center.

Theorem 3 — restated. Consider a swarm of robots in $\mathbb{R}^2$ with diameter Δ that is *well-connected* Every λ-contracting NEAR-GATHERING protocol solves GATHERING-WITHOUT-EARLY-COLLISION in $\mathcal{O}\left(\Delta^2\right)$ fully-synchronous rounds.

2.2.1 High-Level Description of Running Time Proof

The proof is inspired by the proof of the GTC protocol [24]. The proof aims to show that the radius of the global smallest enclosing circle (SEC), i.e., the SEC that encloses all robots' positions in a global coordinate system, decreases by $\Omega\left(1/\Delta\right)$ every two rounds. Since the initial radius is upper bounded by Δ, the runtime of $\mathcal{O}\left(\Delta^2\right)$ follows. See Figure 2.4 for a visualization.

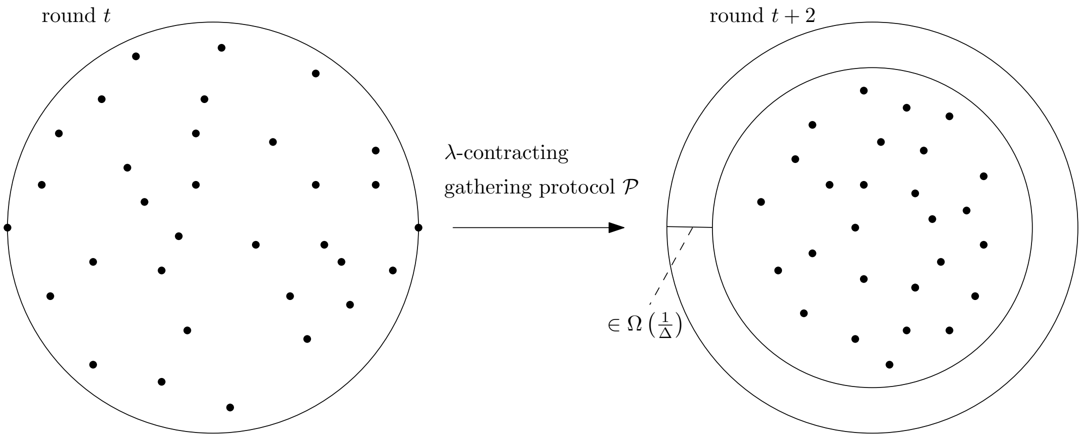

round t
round $t+2$
λ-contracting
gathering protocol $\mathcal{P}$
$\in \Omega\left(\frac{1}{\Delta}\right)$

Figure 2.4: We show that the radius of the global SEC decreases by $\Omega\left(1/\Delta\right)$ every round.

We consider the fixed circular segment S_λ of the global SEC and analyze how the inside robots behave. A circular segment is a region of a circle “cut off” by a chord. The circular segment $S_{\lambda\cdot\tau/2}$ has a chord length of at most $\lambda\cdot\tau$ (for a formal definition, see below) and we can prove a height h of $S_{\lambda\cdot\tau/2}$ in the order of $\Omega\left(1/\Delta\right)$ (Lemma 2.1). Observe that in any circular segment, the chord’s endpoints are the points that have a maximum distance within the circular segment, and hence, the maximum distance between any pair of points in $S_{\lambda\cdot\tau/2}$ is at most $\lambda\cdot\tau/2$. Recall that every robot r_i moves to the λ-centered point $\text{target}_i^{\mathcal{P}}(t)$. Moreover, $\text{target}_i^{\mathcal{P}}(t)$ is the midpoint of a line segment ℓ of length $\lambda\cdot\text{diam}_i(t)$ that is completely contained in the local convex hull of r_i. For robots with $\text{diam}_i(t)\geq\tau$, we see that ℓ is larger than $\lambda\cdot\tau$ and thus $\text{target}_i^{\mathcal{P}}(t)$ cannot lie inside $S_{\lambda\cdot\tau/2}$. From *well-connectedness* (Definition 1.5) follows that $\text{diam}_i(t)\geq\tau$ as long as NEAR-GATHERING is not finished (Lemma 2.2). Finally, the previous statements can be combined to give a lower bound on how much the radius of the global SEC decreases.

2.2.2 Detailed Analysis

First, we introduce some definitions. Let $GS := GS(t)$ be the (global) smallest enclosed circle of all robots in the round t and $R := R(t)$ be its radius. Now, fix any point b on the boundary of GS. The two points in distance $\lambda/2$ of b on the boundary of GS determine the circular segment S_λ with height h (analog distance $\tau\cdot\lambda/4$ for $S_{\lambda\cdot\tau/2}$). See Figure 2.5 for a depiction of the circular segments S_λ and $S_{\lambda\cdot\tau/2}$. In the following, all lemmata consider robots that move according to a λ-contracting NEAR-GATHERING protocol $\mathcal{P}$.

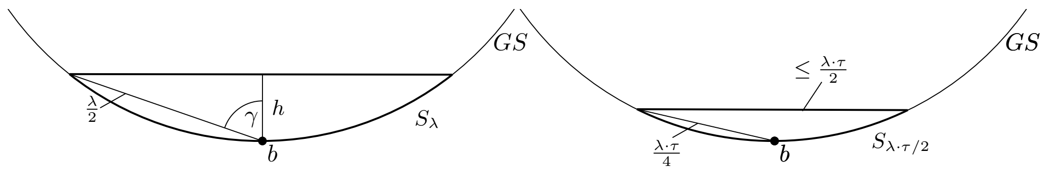


Figure 2.5: The circular segments S_λ (to the left) and $S_{\lambda\cdot\tau/2}$ of the global SEC GS are depicted.

In the following, we prove that all robots leave the circular segment $S_{\lambda\cdot\tau/2}$ every round. As a consequence, the radius of GS decreases by at least h. Initially, we give a bound on h. We use Jung's Theorem (Theorem 11) to obtain a bound on R and also on h.

Theorem 11 — Jung's Theorem [43, 44]. The smallest enclosing hypersphere of a point set $K \subset \mathbb{R}^d$ with diameter diam has a radius of at most diam $\cdot \sqrt{\frac{d}{2\cdot(d+1)}}$.

Lemma 2.1 The height of $S_{\lambda\cdot\tau/2}$ is $h \geq \frac{\sqrt{3}\lambda^2\tau^2}{64\pi\Delta}$.

Proof. In this proof, we show the height of S_λ, because this simplifies the equations. The respective height of $S_{\lambda\cdot\tau/2}$ can be followed analogously.

Initially, we give an upper bound on the angle γ; see Figure 2.5 for its definition. The circumference of GS is $2\pi R$. We can position at most $\frac{2}{\lambda}\pi R$ points on the boundary of GS that are at a distance $\frac{\lambda}{2}$ from the points closest to them and form a regular convex polygon. The internal angle of this regular polygon is 2γ. Hence, the sum of all internal angles is $\left(\frac{4}{\lambda}\pi R - 2\right)\cdot\pi$. Thus, each individual angle has a size of at most $\frac{\left(\frac{4}{\lambda}\pi R-2\right)\cdot\pi}{\frac{4}{\lambda}\pi R} = \pi - \frac{2\pi}{\frac{4}{\lambda}\pi R} = \pi - \frac{\lambda}{2R}$. Hence, $\gamma \leq \frac{\pi}{2} - \frac{\lambda}{4R}$. Now, we are able to bound h. First, we derive a relation between h and γ: $\cos(\gamma) = \frac{h}{\frac{\lambda}{2}} = \frac{2h}{\lambda} \iff h = \frac{\lambda\cdot\cos(\gamma)}{2}$. In the following upper bound, we make use of the fact that $\cos(x) \geq -\frac{2}{\pi}x + 1$ for $x \in [0, \frac{\pi}{2}]$.

$$h = \frac{\lambda\cdot\cos(\gamma)}{2} \geq \frac{\lambda\cdot\cos\left(\frac{\pi}{2}-\frac{\lambda}{4R}\right)}{2} \geq \frac{\lambda\cdot\left(-\frac{2}{\pi}\cdot\left(\frac{\pi}{2}-\frac{\lambda}{4R}\right)+1\right)}{2} = \frac{\lambda\cdot\frac{\lambda}{8\pi R}}{2} = \frac{\lambda^2}{16\pi R}$$

Applying Theorem 11 with $d = 2$ yields $h \geq \frac{\sqrt{3}\cdot\lambda^2}{16\pi\Delta}$ for S_λ.

Analog, the height of $S_{\lambda\cdot\tau/2}$ is $\geq \frac{\sqrt{3}\lambda^2\tau^2}{64\pi\Delta}$. ■

For any well-connected configuration, we state an important observation.

Lemma 2.2 Let the considered swarm be *well-connected*, i.e. let the viewing range be $1+\tau$ for a constant $\tau > 0$ and let the UDG be connected. If $\mathrm{diam}(t) > \tau$, then $\mathrm{diam}_i(t) > \tau$, for every robot r_i.

Proof. We prove the claim by contradiction. Let the UDG be connected and $\mathrm{diam}(t) > \tau$. To derive the contradiction, we assume that there is a robot r_i with $\mathrm{diam}_i(t) \leq \tau$. By definition, $|p_i(t) - p_k(t)| \leq \tau$ for all $r_k \in N_i(t)$ (the neighborhood of r_i). Consequently, there exists at least one robot $r_j \notin N_i(t)$. For all robots $r_j \notin N_i(t)$, $|p_i(t) - p_j(t)| > 1+\tau$. For all $r_k \in N_i(t)$ and $r_j \notin N_i(t)$, we have $|p_k(t) - p_j(t)| > 1$. Hence, none of r_i's neighbors is at a distance of at most 1 from a robot that r_i cannot see. This is a contradiction since we have assumed that the UDG is connected. Hence, $\mathrm{diam}_i(t) > \tau$. ■

Due to the λ-contracting property, robots close to the boundary of the global smallest enclosing circle (SEC) move upon activation at least $\Omega\left(\frac{\mathrm{diam}_i(t)}{\Delta}\right)$ inwards. With $\mathrm{diam}_i(t) > \tau$, it follows that the radius of the SEC decreases by $\Omega(\tau/\Delta)$ after each robot was active at least once (see Lemma 2.3). Consequently, $\mathrm{diam}(t) \leq \tau$ after $\mathcal{O}(\Delta^2)$ rounds.

Lemma 2.3 Let $\mathcal{P}$ be a λ-contracting protocol. For a robot r_i with $\operatorname{diam}_i(t) > \tau$, $\operatorname{target}_i^{\mathcal{P}}(t) \in N \setminus S_{\lambda\cdot\tau/2}$.

Proof. Since $\operatorname{diam}_i(t) > c$ and $\mathcal{P}$ is λ-contracting, $\operatorname{target}_i^{\mathcal{P}}(t)$ is the midpoint of a line segment $\ell_i^{\mathcal{P}}(t)$ of length at least $\lambda \cdot \operatorname{diam}_i(t) > \lambda \cdot \tau$. Observe that the maximum distance between any pair of points in $S_{\lambda\cdot\tau/2}$ is $\tau \cdot \lambda/2$. It follows directly, that the midpoint of $\ell_i^{\mathcal{P}}(t)$ cannot lie inside $S_{\lambda\cdot\tau/2}$. ■

Theorem 2 — restated. Consider a swarm of robots in $\mathbb{R}^2$ with diameter Δ that is *well-connected* Every λ-contracting NEAR-GATHERING protocol solves NEAR-GATHERING after $\mathcal{O}\left(\Delta^2\right)$ rounds.

Note, that the protocol works under the $\mathcal{S}$SYNC scheduler as well. We provide the proof directly considering the $\mathcal{S}$SYNC scheduler which allows us follow directly observation Observation 2.3 below.

Proof. The initial configuration is well-connected, and a λ-contracting NEAR-GATHERING protocol keeps it well-connected by definition. W.l.o.g let the viewing range of $\mathcal{P}$ be $1+\tau$ for a constant $\tau > 0$ and let the UDG (with respect to distance 1) be connected. As long $\operatorname{diam}(t) > \tau$, by Lemma 2.2 we know that $\operatorname{diam}_i(t) > \tau$. From Lemma 2.3, it follows that a robot leaves $S_{\lambda\cdot\tau/2}$ when it becomes active. This happens for all robots at most once per epoch. Hence, $R(t)$, the radius of the (global) SEC of all robots in epoch t, decreases by h, where h denotes the height of $S_{\lambda\cdot\tau/2}$. In Lemma 2.1, we showed that the height h of $S_{\lambda\cdot\tau/2} \geq \frac{\sqrt{3}\lambda^2\tau^2}{64\pi\Delta}$. Thus, $R(t)$ decreases by at least $\frac{\sqrt{3}\lambda^2\tau^2}{64\pi\Delta}$ in one epoch. By Theorem 11, we know that the initial global SEC has a radius of at most $\Delta/\sqrt{2}$. After $\frac{\Delta/\sqrt{2}}{h} = \mathcal{O}(\Delta^2)$ epochs, the global SEC has a radius $\leq \tau/2$ and $\operatorname{diam}(t) \leq \tau$.

From Lemma 2.2 follows directly, that robot r_i can locally decide that the swarm has finished NEAR-GATHERING if $\operatorname{diam}_i(t) < \tau$. All robots detect this in the same epoch and terminate simultaneously. This matches the definition of NEAR-GATHERING (Definition 1.4). ■

Observation 2.3 Consider a swarm of robots in $\mathbb{R}^2$ with diameter Δ that is *well-connected* Every λ-contracting NEAR-GATHERING protocol solves NEAR-GATHERING after $\mathcal{O}\left(\Delta^2\right)$ semi-synchone epoch.

2.3 From Contracting Protocols to Near-Gathering

In this section, we introduce a method to transform any λ-contracting protocol into a λ-contracting NEAR-GATHERING protocol. A λ-contracting protocol $\mathcal{P}$, in general, does neither keep the swarm well-connected nor prevents collisions; both properties are crucial for λ-contracting NEAR-GATHERING protocols. Well-connectedness requires that the connectivity range is by a constant smaller than the viewing range, but $\mathcal{P}$ has a viewing range that might be identical to the connectivity range. We normalized the distances so that the connectivity range of $\mathcal{P}$ is 1. Our new protocol $\mathcal{P}^{cl}$ will have the same connectivity range and a viewing range of $1+\tau$ for a constant $0 < \tau < 2/3$. Note that the upper bound on τ is only required because $\tau/2$ also represents the maximum movement distance of a robot (see below). In general, the viewing range could also be chosen larger than $1+\tau$ without any drawbacks while keeping the maximum movement distance at $\tau/2$.

The main idea of our approach can be summarized as follows: first, robots compute a *potential* target point based on a λ-contracting protocol $\mathcal{P}$ that considers only robots at a distance at most 1. Afterward, a robot r_i uses the viewing range of $1+\tau$ to determine whether its potential target point collides with any potential target point of a nearby neighbor. If there might be a collision, r_i does not move to its potential target point. Instead, it only moves to a point between itself and the potential target point where no other robot moves. At the same time, it is also ensured that r_i moves sufficiently far towards the potential target point to maintain the time bound of $\mathcal{O}\left(\Delta^2\right)$ rounds. To realize the ideas with a viewing range of $1+\tau$, we restrict the maximum movement distance of any robot to $\tau/2$. More precisely, if the potential target point of any robot given by $\mathcal{P}$ is at a distance of more than $\frac{\tau}{2}$, the robot moves at most $\frac{\tau}{2}$ towards it. With this restriction, each robot could only collide with other robots at a distance of at most τ. The viewing range of $1+\tau$ allows computing the potential target point based on $\mathcal{P}$ of all neighbors at a distance at most τ. By knowing all these potential target points, the own target point of the collision-free protocol can be chosen. While this only summarizes the key ideas, we give a more technical intuition and a summary of the proof in Section 2.4.

Theorem 4 — restated. For every λ'-contracting protocol $\mathcal{P}'$ there exist a λ-contracting NEAR-GATHERING protocol $\mathcal{P}$ with $\lambda \in \mathcal{O}(\lambda')$.

2.3.1 The Protocol

In this subsection, we define the collision-free protocol $\mathcal{P}^{cl}(\mathcal{P},\tau,\varepsilon)$ (usually abbreviated with $\mathcal{P}^{cl}$). Its construction depends on the parameters $\mathcal{P}$, ε, and τ that we briefly define. $\mathcal{P}$ is a λ-contracting protocol (designed for robots with a viewing range of 1). The constant τ has two purposes. The robots have a viewing range of $1+\tau$, and $\tau/2$ is the maximum movement distance of any robot, $0<\tau\leq 2/3$. Lastly, the constant $\varepsilon \in (0,1/2)$ determines how close each robot moves towards its target point based on $\mathcal{P}$. To define the protocol $\mathcal{P}^{cl}(\mathcal{P},\tau,\varepsilon)$, the local by robot r_i computed target function $\text{target}_i^{\mathcal{P}^{cl}(\mathcal{P},\tau,\varepsilon)}(t)$ (usually abbreviated with $\text{target}_i^{\mathcal{P}^{cl}}(t)$) is stated in Algorithm 1.

The computation of $\text{target}_i^{\mathcal{P}^{cl}}(t)$ is based on the movement r_i would do in a slightly modified version of $\mathcal{P}$, denoted as $\mathcal{P}^{\tau}$. The protocol $\mathcal{P}^{\tau}$ is defined by $\text{target}_i^{\mathcal{P}^{\tau}}(t)$ in Algorithm 3 and a detailed intuition of why it is needed can be found alongside the proof in Section 2.2. The position of $\text{target}_i^{\mathcal{P}^{cl}}(t)$ lies on the *collision vector* $\text{collvec}_i^{\mathcal{P}^{\tau}}(t)$, the vector from $p_i(t)$ to $\text{target}_i^{\mathcal{P}^{\tau}}(t)$. On $\text{collvec}_i^{\mathcal{P}^{\tau}}(t)$, there may be several *collision points*. These are either current positions, potential target points ($\text{target}_k^{\mathcal{P}^{\tau}}(t)$) of other robots r_k or single intersection points between $\text{collvec}_i^{\mathcal{P}^{\tau}}(t)$ and another collision vector $\text{collvec}_k^{\mathcal{P}^{\tau}}(t)$. The computation $\text{target}_i^{\mathcal{P}^{\tau}}(t)$ of collision points is defined in Algorithm 2. Let $d_i>0$ be the minimal distance between a collision point and $\text{target}_i^{\mathcal{P}^{\tau}}(t)$. The final target point $\text{target}_i^{\mathcal{P}^{cl}}(t)$ is exactly at a distance $d_i\cdot\varepsilon\cdot 2/\tau\cdot\left|\text{collvec}_i^{\mathcal{P}^{\tau}}(t)\right|$ from $\text{target}_i^{\mathcal{P}^{\tau}}(t)$. Figure 2.6 gives an example of collision points and target points of $\mathcal{P}^{cl}$.

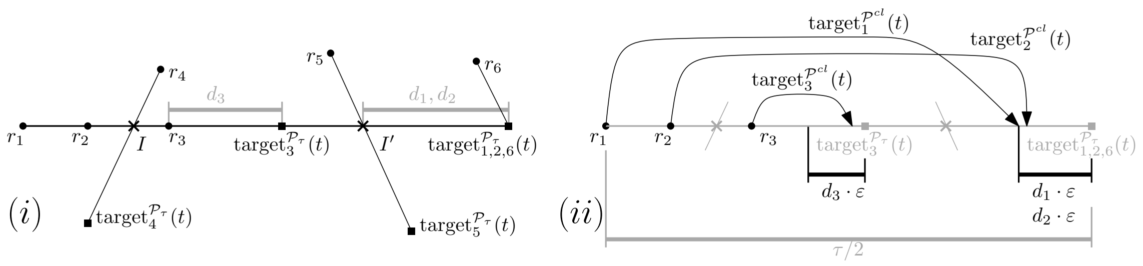

r_1
r_2
r_3
r_4
r_5
r_6
I
I'
d_3
d_1, d_2
$\text{target}_3^{\mathcal{P}^\tau}(t)$
$\text{target}_{1,2,6}^{\mathcal{P}^\tau}(t)$
$\text{target}_4^{\mathcal{P}^\tau}(t)$
$\text{target}_5^{\mathcal{P}^\tau}(t)$
(i)
$\text{target}_1^{\mathcal{P}^{cl}}(t)$
$\text{target}_2^{\mathcal{P}^{cl}}(t)$
$\text{target}_3^{\mathcal{P}^{cl}}(t)$
r_1
r_2
r_3
$\text{target}_3^{\mathcal{P}^\tau}(t)$
$\text{target}_{1,2,6}^{\mathcal{P}^\tau}(t)$
$d_3 \cdot \varepsilon$
$d_1 \cdot \varepsilon$
$d_2 \cdot \varepsilon$
(ii)
$\tau/2$

Figure 2.6: Example of $\text{target}_i^{\mathcal{P}^{cl}}(t)$ with $\tau = 2/3$ and $\varepsilon = 0.49$. (i) shows the collision points and computation of d_1, d_2 and d_3 (line 3 in Algorithm 1). (ii) shows the positions where r_1, r_2 and r_3 will move to in protocol $\mathcal{P}^{cl}$ as returned by Algorithm 1.

Algorithm 1 $\text{target}_i^{\mathcal{P}^{cl}(\mathcal{P},\tau,\varepsilon)}(t)$

```
1: $R_i \leftarrow \{r_k : |p_k(t) - p_i(t)| \leq \tau\}$    ▷ Robots in radius $\tau$ around $r_i$ (including $r_i$)
2: $C_i \leftarrow \text{collisionPoints}_i^{\mathcal{P}^\tau}(R_i, t)$    ▷ Collision points on $\text{collvec}_i^{\mathcal{P}^\tau}(t)$, see Algorithm 2
3: $d_i \leftarrow \min\left(\left\{\left|c - \text{target}_i^{\mathcal{P}^\tau}(t)\right| : c \in C_i \setminus \{\text{target}_i^{\mathcal{P}^\tau}(t)\}\right\}\right)$    ▷ min. dist. to collision point
4: return point on $\text{collvec}_i^{\mathcal{P}^\tau}(t)$ with distance $d_i \cdot \varepsilon \cdot {}^2/_\tau \cdot \left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right|$ to $\text{target}_i^{\mathcal{P}^\tau}(t)$
```

2.4 Proofs for Near-Gathering Protocols $\mathcal{P}^{cl}(\mathcal{P}, \tau, \varepsilon)$

In this section, the proof is given that $\mathcal{P}^{cl}(\mathcal{P}, \tau, \varepsilon)$ is a λ-contracting NEAR-GATHERING protocol and therefore solves the NEAR-GATHERING problem for arbitrary initial configurations where the UDG is connected. Because the proof is closely interconnected with the technical intuitions behind the protocol, both are explained side by side. The entire protocol $\mathcal{P}^{cl}$ is described in Section 2.3.1. The idea for $\mathcal{P}^{cl}$ is straightforward: robots compute a potential target point based on a λ-contracting protocol $\mathcal{P}$ (that uses a viewing range of 1), restrict the maximum movement distance to ${}^\tau/_2$ and use the viewing range of $1 + \tau$ to avoid collisions with robots in the distance at most τ. However, there are several technical details we want to emphasize.

The protocol $\mathcal{P}^\tau$ is merely an intermediate protocol to increase the viewing range without violating the λ-contracting properties of a protocol. It is analyzed in subsection

Algorithm 2 $\text{collisionPoints}_i^{\mathcal{P}}(R_i, t)$

```
1: $C_i \leftarrow$ empty set
2: for all $r_k \in R_i$ do
3:     compute $\text{target}_k^{\mathcal{P}}(t)$ and $\text{collvec}_k^{\mathcal{P}}(t)$ in local coordinate system of $r_i$
4:     if $p_k(t) \in \text{collvec}_k^{\mathcal{P}}(t)$ then
5:         add $p_k(t)$ to $C_i$    ▷ position of $r_k$
6:     if $\text{target}_k^{\mathcal{P}}(t) \in \text{collvec}_i^{\mathcal{P}}(t)$ then
7:         add $\text{target}_k^{\mathcal{P}}(t)$ to $C_i$
8:     if $\text{collvec}_k^{\mathcal{P}}(t)$ intersects $\text{collvec}_i^{\mathcal{P}}(t)$ and is not collinear to $\text{collvec}_i^{\mathcal{P}}(t)$ then
9:         add intersection point between $\text{collvec}_k^{\mathcal{P}}(t)$ and $\text{collvec}_i^{\mathcal{P}}(t)$ to $C_i$
10: return $C_i$
```

Algorithm 3 $\text{target}_i^{\mathcal{P}^\tau}(t)$

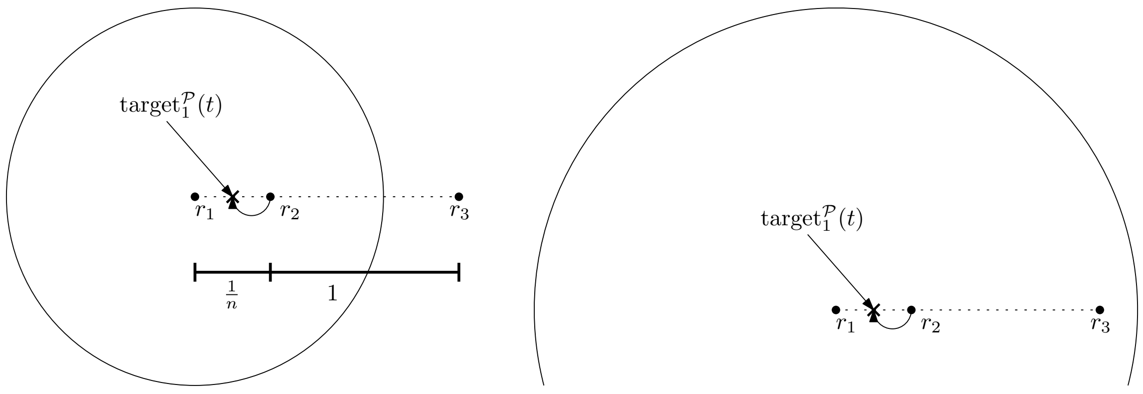
```
1: if robots in range 1 have pairwise distance $\leq \tau/2$ then
2:     $\mathcal{P}^{1+\tau/2} \leftarrow$ protocol $\mathcal{P}$ scaled to viewing range $1+\tau/2$
3:     $P_i \leftarrow \text{target}_i^{\mathcal{P}^{1+\tau/2}}(t)$
4: else
5:     $P_i \leftarrow \text{target}_i^{\mathcal{P}}(t)$
6: if distance $p_i(t)$ to $P_i > \tau/2$ then
7:     return point with distance $\tau/2$ to $p_i(t)$ between $p_i(t)$ and $P_i$
8: else
9:     return $P_i$
```

Section 2.4.1, where we show that it keeps the UDG connected (Lemma 2.4) and is λ-contracting (Lemma 2.5). These properties are important because they are directly translatable to $\mathcal{P}^{cl}$ (Lemmas 2.11 and 2.12). The protocol $\mathcal{P}^{cl}$ adds collision avoidance and is analyzed in Section 2.4.2. Some of the more technical proofs showing that the protocol indeed prevents collisions (Lemma 2.10) can be found in Section 2.4.5. All statements are combined in Section 2.4.3 to prove Theorem 2.

2.4.1 Analysis of the Intermediate Protocol $\mathcal{P}^\tau$

Recall that the main goal is to compute potential target points based on a λ-contracting protocol $\mathcal{P}$ with viewing range 1. Let us ignore the collision avoidance in this section and only concentrate on the λ-contracting properties of such a protocol if applied to a scenario with a viewing range of $1+\tau$. Unfortunately, a direct translation of the protocol loses the λ-contracting property in general. Consider the following example, which is depicted in Figure 2.7. The robots r_1, r_2 and r_3 are on one line with respective distances to r_1 of $1/n$ and $1+1/n$. Now assume the protocol $\mathcal{P}$ with viewing range 1, where r_1 moves to the midpoint between r_1 and r_2 robot; $\text{diam}_1(t) = 1/n$ and $\text{target}_1^{\mathcal{P}}(t)$ is 1-centered. By increasing r_1's viewing range to $1+\tau$, r_1 observes r_3, which it could not see before. Thus, $\text{diam}_1(t)$ increases significantly to $1+1/n$. Now assume r_1 still moves to the midpoint between r_1 and r_2. The maximal line segment with the target point as the center has length $1/n$. However, there is no *constant* λ such that $1/n \geq \lambda \cdot (1+1/n)$ for all n (λ depends on n). Hence, in the example, the protocol $\mathcal{P}$ is not λ-contracting (Definition 2.2) anymore because of the increased viewing range.

Next, we argue how to transform the protocol $\mathcal{P}$ with viewing range 1 into a protocol $\mathcal{P}^\tau$ with viewing range $1+\tau$ such that $\mathcal{P}^\tau$ is λ-contracting gathering protocol. The example above already emphasizes the main problem: robots can have very small local diameters $\text{diam}_i(t)$. Instead of moving according to $\mathcal{P}$, those robots compute a target point based on $\mathcal{P}^{1+\tau/2}$, which is a λ-contracting gathering protocol concerning the viewing range of $1+\tau/2$. Protocol $\mathcal{P}^{1+\tau/2}$ is obtained by scaling $\mathcal{P}$ to the larger viewing range of $1+\tau/2$. More precisely, robots r_i with $\text{diam}_i(t) \leq \tau/2$ compute their target points based on $\mathcal{P}^{1+\tau/2}$ and all others according to $\mathcal{P}$. In addition, $\mathcal{P}^\tau$ ensures that no robot moves more than a distance of $\tau/2$ towards the target points computed in $\mathcal{P}$ and $\mathcal{P}^{1+\tau/2}$. The first reason is to maintain the connectivity of $\text{UDG}(t)$. While the protocol $\mathcal{P}$ maintains connectivity by definition, the protocol $\mathcal{P}^{1+\tau/2}$ could violate the connectivity of $\text{UDG}(t)$. Restricting the movement distance to $\tau/2$ and upper bounding τ by $2/3$ resolves this issue.



$\text{target}_1^{\mathcal{P}}(t)$
r_1
r_2
r_3
$\frac{1}{n}$
1
$\text{target}_1^{\mathcal{P}}(t)$
r_1
r_2
r_3

Figure 2.7: To the left, r_1 has a viewing range of 1 (circle) and $\text{diam}_1(t) = 1/n$. It moves to the 1-centered point $\text{target}_1^{\mathcal{P}}(t)$. The robot r_3 is not visible to r_1.
To the right, the same setup with a viewing range of $1+\tau$ is depicted; now, r_1 can also see r_3 and $\text{diam}_1(t) = 1 + 1/n$. $\text{target}_1^{\mathcal{P}}(t)$ remains unchanged and is not λ-centered for $\lambda \in \omega(1/n)$.

Lemma 2.4 Let $\mathcal{P}$ be a λ-contracting protocol with a viewing range of 1. The UDG stays connected while executing $\mathcal{P}^\tau$.

Proof. Any protocol $\mathcal{P}$ must hold the connectivity concerning its connectivity range 1. However, with scaling the protocol up to a viewing range of $1+\tau/2$, its connectivity range increases as well, hence the protocol $\mathcal{P}^{1+\tau/2}$ only guarantees connectivity with respect to distance $1+\tau/2$. Now suppose that a robot r_i moves according to $\mathcal{P}^{1+\tau/2}$ in $\mathcal{P}^\tau$. This only happens if there is no robot in distance $\text{dist} \in (\tau/2, 1]$ around r_i, all connected robots r_k have a distance $\leq \tau/2$ before the movement. r_i and r_k can both move at most a distance of $\tau/2$ in one round. It follows that in the next round their distance is $\leq 3 \cdot \tau/2 \leq 1$ because $\tau \leq 2/3$ by definition. ■

The second reason for bounding the movement distance by $\tau/2$ is that collisions are only possible within a range of τ. While $\mathcal{P}^\tau$ has a viewing range of $1+\tau$, it never uses its full viewing range for computing its target point. The additional viewing range is utilized in $\mathcal{P}^{cl}$ for collision avoidance. It is easy to see that the configuration in Figure 2.7 does not violate the λ-contracting property of $\mathcal{P}^\tau$. If $1/n > \tau/2$ it is trivial that $\text{target}_1^{\mathcal{P}}(t)$ is λ-centered in $\mathcal{P}^\tau$ with $\lambda \in \mathcal{O}(\tau)$. Else, $\text{target}_1^{\mathcal{P}^\tau}(t) = \text{target}_1^{\mathcal{P}^{1+\tau/2}}(t)$. $\mathcal{P}^{1+\tau/2}$ also considers r_3 in distance $1 + 1/n \leq 1 + \tau/2$ (note, there always exists a robot between r_2 and r_4 to ensure $\text{UDG}(t)$ is connected). Hence, $\text{target}_1^{\mathcal{P}^{1+\tau/2}}(t)$ is $\lambda \cdot \frac{1+\tau/2}{1+\tau}$-centered in $\mathcal{P}^\tau$. This argument is generalized in Lemma 2.5 to show that $\mathcal{P}^\tau$ is a λ-contracting gathering protocol. Technically, the same problem as described above can still happen: The robot set used for the computation of $\text{target}_i^{\mathcal{P}^\tau}(t)$ has a relatively small diameter while $\text{diam}_i(t) > 1 + \tau/2$. Nevertheless, contrary to the example above, the robot set cannot have an arbitrarily small diameter in such a configuration. $\mathcal{P}^{1+\tau/2}$ is simulated if the set has a diameter $\leq \tau/2$. $\mathcal{P}^{1+\tau/2}$ utilizes an additional radius of $\tau/2$ of the viewing range to make sure that always a robot in constant distance > 1 is used for the computation. This property leads to the following proof, showing that $\mathcal{P}^\tau$ is λ-contracting.

Lemma 2.5 Let $\mathcal{P}$ be a λ-contracting protocol. $\mathcal{P}^\tau$ is λ'-contracting with $\lambda' = \lambda \cdot \frac{\tau}{4\cdot(1+\tau)}$.

Proof. By Lemmas 2.2 and 2.4, it follows that $\mathrm{diam}_i(t) \geq \tau/2$ for all i. The protocol $\mathcal{P}^\tau$ has a viewing range of $1+\tau$. $\mathrm{target}_i^{\mathcal{P}^\tau}(t)$ either equals to $\mathrm{target}_i^{\mathcal{P}}(t)$ ($\mathcal{P}$ with viewing range 1) or $\mathrm{target}_i^{\mathcal{P}^{1+\tau/2}}(t)$ ($\mathcal{P}^{1+\tau/2}$ with viewing range $1+\tau/2$), dependent on the local diameter $\mathrm{diam}_i(t)$ of a robot. If $\mathrm{target}_i^{\mathcal{P}^\tau}(t) = \mathrm{target}_i^{\mathcal{P}}(t)$, we know that a line segment with length $\lambda \cdot \tau/2$ exists with $\mathrm{target}_i^{\mathcal{P}}(t)$ as midpoint, because the robots used to simulate have at least a diameter of $\tau/2$ (see condition in line 1 of Algorithm 3) and $\mathcal{P}$ is λ-contracting.

By Lemma 2.4 we know that $\mathrm{UDG}(t)$ stays connected. $\mathcal{P}^{1+\tau/2}$ has a viewing range of $1+\tau/2$ By Lemma 2.2, it follows that the diameter of robots used for simulating $\mathcal{P}^{1+\tau/2}$ is $\geq \tau/2$ if $\mathrm{diam}(t) \geq \tau/2$. Because $\mathcal{P}^{1+\tau/2}$ is λ-contracting, there exists a line segment with length $\lambda \cdot \tau/2$ trough $\mathrm{target}_i^{\mathcal{P}^{1+\tau/2}}(t)$.

The local diameter is naturally bounded by $\mathrm{diam}_i(t) \leq 2(1+\tau)$. The length of the above described line segment with $\mathcal{P}^\tau$ as midpoint is $\lambda \cdot \tau/2 = \lambda \cdot \frac{\tau/2}{2(1+\tau)} \cdot 2(1+\tau) \geq \lambda \cdot \frac{\tau}{4(1+\tau)} \cdot \mathrm{diam}_i(t)$. Therefore, $\mathcal{P}^\tau$ is λ'-contracting GATHERING protocol with $\lambda' = \lambda \cdot \frac{\tau}{4\cdot(1+\tau)}$. ■

To conclude, the protocol $\mathcal{P}^\tau$ has two main properties: it restricts the movement distance of any robot to at most $\tau/2$ and robots r_i with $\mathrm{diam}_i(t) \leq \tau/2$ compute their target points based on protocol $\mathcal{P}^{1+\tau/2}$ with viewing range $1+\tau/2$.

2.4.2 Analysis of the Protocol $\mathcal{P}^{cl}$

Next, we argue how to transform the protocol $\mathcal{P}^\tau$ into the collision-free protocol $\mathcal{P}^{cl}$. The viewing range of $1+\tau$ in $\mathcal{P}^{cl}$ allows a robot r_i to compute $\mathrm{target}_k^{\mathcal{P}^\tau}(t)$ (the target point in protocol $\mathcal{P}^\tau$) for all robots r_k within distance at most τ. Since the maximum movement distance of a robot in $\mathcal{P}^\tau$ is $\tau/2$, this enables r_i to know the movement directions of all robots r_k which can collide with r_i. $\mathcal{P}^{cl}$ ensures that each robot r_i moves to some position on $\mathrm{collvec}_i^{\mathcal{P}^\tau}(t)$ and avoids positions of all other $\mathrm{collvec}_k^{\mathcal{P}^\tau}(t)$. Henceforth, no collision can happen. While this is the basic idea of our collision avoidance, there are some details to add.

First of all, $\mathcal{P}^\tau$ has the same viewing range as $\mathcal{P}^{cl}$ of $1+\tau$. However, it never uses the full viewing range to compute the target position $\mathrm{target}_i^{\mathcal{P}^\tau}(t)$. This ensures that each robot is able to compute $\mathrm{target}_k^{\mathcal{P}^\tau}(t)$ for robots in the nearest surroundings.

Lemma 2.6 Let $\mathcal{P}$ be a λ-contracting protocol with a viewing range of 1. A viewing range of $1+\tau$ is sufficient to compute $\mathrm{target}_k^{\mathcal{P}^\tau}(t)$ for all robots r_k within a radius of τ.

Proof. The computation requires that the entire neighborhood of r_k relevant to compute $\mathrm{target}_k^{\mathcal{P}^\tau}(t)$ is also in r_i's neighborhood. Depending on whether there exists a robot in distance $\mathrm{dist} \in (\tau/2, 1]$ around r_k, is this the neighborhood relevant for $\mathrm{target}_k^{\mathcal{P}}(t)$ or $\mathrm{target}_k^{\mathcal{P}^\tau}(t)$ (first if/else block in Algorithm 3).

$\mathrm{target}_k^{\mathcal{P}}(t)$ needs a viewing range of 1 around r_k. The distance between r_k and r_i is at most τ and r_i has a viewing range of $1+\tau$, therefore all robots relevant for computing $\mathrm{target}_k^{\mathcal{P}}(t)$ are in r_i's neighborhood. $\mathrm{target}_k^{\mathcal{P}^\tau}(t)$ needs a viewing range of $1+\tau/2$ around r_k. The condition that the pairwise distance between robots in range 1 around r_k is $\leq \tau/2$

makes sure that $|p_i(t) - p_k(t)| \leq \tau/2$. r_i has a viewing range of $1+\tau$, therefore are all robots relevant for computing $\text{target}_k^{\mathcal{P}^\tau}(t)$ are in r_i's neighborhood. ■

Secondly, r_i cannot avoid positions on all other $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ in some cases. For instance, $\text{collvec}_i^{\mathcal{P}^\tau}(t)$ may be completely contained in $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ (e.g., $\text{collvec}_2^{\mathcal{P}^\tau}(t) \in \text{collvec}_1^{\mathcal{P}^\tau}(t)$ in the example depicted in Figure 2.6). In case $\text{collvec}_i^{\mathcal{P}^\tau}(t)$ and $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ are not collinear and intersect in a single point, both robots simply avoid the intersection point (e.g. r_1 and r_4 in the example).

Lemma 2.7 No robot moves to a point that is the intersection of two collision vectors that are not collinear.

If $\text{collvec}_i^{\mathcal{P}^\tau}(t)$ and $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ are collinear, both robots move to a point closer to their target point than to the other one (e.g., r_1 and r_3 in the example).

Lemma 2.8 If the target points of robots are different in $\mathcal{P}^\tau$ they are different in $\mathcal{P}^{cl}$.

But there are cases, in which robots have the same target point in $\mathcal{P}^\tau$ (e.g. r_1, r_2 and r_6 in the example). Because robots stay in the same direction towards the target point, collisions can only happen if one robot is currently on the collision vector of another one (e.g., r_2 is on $\text{collvec}_1^{\mathcal{P}^\tau}(t)$). Their movement is scaled by the distance to the target point, which must be different. Therefore, their target points in $\mathcal{P}^{cl}$ must be different as well.

Lemma 2.9 If the target points of robots are the same in $\mathcal{P}^\tau$ they are different in $\mathcal{P}^{cl}$.

The proof of Lemmas 2.8 and 2.9 can be found in Section 2.4.5. Both lemmas combined yield the following statement.

Lemma 2.10 The protocol $\mathcal{P}^{cl}$ is collision-free.

It remains to show that $\mathcal{P}^{cl}$ is still λ-contracting and keeps the UDG connected.

Lemma 2.11 If $\mathcal{P}$ is a λ'-contracting protocol, $\mathcal{P}^{cl}$ is λ-contracting with $\lambda = \lambda' \cdot \frac{\tau}{4\cdot(1+\tau)} \cdot (1-\varepsilon)$.

Proof. From Lemma 2.5, we obtain that $\mathcal{P}^\tau$ is λ''-contracting with $\lambda'' = \lambda' \cdot \frac{\tau}{4\cdot(1+\tau)}$. Because $\mathcal{P}^\tau$ and $\mathcal{P}^{cl}$ have the same viewing range, $\text{target}_i^{\mathcal{P}^\tau}(t)$ is always λ-centered for $\mathcal{P}^{cl}$.

$\text{target}_i^{\mathcal{P}^{cl}}(t)$ is chosen such that a robot moves in direction $\text{target}_i^{\mathcal{P}^\tau}(t)$ and $\frac{\text{target}_i^{\mathcal{P}^{cl}}(t)}{\text{target}_i^{\mathcal{P}^\tau}(t)} \geq (1-\varepsilon)$. Analogous to the arguments in Theorem 15 of [10], we can follow with the intercept theorem, that $\lambda = \lambda'' \cdot (1-\varepsilon) = \lambda' \cdot \frac{\tau}{4\cdot(1+\tau)} \cdot (1-\varepsilon)$.

■

Lemma 2.12 Let $\mathcal{P}$ be a λ-contracting protocol with a viewing range of 1. $\mathcal{P}^{cl}$ has a viewing range of $1+\tau$ and the UDG stays connected while executing $\mathcal{P}^{cl}$.

From this lemma can directly be followed that $\mathcal{P}^{cl}$ keeps the swarm well-connected (Definition 1.5).

Proof. $\mathcal{P}^{cl}$ has the same viewing range as $\mathcal{P}^{\tau}$ that is $1+\tau$ (Lemma 2.6). If the UDG is connected in the initial configuration, it will stay connected while executing $\mathcal{P}^{\tau}$ (Lemma 2.4). Because of the semi-synchronous environment, we know that r_i, r_k with $|p_i(t)-p_k(t)| \leq 1$ implies $\left|\text{target}_i^{\mathcal{P}^{\tau}}(t)-\text{target}_k^{\mathcal{P}^{\tau}}(t)\right| \leq 1$ and $\left|\text{target}_i^{\mathcal{P}^{\tau}}(t)-p_k(t)\right| \leq 1$, respectively $\left|p_i(t)-\text{target}_k^{\mathcal{P}^{\tau}}(t)\right| \leq 1$ (in case r_k, respectively r_i, is inactive in round t and $p_k(t)=p_k(t+1)$). If both endpoints of both collision vectors $\text{collvec}_i^{\mathcal{P}^{\tau}}(t)$ and $\text{collvec}_k^{\mathcal{P}^{\tau}}(t)$ are pairwise at a distance ≤ 1, all points on both vectors are pairwise at a distance ≤ 1. For all robots $\text{target}_i^{\mathcal{P}^{cl}}(t) \in \text{collvec}_i^{\mathcal{P}}(t)$ and therefore, the UDG stays connected while executing $\mathcal{P}^{cl}$. ■

2.4.3 Proof of Theorem 4

Theorem 4 — restated. For every λ'-contracting protocol $\mathcal{P}'$ there exist a λ-contracting NEAR-GATHERING protocol $\mathcal{P}$ with $\lambda \in \mathcal{O}(\lambda')$.

We prove the following more precise formulation of the theorem.

For every λ'-contracting protocol $\mathcal{P}$, every $\tau \in (0, 2/3]$ and every $\varepsilon(0, 1/2)$ there exists the λ-contracting NEAR-GATHERING protocol $\mathcal{P}^{cl}(\mathcal{P},\tau,\varepsilon)$ with $\lambda = \lambda' \cdot \frac{\tau}{4\cdot(1+\tau)} \cdot (1-\varepsilon)$

Proof. We have shown the following three properties defined in Definition 2.3, to conclude the statement. (Lemma 2.11) $\mathcal{P}^{cl}$ is a collisionless λ-contracting protocol with $\lambda = \lambda' \cdot \frac{\tau}{4\cdot(1+\tau)} \cdot (1-\varepsilon)$. (Lemma 2.10) $\mathcal{P}^{cl}$ is collision-free. (Lemma 2.12) $\mathcal{P}^{cl}$ keeps the swarm well-connected.

■

2.4.4 Examples of a λ-contracting NEAR-GATHERING protocol

Jannik Castenow gives in his doctoral thesis [7] a few examples of λ-contracting protocols. He most notably proved that GO-TO-THE-CENTER [24] is $\left(\sqrt{3}/16\right)$-contracting. Together with our theorem above that leads to the following result.

Theorem 12 GO-TO-THE-CENTER [24] is a λ'-contracting protocol that can be transformed into an λ-contracting NEAR-GATHERING protocol with $\lambda = \mathcal{O}(\lambda')$.

2.4.5 Proof of Lemma 2.10

Lemma 2.10 — restated. The protocol $\mathcal{P}^{cl}$ is collision-free.

The lemma follows directly from Lemmas 2.8 and 2.9.

Lemma 2.7 — restated. No robot moves to a point that is the intersection of two collision vectors that are not collinear.

Proof. More formally, we prove the following statement: Let $\text{collvec}_i^{\mathcal{P}^{\tau}}(t)$ and $\text{collvec}_k^{\mathcal{P}^{\tau}}(t)$ be collision vectors that intersect in a single point I. $\text{target}_i^{\mathcal{P}^{cl}}(t) \neq I$. r_i and r_k have a distance of at most τ, because the movement distance of $\tau/2$ is an upper bound for the length of $\text{collvec}_i^{\mathcal{P}^\tau}(t)$, respectively $\text{collvec}_k^{\mathcal{P}^\tau}(t)$. Hence, r_k is in R_i as computed in line 1 of Algorithm 1. In line 3 of Algorithm 3, the collision vector $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ is checked for intersections with $\text{collvec}_i^{\mathcal{P}^\tau}(t)$. By Lemma 2.6, we know, that $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ is computable by r_i with the available viewing range of $1+\tau$. It follows that I is in C_i (l. 2 Algorithm 1). $\text{target}_i^{\mathcal{P}^{cl}}(t)$ is some point in between the nearest points in $C_i \setminus \{p_i\}$ and $\text{target}_i^{\mathcal{P}^\tau}(t)$. This can never be I. ■

Lemma 2.8 — restated. If the target points of robots are different in $\mathcal{P}^\tau$ they are different in $\mathcal{P}^{cl}$.

Proof. More formally, we prove the following statement: Let r_i and r_k be two robots with $\text{target}_i^{\mathcal{P}^\tau}(t) \neq \text{target}_k^{\mathcal{P}^\tau}(t)$. It follows that $\text{target}_i^{\mathcal{P}^{cl}}(t) \neq \text{target}_k^{\mathcal{P}^{cl}}(t)$. If $\text{collvec}_i^{\mathcal{P}^\tau}(t)$ and $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ are not collinear, the statement follows directly from Lemma 2.7. We consider both robots with collinear $\text{collvec}_i^{\mathcal{P}^\tau}(t)$ and $\text{collvec}_k^{\mathcal{P}^\tau}(t)$. Let $P_i = \text{target}_i^{\mathcal{P}^\tau}(t)$, $P_k = \text{target}_k^{\mathcal{P}^\tau}(t)$. We distinguish all three cases how r_i, P_i and P_k can be arranged: P_k is between r_i and P_i; r_i is between P_i and P_k; P_i is between r_i and P_k.

- Case P_k is between r_i and P_i: Analogous to the arguments in Lemma 2.7, $r_k \in R_i$ (line 1 Algorithm 1) and P_k is added to C_i (line 7 Algorithm 2). d_i (line 3 Algorithm 1) is at most the distance between P_i and P_k. r_i stops a distance of $d_i \cdot \varepsilon \cdot 2/\tau \cdot \left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right|$ away from P_i. By definition is $\varepsilon < 0.5$ and $\left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right| \leq \tau/2$. It follows $d_i \cdot \varepsilon \cdot 2/\tau \cdot \left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right| \leq d_i \cdot \varepsilon < d_i/2$. r_i will move onto a point closer to P_i than to P_k.
- Case r_i is between P_i and P_k: r_i will move onto a point closer to P_i than to P_k because $d_i \leq \left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right|$ which is in this case less than the distance between P_i and P_k.
- Case P_i is between R_i and P_k: r_i will move to a point between its current position and P_i, this is naturally closer to P_i than to P_k.

In all cases, $\text{target}_i^{\mathcal{P}^{cl}}(t)$ is closer to P_i than to P_k and analogously, $\text{target}_k^{\mathcal{P}^{cl}}(t)$ is closer to P_k than to P_i. Hence, $\text{target}_i^{\mathcal{P}^{cl}}(t) \neq \text{target}_k^{\mathcal{P}^{cl}}(t)$.

■

Lemma 2.9 — restated. If the target points of robots are the same in $\mathcal{P}^\tau$ they are different in $\mathcal{P}^{cl}$.

Proof. More formally, we prove the following statement: Let r_i and r_k be two robots with $\text{target}_i^{\mathcal{P}^\tau}(t) = \text{target}_k^{\mathcal{P}^\tau}(t)$. It follows that $\text{target}_i^{\mathcal{P}^{cl}}(t) \neq \text{target}_k^{\mathcal{P}^{cl}}(t)$.

Let $P = \text{target}_i^{\mathcal{P}^\tau}(t) = \text{target}_k^{\mathcal{P}^\tau}(t)$. A robot moving towards P will stay on the same side of P, and none will reach P. So collisions can solely happen if $\text{collvec}_i^{\mathcal{P}^\tau}(t)$ and $\text{collvec}_k^{\mathcal{P}^\tau}(t)$ are collinear pointing from the same side to P. We consider this case. W.l.o.g., let r_i be closer to P than to r_k. d_i, respectively d_k, is computed by the point in $C_i \setminus \{P\}$, respectively $C_k \setminus \{P\}$, with minimal distance to P (line 3 in Algorithm 1). Let this be $c_i \in C_i$, respectively $c_k \in C_k$. We assume $c_i \neq c_k$. From $c_i \neq c_k$, it follows directly $c_i \notin C_k$ or $c_k \notin C_i$.

- Case $c_i \notin C_k$: r_i and r_k are chosen in a way that $\text{collvec}_i^{\mathcal{P}^\tau}(t) \subset \text{collvec}_k^{\mathcal{P}^\tau}(t)$, it follows c_i is also on $\text{collvec}_k^{\mathcal{P}^\tau}(t)$. c_i is a point on $\text{collvec}_j^{\mathcal{P}^\tau}(t)$ the collision vector of some robot r_j (see Algorithm 2). $\left|\text{collvec}_j^{\mathcal{P}^\tau}(t)\right| + \left|\text{collvec}_k^{\mathcal{P}^\tau}(t)\right| \leq \tau$ is an upper bound for the distance between r_k and r_j. r_j must be in R_k as computed in line 1 of Algorithm 1. Hence, $\text{collvec}_j^{\mathcal{P}^\tau}(t)$ is checked for collisions and c_i must be in C_k.

- Case $c_k \notin C_i$: Similar to the arguments above, $p_i(t)$ is the position of robot r_i, in C_k. The distance of c_k to P is therefore is not larger than the distance $\left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right|$ (otherwise would $p_i(t)$ be nearer to P than the chosen collision point c_k with minimal distance to P). It follows that c_k is also on $\text{collvec}_i^{\mathcal{P}^\tau}(t)$. Analogous to the case above, $c_k \in C_i$.

$c_i = c_k$ and $d_i = d_k$, accordingly. $\left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right| \neq \left|\text{collvec}_k^{\mathcal{P}^\tau}(t)\right|$, otherwise would r_i and r_k be at the same position and a collision has happened earlier. It follows that $d_i \cdot \varepsilon \cdot 2/\tau \cdot \left|\text{collvec}_i^{\mathcal{P}^\tau}(t)\right| \neq d_k \cdot \varepsilon \cdot 2/\tau \cdot \left|\text{collvec}_k^{\mathcal{P}^\tau}(t)\right|$ in every case such that r_i and r_j move to different positions. ■

2.4.6 Semi-synchronous

Observation 2.3 shows, that λ-contracting NEAR-GATHERING protocols also work under the $\mathcal{S}$SYNC scheduler. In $\mathcal{S}$SYNC, robots may be inactive in one round. Nevertheless, in the same way, single intersection points between collision vectors and the positions of other robots are avoided as well. A robot cannot know which robots are active or inactive. However, the protocol is designed so that no robot moves to the current position of any robot. This can be proven analogously to Lemma 2.7 because in line 5 of Algorithm 2 the positions of robots on the collision vector are added to the set of collision points.

Observation 2.4 The protocol $\mathcal{P}^{cl}$ is collision-free under $\mathcal{S}$SYNC scheduler.

The protocol is still λ-contracting and Lemma 2.4 holds true under $\mathcal{S}$SYNC as well, therefore Theorem 4 holds true as well.

3. Near-Gathering with Symmetry Preservation

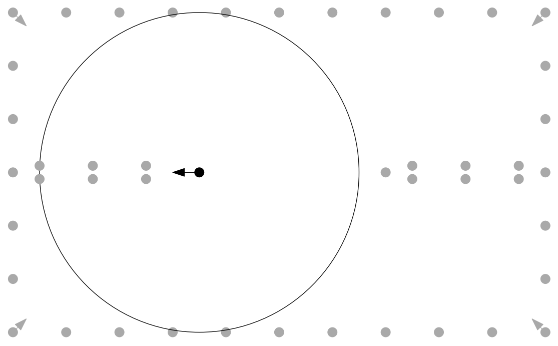
In this chapter, we initiate the systematical study of when and how a swarm of oblivious robots with limited viewing range can perform global tasks like NEAR-GATHERING without increasing the swarm's initial symmetricity. We derive a mathematical framework based on methods from the theory of dynamical systems. In particular, we formulate the following simple but useful theorem (see Section 3.1) that provides sufficient properties for a given swarm protocol to preserve symmetricity. A protocol is represented by its evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ that describes the configuration $\mathbf{z}^+ := F(\mathbf{z})$ after one protocol step on a given configuration $\mathbf{z} \in \mathbb{R}^{2n}$ of n robots in the Euclidean plane. .For the precise definitions of the used terms, like a configuration $\mathbf{z}$'s symmetries and (local) invertibility, we refer to Section 3.1 and Section 1.3.

Theorem 5 — restated. Consider an arbitrary swarm protocol with evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$. Assume that F is (locally) invertible (Definition 3.3). Then, any configuration $\mathbf{z} \in \mathbb{R}^{2n}$ and its successor configuration $\mathbf{z}^+ := F(\mathbf{z})$ have the same symmetricity $\operatorname{sym}(\mathbf{z}^+) = \operatorname{sym}(\mathbf{z})$.

The framework of dynamical systems provides a clean mathematical basis to formulate the symmetries of a given configuration and how they are affected by a protocol step. To prove the usefulness of our framework, we provide two example protocols that, under certain conditions, achieve a NEAR-GATHERING without increasing the swarms symmetricity.

The first protocol (see Section 3.2) is a variant of the GO-TO-THE-AVERAGE protocol. This protocol is known to *not* always preserve the swarm's initial connectivity (see Figure 3.1b for an example), but if it does, it leads to NEAR-GATHERING (Lemma 3.7). Our framework easily implies that it preserves the swarm's initial symmetry (Theorem 6).

The second protocol (see Section 3.3) is an adaption of the GO-TO-THE-MIDDLE strategy and called WAVE-PROTOCOL. It restricts movements to robots close to the swarm's boundary and coordinates the movement of nearby robots to ensure that no symmetries are created. While WAVE-PROTOCOL always preserves connectivity and leads to NEAR-GATHERING (Lemma 3.17), it is not always symmetry preserving (see Figure 3.1a for an example). A proof is provided that its evolution function is locally


(a) Example with hole. Wave-Protocol: Near-Gathering but introduces new symmetries.[1] Go-To-The-Average: Near-Gathering with symmetry preservation.[2]

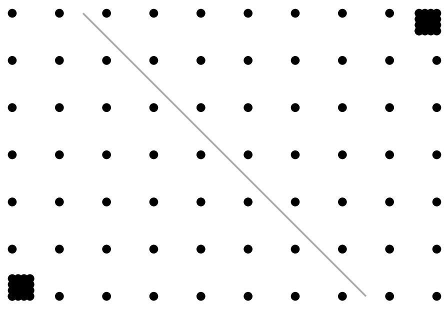

(b) Example with convex boundary and without hole. Wave-Protocol: Near-Gathering with symmetry preservation. Go-To-The-Average: Keeps symmetry but looses connectivity (swarm splits at the gray line)[3].

Figure 3.1: Visualization of example configurations where either protocol fails.

invertible (and, thus, the protocol symmetry preserving) for configurations that contain no "holes" and are convex (these requirements are formalized in Section 3.3). Note, that the Wave-Protocol is inferior to the λ-contracting Near-Gathering protocols from Chapter 2 when symmetry preservation is not relevant (see Section 3.3.4). The content of this chapter was first published in

Symmetry Preservation in Swarms of Oblivious Robots with Limited Visibility [36]
Raphael Gerlach, Sören von der Gracht, Christopher Hahn, Jonas Harbig and Peter Kling
Conference on Priciples of Distributed Systems (OPODIS) 2024

The application of methods from dynamical systems is largely contributed by the co-authors. This mainly concerns the results in Sections 3.1 and 3.2.

Outline. This chapter is outlined as follows: A formal description of the problem is given and a sufficient condition for the preservation of symmetries based on the theory of dynamical systems formulated (Section 3.1). The two protocols are presented and analyzed in Section 3.2 and Section 3.3.

3.1 Preserving Symmetries via Local Invertibility

This section introduces some formal notation and definitions we use throughout the rest of the chapter. In particular, we use some tools from the theory of dynamical systems to formulate sufficient conditions for swarm protocols that preserve symmetries.

As explained in the introduction, we seek Near-Gathering strategies that do not increase the swarm's *symmetricity* (because of the *symmetricity condition* for pattern formation, see [34, Theorem 1]). The symmetricity measures the rotational symmetries of a finite set $P \subseteq \mathbb{R}^2$ (in our case the set of robot positions), see Definition 1.2. Since

[1]The black robot does not observe robots on the outer rectangle (circle depicts viewing range). It assumes it is a boundary-robot and moves accordingly. All other robots in the inside of the rectangle do not move, because they see enough robots on the outer rectangle to detect, that they are inner-robots.

[2]We simulated this example with the shown viewing range.

[3]The distance between grid points is $1/\sqrt{2}$ and the viewing range $2+\sqrt{2}$. We simulated it with 400 robots at each cluster.

here we consider the configuration typically in the global coordinate system of the external observer (which can be chosen arbitrarily), we assume (without loss of generality) that the swarm's center c is the origin.

We now formalize the notion of a *symmetry* in such a way that we can apply the theory of dynamical systems to argue how a protocol's evolution function influences those symmetries. For a swarm of symmetricity $m > 1$ there are exactly m rotations $\rho\colon \mathbb{R}^2 \to \mathbb{R}^2$ around the origin (center) under which the *set* of robot positions is invariant (i.e., $\{\rho(z_1), \dots, \rho(z_n)\} = \{z_1, \dots, z_n\}$). We represent configurations as tuples instead of sets, which is more typical in the context of dynamical systems. Thus, we define a symmetry of a configuration $\mathbf{z} \in (\mathbb{R}^2)^n$ as follows (using a permutation to "relabel" the tuple suitably after the rotation).

Definition 3.1 — Symmetry of a Configuration. Consider a rotation $\rho\colon \mathbb{R}^2 \to \mathbb{R}^2$ centered at the origin and a configuration $\mathbf{z} = (z_1, \dots, z_n)^{\mathrm{T}}$. Then ρ is a *symmetry of the configuration* $\mathbf{z}$ if and only if there exists a permutation $\kappa\colon \{1, \dots, n\} \to \{1, \dots, n\}$ such that $\rho(z_i) = z_{\kappa(i)}$ for all $i \in \{1, \dots, n\}$.

To lift the rotation ρ and permutation κ from Definition 3.1 to the entire configuration $\mathbf{z} \in (\mathbb{R}^2)^n \equiv \mathbb{R}^{2n}$, we use the following block matrices M_ρ and M_κ ($n \times n$ matrices with entries in $\mathbb{R}^{2\times 2}$) implied by ρ and κ:

$$M_\rho = \begin{pmatrix} \rho & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \rho & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \rho \end{pmatrix} \quad \text{and} \quad (M_\kappa)_{ij} = \begin{cases} \mathbf{1}_2, & \text{if } \kappa(i) = j \\ \mathbf{0}, & \text{otherwise.} \end{cases} \tag{3.1}$$

Here, $\mathbf{1}_l$ denotes for the $l \times l$ identity matrix and $\mathbf{0}$ the zero matrix of suitable dimensions. Without further mention, we identify both matrices with their $\mathbb{R}^{2n\times 2n}$ counterparts.

The matrices above allows us to reformulate the condition from Definition 3.1 as $M_\rho \mathbf{z} = M_\kappa \mathbf{z}$. Since M_κ is furthermore invertible with $M_\kappa^{-1} = M_{\kappa^{-1}}$ (and the inverse κ^{-1} is also a permutation), we can reformulate Definition 3.1:

Definition 3.2 — Symmetry of a Configuration (alternative). Consider a configuration $\mathbf{z} \in \mathbb{R}^{2n}$ and a rotation $\rho\colon \mathbb{R}^2 \to \mathbb{R}^2$, both centered at the origin. Then ρ is a *symmetry of the configuration* $\mathbf{z}$ if and only if there exists a permutation $\kappa\colon \{1, \dots, n\} \to \{1, \dots, n\}$ such that $M_\kappa M_\rho \mathbf{z} = \mathbf{z}$.

We denote the set of all *potential symmetries* as

$$G = \{ M_\kappa M_\rho \mid \kappa\colon \{1, \dots, n\} \to \{1, \dots, n\} \text{permutation}, \rho\colon \mathbb{R}^2 \to \mathbb{R}^2 \text{ rotation} \}. \tag{3.2}$$

and the subset of *(actual) symmetries* of a configuration $\mathbf{z}$ as

$$G_{\mathbf{z}} = \{ M_\kappa M_\rho \in G \mid M_\kappa M_\rho \mathbf{z} = \mathbf{z} \}. \tag{3.3}$$

With this formalization at hand, we can study how a protocol (via its evolution function) influences a configurations actual symmetries. In fact, the classical theory of equivariant dynamics [15, 38] immediately yields that a protocol can never cause the *loss* of symmetries.[5] Our Theorem 5 states that if the evolution function F is additionally invertible, then we also cannot *gain* symmetries. Its proof is given below in this section.

[5]For example, robots forming an identical n-gon have identical views and, thus, perform the same local calculations. Thus, the swarm would be forever trapped in a, possibly scaled, n-gon formation.

Note that the assumption of invertibility is required on the level of the configuration. This means there exists a function $F^{-1}\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ such that $F(F^{-1}(\mathbf{z})) = F^{-1}(F(\mathbf{z})) = \mathbf{z}$ for all $\mathbf{z} \in \mathbb{R}^{2n}$. In general, F^{-1} is not the evolution function of a distributed protocol. In particular, a robot *does not* have to be able to determine its previous position based on its local information (f is not invertible). Informally speaking, from our perspective as an external observer we must always be able to determine the swarm's configuration in the previous round. The term "locally" in Theorem 5 is a mathematical characterization of the inverse function F^{-1}. It makes the statement stronger, since the inverse function needs to be defined only locally, meaning for all configurations that are sufficiently similar to a given one. This is, for example, central for the analysis of the averaging strategy from Section 3.2.

Definition 3.3 — (Local) invertibility. An evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is called *invertible* if there exists an *inverse function* $F^{-1}\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ such that $F(F^{-1}(\mathbf{z})) = F^{-1}(F(\mathbf{z})) = \mathbf{z}$ for all $\mathbf{z} \in \mathbb{R}^{2n}$. The evolution function is called *locally invertible* if for any configuration $\mathbf{z}$ there are open subsets $U_\mathbf{z}$ and $V_{F(\mathbf{z})}$ containing $\mathbf{z}$ and $F(\mathbf{z})$, respectively, such that $F_\mathbf{z}\colon U_\mathbf{z} \to V_{F(\mathbf{z})}$ is invertible, i.e., there exists a *local inverse function* $F_\mathbf{z}^{-1}\colon V_\mathbf{z} \to U_{F(\mathbf{z})})$ such that $F_\mathbf{z}(F_\mathbf{z}^{-1}(\mathbf{y})) = \mathbf{y}$ for all $\mathbf{y} \in U_{F(\mathbf{z})}$ and $F_\mathbf{z}^{-1}(F_\mathbf{z}(\mathbf{y})) = \mathbf{y}$ for all $\mathbf{y} \in U_\mathbf{z}$.

Moreover, it is important to emphasize that Theorem 5 does not require any regularity of the evolution function F. In particular, it remains true even if F is non-continuous, which will be essential for the protocol presented in Section 3.3 below.

3.1.1 Symmetries of a protocol

We begin by investigating the interplay of the symmetries defined above, and the dynamics induced by a protocol. The robots do not know the global coordinate system that we choose as the external observer. Hence, their computation and movement is insensitive to rotations of the global coordinates in the sense that collectively rotating all robots positions by ρ causes each robot to move to a rotated target point in each round.

R In principle, the same holds true for translations of the global coordinate systems. However, since we fixed the origin, we typically omit these transformations without loss of generality.

On the other hand, recall that the robots are indistinguishable. That means if a robot observes another robot in a certain position, it does not know which label this robot has. More precisely, the computations and movement of every single robot depend on the set of the other robots' positions rather than their ordered tuple. Once again, this can be recast by stating that computation and movement are insensitive to arbitrary permutations of all robots positions.

Summarizing these observations and reformulating them in terms of the function governing the dynamics of all robots f, we obtain

Lemma 3.1 Let $\eta \in \mathbb{R}^2$ and $\mathbf{z} = (z_1, \dots, z_n)^\mathrm{T} \in \mathbb{R}^{2n}$ be arbitrary. The function governing the dynamics of all robots f has the following *symmetry properties*:

(i) $f(\rho\eta; \rho z_1, \dots, \rho z_n) = \rho f(\eta; \mathbf{z})$ for all rotations $\rho\colon \mathbb{R}^2 \to \mathbb{R}^2$;

(ii) $f(\eta; z_{\kappa(1)}, \dots, \rho z_{\kappa(n)}) = f(\eta; \mathbf{z})$ for all permutations $\kappa\colon \{1, \dots, n\} \to \{1, \dots, n\}$.

These can be restated using matrices (3.1) as

(i) $f(\rho\eta; M_\rho \mathbf{z}) = \rho f(\eta; \mathbf{z})$ for all rotations $\rho\colon \mathbb{R}^2 \to \mathbb{R}^2$;
(ii) $f(\eta; M_\kappa \mathbf{z}) = f(\eta; \mathbf{z})$ for all permutations $\kappa\colon \{1,\dots,n\} \to \{1,\dots,n\}$.

From the global perspective, considering the collective evolution of the entire formation, these symmetry properties imply that the evolution is insensitive to arbitrary rotations and arbitrary permutations of the robots. More precisely, it does not matter if first all robots compute/move and then rotate/permute or do it the other way around. More precisely, we may even replace rotate/permute by rotate and permute which gives us the combined transformations in G.

Lemma 3.2 The evolution function F is symmetric—or *equivariant*—with respect to all potential symmetries of formations $M_\kappa M_\rho \in G$:

$$F \circ (M_\kappa M_\rho) = (M_\kappa M_\rho) \circ F. \tag{3.4}$$

Proof. We claim that it suffices to prove that F is equivariant as in (3.4) with respect to M_κ for all κ and M_ρ for all ρ individually to prove the statement. In fact, if this holds true we immediately see

$$\begin{aligned} F \circ (M_\kappa M_\rho) &= F \circ (M_\kappa \circ M_\rho) \\ &= (F \circ M_\kappa) \circ M_\rho \\ &= (M_\kappa \circ F) \circ M_\rho \\ &= M_\kappa \circ (F \circ M_\rho) \\ &= M_\kappa \circ (M_\rho \circ F) \\ &= (M_\kappa M_\rho) \circ F. \end{aligned}$$

Hence, we prove the claim using the symmetry properties in Lemma 3.1. To that end let $\mathbf{z} = (z_1, \dots, z_n)^{\mathrm{T}} \in \mathbb{R}^{2n}$ be an arbitrary point in configuration space, $\kappa\colon \{1,\dots,n\} \to \{1,\dots,n\}$ an arbitrary permutation, and $\rho\colon \mathbb{R}^2 \to \mathbb{R}^2$ an arbitrary rotation. Then, we compute

$$\begin{aligned} (F \circ M_\kappa)(\mathbf{z}) &= F(M_\kappa \mathbf{z}) \\ &= \begin{pmatrix} f((M_\kappa \mathbf{z})_1, M_\kappa \mathbf{z}) \\ \vdots \\ f((M_\kappa \mathbf{z})_n, M_\kappa \mathbf{z}) \end{pmatrix} \\ &= \begin{pmatrix} f(z_{\kappa(1)}, M_\kappa \mathbf{z}) \\ \vdots \\ f(z_{\kappa(n)}, M_\kappa \mathbf{z}) \end{pmatrix} \\ &= \begin{pmatrix} f(z_{\kappa(1)}, \mathbf{z}) \\ \vdots \\ f(z_{\kappa(n)}, \mathbf{z}) \end{pmatrix} \\ &= \begin{pmatrix} F(\mathbf{z})_{\kappa(1)} \\ \vdots \\ F(\mathbf{z})_{\kappa(n)} \end{pmatrix} \\ &= M_\kappa F(\mathbf{z}) \\ &= (M_\kappa \circ F)(\mathbf{z}), \end{aligned} \tag{3.5}$$

where the fourth equality holds due to Lemma 3.1. Similarly, we obtain

$$\begin{aligned}(F\circ M_\rho)(\mathbf{z}) &= F(M_\rho\mathbf{z})\\ &= \begin{pmatrix} f((M_\rho\mathbf{z})_1, M_\rho\mathbf{z})\\ \vdots \\ f((M_\rho\mathbf{z})_n, M_\rho\mathbf{z})\end{pmatrix}\\ &= \begin{pmatrix} f(\rho z_1, M_\rho\mathbf{z})\\ \vdots \\ f(\rho z_n, M_\rho\mathbf{z})\end{pmatrix}\\ &= \begin{pmatrix} \rho f(z_1, \mathbf{z})\\ \vdots \\ \rho f(z_n, \mathbf{z})\end{pmatrix}\\ &= M_\rho F(\mathbf{z})\\ &= (M_\rho F)(\mathbf{z})\end{aligned} \tag{3.6}$$

again using Lemma 3.1. This completes the proof of the claim. ■

By the previous lemma, F commutes with all elements of G. Hence, whenever we refer to an arbitrary symmetry without the need to specify rotation and permutation separately, we use $M, M', \ldots \in G$ from now on.

3.1.2 Proof of Theorem 5

Theorem 5 — restated. Consider an arbitrary swarm protocol with evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$. Assume that F is (locally) invertible (Definition 3.3). Then, any configuration $\mathbf{z} \in \mathbb{R}^{2n}$ and its successor configuration $\mathbf{z}^+ := F(\mathbf{z})$ have the same symmetricity $\mathrm{sym}(\mathbf{z}^+) = \mathrm{sym}(\mathbf{z})$.

Lemma 3.2 places our mathematical framework in the context of *equivariant dynamics*, for which there exists a well developed theory to investigate the interplay of dynamics and symmetries (e.g. [15, 38]). It allows us to prove the theorem. We do so in the following three lemmas, which combined give the statement of the theorem.

Lemma 3.3 Consider the dynamics of a configuration according to an arbitrary protocol (1.2). Then the configuration after one round cannot have fewer symmetries than the initial one: $G_{\mathbf{z}} \subset G_{\mathbf{z}^+}$.

Proof. Let $\mathbf{z} \in \mathbb{R}^{2n}$ be some arbitrary configuration and $M \in G_{\mathbf{z}}$. Consider the evolution $\mathbf{z}^+ = F(\mathbf{z})$ in one round. Then

$$M\mathbf{z}^+ = MF(\mathbf{z}) = F(M\mathbf{z}) = F(\mathbf{z}) = \mathbf{z}^+,$$

where we have exploited the symmetry of F (Lemma 3.2) as well as the fact that M leaves $\mathbf{z}$ unchanged. In particular, this implies $M \in G_{\mathbf{z}^+}$ proving the statement. ■

In a very similar manner we may prove that a configuration cannot gain any symmetries during the temporal evolution. This statement, however, is only true in general if the evolution function F is invertible.

Lemma 3.4 Consider the dynamics of a configuration according to an arbitrary protocol (1.2). Assume that the evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is invertible. Then the configuration after one round cannot have more symmetries than the initial one. In particular, $G_{\mathbf{z}} = G_{\mathbf{z}^+}$.

Proof. The setup for the proof is the same as in the previous. Let $\mathbf{z} \in \mathbb{R}^{2n}$ be some arbitrary configuration. Consider the evolution $\mathbf{z}^+ = F(\mathbf{z})$ in one round. By assumption, F is invertible and we may restate $\mathbf{z} = F^{-1}(\mathbf{z}^+)$. It can readily be seen that the inverse F^{-1} has the same symmetry properties as F:

$$F \circ M = M \circ F \iff F \circ M \circ F^{-1} = M \iff M \circ F^{-1} = F^{-1} \circ M$$

for any $M \in G$.

In particular, for $M \in G_{\mathbf{z}^+}$ we may apply Lemma 3.3 to F^{-1} to obtain $M \in G_{\mathbf{z}}$. This implies $G_{\mathbf{z}^+} \subset G_{\mathbf{z}}$, which in combination with Lemma 3.3 proves the claim. ■

The previous lemma requires the existence of a *global* inverse F^{-1} to F. However, the weaker notion of *local* invertibility is sufficiently strong to draw the conclusions of Lemma 3.4.

Lemma 3.5 Consider the dynamics of a configuration according to an arbitrary protocol (1.2). Assume that the evolution function $F\colon \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is *locally* invertible. Then, the configuration after one round cannot have more symmetries than the initial one. In particular, $G_{\mathbf{z}} = G_{\mathbf{z}^+}$.

Proof. As we have a local inverse for every configuration $\mathbf{z} \in \mathbb{R}^{2n}$, we may compare $G_{\mathbf{z}}$ and $G_{\mathbf{z}^+}$ using the local inverse as before. To that end, let $\mathbf{z} \in \mathbb{R}^{2n}$ be an arbitrary configuration, $\mathbf{z}^+ = F(\mathbf{z})$, $F_{\mathbf{z}}^{-1}$ the local inverse, and $M \in G_{\mathbf{z}^+}$. Then

$$\begin{aligned}
\mathbf{z} &= F_{\mathbf{z}}^{-1}(\mathbf{z}^+)\\
&= F_{\mathbf{z}}^{-1}(M\mathbf{z}^+)\\
&= F_{\mathbf{z}}^{-1}(MF(F_{\mathbf{z}}^{-1}(\mathbf{z}^+)))\\
&= F_{\mathbf{z}}^{-1}(F(MF_{\mathbf{z}}^{-1}(\mathbf{z}^+)))\\
&= MF_{\mathbf{z}}^{-1}(\mathbf{z}^+) = M\mathbf{z}.
\end{aligned}$$

All applications of the local inverse are well-defined, as M leaves $\mathbf{z}^+$ unchanged. In particular, we have shown that $M \in G_{\mathbf{z}}$ proving the necessary inclusion as in Lemma 3.4. ■

3.2 Preserving Symmetries via Averaging

This section presents a protocol that preserves the symmetries of a configuration (which we prove via Theorem 5). However, it does not always achieve NEAR-GATHERING as certain initial configurations may result in several clusters of NEAR-GATHERINGs that have a mutual distance of ≤ 1. Our proposed protocol ε-GO-TO-THE-AVERAGE does the following in each round:

Algorithm 4 ε-Go-To-The-Average protocol.

```
neighborhood ← positions of visible neighbors (viewing range 1).
$p$ ← weighted average (see below) of neighborhood
Move an $\varepsilon$-fraction towards $p$ for an $\varepsilon \in (0,1)$.
```

The weights of the i-robot at position $\mathbf{z}_i$ for a visible neighbor at position $\mathbf{z}_j$ is derived via a monotonically decreasing *bump function* of their squared distances $X := \|z_i - z_j\|^2$ defined via

$$b(X) = \begin{cases} \exp\left(-\frac{X^2}{1-X^2}\right) & \text{if } X \in [0,1] \\ 0 & \text{if } X > 1. \end{cases} \tag{3.7}$$

whose graph is shown in Figure 3.2.

With the bump function, we can model the computation of local weighted average of a robot in position $z_i \in \mathbb{R}^2$ of a configuration $\mathbf{z} \in \mathbb{R}^{2n}$ as

$$T(z_i;\mathbf{z}) = \frac{1}{n}\sum_{j=1}^{n} b(\|z_i - z_j\|^2)(z_j - z_i). \tag{3.8}$$

We assume that the global information n, i.e., the total number of robots, is known to all robots. This is a reasonable assumption, especially for a protocol that is used as part of Arbitrary-Pattern-Formation, because APF defines that all robots know P with $|P| = n$. Beside the knowledge of n, the protocol does not need any global information. Note that the weights (values of the bump function) for robots outside of the viewing range of i are 0 and, hence, do not affect the computation. Therefore the protocol can indeed be executed with limited visibility. The target function of robot z_i is defined by

$$z_i^+ = f(z_i;\mathbf{z}) = z_i + \varepsilon \cdot T(z_i;\mathbf{z}) = z_i + \frac{\varepsilon}{n}\sum_{j=1}^{n} b(\|z_i - z_j\|^2)\cdot(z_j - z_i) \tag{3.9}$$

for some fixed $\varepsilon \in (0,1)$. This yields our protocol's evolution function F as specified in Equation (1.2).

The next lemma states that if ε is chosen small enough, F is locally invertible.As a direct consequence of Theorem 5, this implies that ε-Go-To-The-Average preserves symmetries.

Lemma 3.6 Consider the evolution function F of the ε-Go-To-The-Average protocol for $\varepsilon \leq 1/29$. Then F is locally invertible.

The proof is given in the next subsection (Section 3.2.1). The following theorem can directly be followed by applying Theorem 5.

Theorem 6 — restated. The execution of ε-Go-To-The-Average (Algorithm 4) protocol for $\varepsilon < \frac{1}{29}$ does not change the symmetricity of the swarm.

Unfortunately, as with the standard Go-To-The-Average protocol, we cannot guarantee that connectivity is preserved by this adaptation. Thus, in general we might not

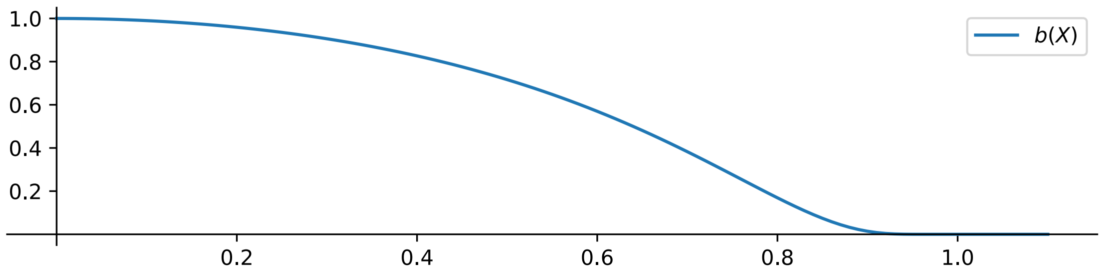


Figure 3.2: Graph of bump function $b(X)$. It decreases monotonically from 1 (for $X = ||z_i - z_j||^2 = 0$; i.e., when j is at the same position as i) to 0 (for $X = 1$; i.e., when j is at the brink of being invisible).

end up in a NEAR-GATHERING, but in several clusters at distance ≥ 1, each of which can be seen as a "separate" NEAR-GATHERINGS. However, if we start in a configuration for which GO-TO-THE-AVERAGE maintains connectivity (like highly regular meshes), we achieve NEAR-GATHERING without increasing the symmetries. This is formulated in the following lemma. Note that it is not possible to simply assume connectivity (distance ≤ 1 between robots) to show that, because the bump function b excludes robots at distance ≥ 1. If the distance between two robots approaches 1, the bump function becomes infinitesimal. This can lead to an arbitrary high number of rounds until robots moved significantly. Therefore, we assume a strongly connected swarm (the UDG with unit distance $1-\delta$ is connected) and state the following lemma.

Lemma 3.7 Let a swarm of robots be and stays strongly connected regarding distance $1-\delta$. After finite many rounds executing ε-GO-TO-THE-AVERAGE NEAR-GATHERING is solved.

Proof. In strongly connected swarm, ε-GO-TO-THE-AVERAGE has properties comparable to a λ-contracting NEAR-GATHERING protocol (Definition 2.3) with

$$\lambda = \frac{\varepsilon}{2n} \cdot \min(b(X), X \in [\delta, 1-\delta]),$$

such that we can adapt the proof of Theorem 2 to show that after finite rounds the global enclosing circle GS decreases it radius R by $\mathcal{O}\left(\frac{(\lambda \cdot \delta)^2}{R}\right)$ (see Section 2.2.2 for the definitions and notation). This leads, eventually, to NEAR-GATHERING.

The value for λ is composed as follows. Concerning the λ-contracting properties we consider a viewing range of $1-\delta$, let z_j be the robot with maximal distance to z_i within range of $1-\delta$. In the worst case, all other $n-2$ robots are on the exact same position as z_i. Then, z_i would move distance $\frac{\varepsilon}{2n} \cdot b(||z_i - z_j||^2)$ towards z_j (see Equation (3.9)).

Consider all robots that are inside $S_{\delta \cdot \lambda}$ (see Section 2.2.2 for the definition). Whenever a robot $r_i \in S_{\delta \cdot \lambda}$ sees another robots in distance $\geq \delta$ analog to Lemma 2.3 can be shown, that it leaves $S_{\delta \cdot \lambda}$. All robots in $S_{\delta \cdot \lambda}$ have mutual visibility and will converge towards one point, if they observe no robots outside $S_{\delta \cdot \lambda}$. At least on of the robots in $S_{\delta \cdot \lambda}$ (w.l.o.g z_j) must observe another robot in distance $[\delta, 1-\delta]$ (w.l.o.g. z_k) because the swarm is by assumption connected regarding distance $1-\delta$. While the other robots converge towards one position, z_j will move (slightly) towards z_k. Eventually, z_j reaches a distance of $\geq \delta$ to the other robots in $S_{\delta \cdot \lambda}$ or the other robots in $S_{\delta \cdot \lambda}$ followed z_j up to the point that they observe z_k in distance $\geq \delta$ or left $S_{\delta \cdot \lambda}$. In all cases, $S_{\delta \cdot \lambda}$ contains no robots after finite many rounds.

■

3.2.1 Invertibility of Averaging

Here we prove Lemma 3.6, namely that our first protocol (Section 3.2) satisfies the assumptions of Theorem 5. In fact, we need to confirm, that the induced evolution function is indeed locally invertible. The main ingredients of the proof are

- the inverse function theorem (e.g., [41]), which states that a continuously differentiable function is locally invertible at every point where its Jacobian is an invertible matrix, and
- the Gershgorin circle theorem [37], which states that all eigenvalues of a matrix $A = (a_{i,j})_{i,j=1}^n$ are contained in the union of the circles $\{z \in \mathbb{C} \mid |z - a_{i,i}| \leq \sum_{j=1, j\neq i}^n |a_{i,j}|\}$.

We first fix some notation. Recall from (3.9) that

$$f(z_i, \mathbf{z}) = z_i + \frac{\varepsilon}{n} \sum_{j=1}^{n} b(\|z_i - z_j\|^2)(z_j - z_i),$$

which is smooth—and thus continuously differentiable in particular—by construction. Furthermore, it is a two-dimensional expression. The collection of these expressions for $i = 1, \ldots, n$ is the evolution function F. To specify the x- and y-directions separately, we denote

$$F(\mathbf{z}) = \begin{pmatrix} f(z_1, \mathbf{z}) \\ \vdots \\ f(z_n, \mathbf{z}) \end{pmatrix} = \begin{pmatrix} F_1^x(\mathbf{z}) \\ F_1^y(\mathbf{z}) \\ \vdots \\ F_n^x(\mathbf{z}) \\ F_n^y(\mathbf{z}) \end{pmatrix}. \tag{3.10}$$

We use the representation (3.10) to compute the Jacobian $DF(\mathbf{z})$ at an arbitrary point $\mathbf{z} = ((x_1, y_1), \ldots, (x_n, y_n)) \in \mathbb{R}^{2n}$. It is of the form

$$DF(\mathbf{z}) = (D_{i,j})_{i,j=1}^n \quad \text{with} \quad D_{i,j} = \begin{pmatrix} \partial_{x_j} F_i^x(\mathbf{z}) & \partial_{y_j} F_i^x(\mathbf{z}) \\ \partial_{x_j} F_i^y(\mathbf{z}) & \partial_{y_j} F_i^y(\mathbf{z}) \end{pmatrix}.$$

The $2n$ Gershgorin circles of the Jacobian $DF(\mathbf{z})$ are centered at $\partial_{x_i} F_i^x(\mathbf{z})$ and $\partial_{y_i} F_i^y(\mathbf{z})$ for $i = 1, \ldots, n$. Their radii are given by

$$R_i^x(\mathbf{z}) = \sum_{j\neq i} |\partial_{x_j} F_i^x(\mathbf{z})| + \sum_{j=1}^{n} |\partial_{y_j} F_i^x(\mathbf{z})| \quad \text{and} \quad R_i^y(\mathbf{z}) = \sum_{j\neq i} |\partial_{y_j} F_i^y(\mathbf{z})| + \sum_{j=1}^{n} |\partial_{x_j} F_i^y(\mathbf{z})|$$

respectively.

We compute the partial derivatives as

$$\partial_{x_j} F_i^x(\mathbf{z}) = \begin{cases} 2\frac{\varepsilon}{n} b'(\|z_i - z_j\|^2)(x_i - x_j)^2 + \frac{\varepsilon}{n} b(\|z_i - z_j\|^2), & j \neq i, \\ 1 - 2\frac{\varepsilon}{n} \sum_{j \neq i} (b'(\|z_i - z_j\|^2)(x_i - x_j)^2 + b(\|z_i - z_j\|^2)), & j = i, \end{cases}$$

$$\partial_{y_j} F_i^x(\mathbf{z}) = \begin{cases} 2\frac{\varepsilon}{n} b'(\|z_i - z_j\|^2)(x_i - x_j)(y_i - y_j), & j \neq i, \\ 2\frac{\varepsilon}{n} \sum_{j \neq i} b'(\|z_i - z_j\|^2)(x_i - x_j)(y_i - y_j), & j = i, \end{cases}$$

$$\partial_{x_j} F_i^y(\mathbf{z}) = \begin{cases} 2\frac{\varepsilon}{n} b'(\|z_i - z_j\|^2)(x_i - x_j)(y_i - y_j), & j \neq i, \\ 2\frac{\varepsilon}{n} \sum_{j \neq i} b'(\|z_i - z_j\|^2)(x_i - x_j)(y_i - y_j), & j = i, \end{cases}$$

$$\partial_{y_j} F_i^y(\mathbf{z}) = \begin{cases} 2\frac{\varepsilon}{n} b'(\|z_i - z_j\|^2)(y_i - y_j)^2 + \frac{\varepsilon}{n} b(\|z_i - z_j\|^2), & j \neq i, \\ 1 - 2\frac{\varepsilon}{n} \sum_{j \neq i} (b'(\|z_i - z_j\|^2)(y_i - y_j)^2 + b(\|z_i - z_j\|^2)), & j = i. \end{cases}$$

Using these expressions we follow

Lemma 3.8 For

$$\varepsilon \leq 1/29 \tag{3.11}$$

one has

$$R_i^x(\mathbf{z}) < |\partial_{x_i} F_i^x(\mathbf{z})| \quad \text{and} \quad R_i^y(\mathbf{z}) < |\partial_{y_i} F_i^y(\mathbf{z})|. \tag{3.12}$$

Proof. We estimate the radii $R_i^x(\mathbf{z})$, one can follow $R_i^y(\mathbf{z})$ analogously.

The partial derivatives $\partial_{x_i} F_i^x(\mathbf{z})$ and $\partial_{y_i} F_i^x(\mathbf{z})$ can be written as follows.

$$\partial_{x_i} F_i^x(\mathbf{z}) = 1 - \sum_{j \neq i} \partial_{x_j} F_i^x(\mathbf{z}) - \sum_{j \neq i} \frac{\varepsilon}{n} b(\|z_i - z_j\|^2) \quad \text{and} \quad \partial_{y_i} F_i^x(\mathbf{z}) = \sum_{j \neq i} \partial_{y_j} F_i^x(\mathbf{z}) \tag{3.13}$$

By applying Equation (3.13) as well as the definition of $R_i^x(\mathbf{z})$ on Equation (3.12) we get the following equivalent notations.

$$\begin{aligned} R_i^x(\mathbf{z}) <& \quad |\partial_{x_i} F_i^x(\mathbf{z})| \Leftrightarrow \\ \sum_{j \neq i} |\partial_{x_j} F_i^x(\mathbf{z})| + \sum_{j \neq i} |\partial_{y_j} F_i^x(\mathbf{z})| + |\partial_{y_i} F_i^x(\mathbf{z})| <& \quad |\partial_{x_i} F_i^x(\mathbf{z})| \Leftrightarrow \\ \sum_{j \neq i} |\partial_{x_j} F_i^x(\mathbf{z})| + \sum_{j \neq i} |\partial_{y_j} F_i^x(\mathbf{z})| + |\sum_{j \neq i} \partial_{y_j} F_i^x(\mathbf{z})| <& \quad |1 - \sum_{j \neq i} \partial_{x_j} F_i^x(\mathbf{z}) - \sum_{j \neq i} \frac{\varepsilon}{n} b(\|z_i - z_j\|^2)| \end{aligned}$$

The inequality will only become stronger, if the right side becomes smaller and the left side larger. This can be done by moving the computation of the absolute values to get the following.

$$\sum_{j \neq i} |\partial_{x_j} F_i^x(\mathbf{z})| + \sum_{j \neq i} |\partial_{y_j} F_i^x(\mathbf{z})| + \sum_{j \neq i} |\partial_{y_j} F_i^x(\mathbf{z})| < 1 - \sum_{j \neq i} |\partial_{x_j} F_i^x(\mathbf{z})| - |\sum_{j \neq i} \frac{\varepsilon}{n} b(\|z_i - z_j\|^2)|$$

This is equivalent with

$$1 > 2\sum_{j\neq i} |\partial_{x_j} F_i^x(\mathbf{z})| + 2\sum_{j\neq i} |\partial_{y_j} F_i^x(\mathbf{z})| + |\sum_{j\neq i} \frac{\varepsilon}{n} b(\|z_i - z_j\|^2)|. \tag{3.14}$$

The maxima and minima of the bump function b (see (3.7)) and its derivative b' can be estimated by plotting it in the interval $X \in [0,1]$. For larger X both function are 0 by definition.

$$b'(X) = -\frac{2X \cdot b(X)}{(1-X^2)^2} \in [0,3] \quad \text{and} \quad b(x) \in [0,1] \tag{3.15}$$

It is directly clear, that $|\sum_{j\neq i} \frac{\varepsilon}{n} b(\|z_i - z_j\|^2)| \leq \frac{\varepsilon}{n}(n-1)$. If $\|z_i - z_j\|^2 > 1$ it follows that $\partial_{x_j} F_i^x(\mathbf{z}) = 0$ and $\partial_{y_j} F_i^x(\mathbf{z}) = 0$ because $b(\|z_i - z_j\|^2) = 0$ and $b'(\|z_i - z_j\|^2) = 0$. If $\|z_i - z_j\|^2 \leq 1$ it follows that $(x_i - x_j) \leq 1$ and $(y_i - y_j) \leq 1$. Therefore,

$$|\partial_{x_j} F_i^x(\mathbf{z})| \leq 2\frac{\varepsilon}{n} |b'(\|z_i - z_j\|^2) + b(\|z_i - z_j\|^2)| \quad \text{and} \quad |\partial_{y_j} F_i^x(\mathbf{z})| \leq 2\frac{\varepsilon}{n} |b'(\|z_i - z_j\|^2)| \tag{3.16}$$

We can estimate

$$|\partial_{x_j} F_i^x(\mathbf{z})| \leq 2\frac{\varepsilon}{n} \cdot 4 \quad \text{and} \quad |\partial_{y_j} F_i^x(\mathbf{z})| \leq 2\frac{\varepsilon}{n} \cdot 3$$

from Equations (3.15) and (3.16). Applying this estimation to Equation (3.14) yields

$$\begin{aligned} 1 > &\quad 2\sum_{j\neq i} 2\frac{\varepsilon}{n} \cdot 4 + 2\sum_{j\neq i} 2\frac{\varepsilon}{n} \cdot 3 + \frac{(n-1)}{n}\varepsilon \\ = &\quad \frac{16 \cdot (n-1)}{n}\varepsilon + \frac{12 \cdot (n-1)}{n}\varepsilon + \frac{(n-1)}{n}\varepsilon \\ = &\quad \frac{29 \cdot (n-1)}{n}\varepsilon. \end{aligned} \tag{3.17}$$

This is the case for

$$\varepsilon \leq 1/29 < \frac{n}{29(n-1)}.$$

■

R The estimate (3.11) is not sharp but sufficient to guarantee the estimate of the radii (3.12).

In particular, (3.12) shows that none of the Gershgorin circles contains 0. Therefore, 0 cannot be an eigenvalue and the Jacobian $DF(\mathbf{z})$ is invertible for any $\mathbf{z} \in \mathbb{R}^{2n}$. By the inverse function theorem, F is therefore locally invertible at every $\mathbf{z} \in \mathbb{R}^{2n}$.

3.3 Preserving Symmetries via Contracting Waves

In this section, we will provide the symmetry preserving WAVE-PROTOCOL that yields NEAR-GATHERING. However, it only works on a subset of initial configurations, and it needs a larger viewing range.

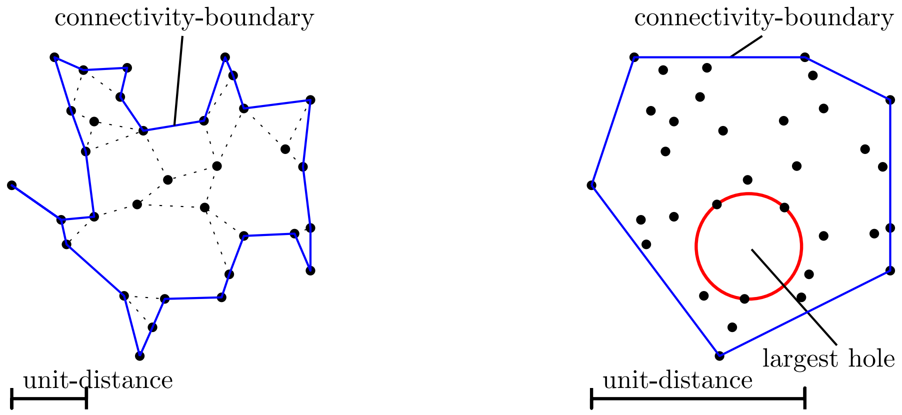


Figure 3.3: **Connectivity-Boundary and Holes.** (left) depicts a configuration with a non-convex Connectivity-Boundary. The small dotted lines are the unit-disc-graph; the Connectivity-Boundary is the outer boundary of this graph. Note that our proof does not apply to this configuration because the Connectivity-Boundary is not convex. (right) depicts the same configuration scaled by factor 0.35, such that the Connectivity-Boundary is convex. It also shows the largest hole. Because the unit-disc-graph in this example is close to a complete graph, we omitted it.

Requirements of the protocol. The Connectivity-Boundary of a configuration is defined in Definition 1.6. A δ*-hole* of a configuration is a circular area inside the Connectivity-Boundary with a diameter of δ that contains no robot. We require that the swarm starts in a configuration that contains no 1-hole and that has a convex[6] Connectivity-Boundary (see Figure 3.3 for an example). The robots have a viewing range of $2+\sqrt{2}$.

Usually, protocols with limited visibility allow initial configurations with a connected unit disc graph. Our requirements only allow a subset of these configurations. However, the subset still allows a high variety of initial configurations of n robots with arbitrary low entropy and a diameter in $\Theta(n)$.

Overview. The robots on the Connectivity-Boundary (we call them *boundary-robots*) will perform the ε-GO-TO-THE-MIDDLE protocol, where robots move towards the midpoint between their two neighbors. The advantage of this protocol is that it is invertible (and therefore connectivity preserving) and a NEAR-GATHERING protocol. The robots near to the boundary will move with the boundary-robots, one may get the impression they are being pushed to the inside by the boundary like a wave (we call them *wave-robots*). We will carefully construct the way wave-robots move such that this movement is invertible. All other robots (called *inner-robots*) do not move. The boundary will contract, more and more inner-robots will become wave-robots and are further pushed inwards until a NEAR-GATHERING is reached.

We split the description of the protocol in three subsections. In Section 3.3.1 we define the boundary-robots and their protocol. We prove, that boundary-robots will remain a convex set during the execution of their protocol (Lemma 3.10) and that their protocol is invertible (Lemma 3.9). In Section 3.3.2 we define an area around the boundary-robots called *Wave*. All robots in this area are wave-robots. We define their protocol and prove that

[6]Note, that we do not consider it to be strictly convex. It may contain multiple collinear robots.

their movement is invertible assuming the Wave is known. Both protocols are combined for the WAVE-PROTOCOL in Section 3.3.3. We depict the execution of one round in Figure 3.8.

3.3.1 Boundary Movement

Definition 3.4 — Boundary-Robots. Robots that are part of the Connectivity-Boundary are called *boundary-robots*. We denote the boundary robots in round t by $\mathbf{b}^t = (b_0^t, \cdots, b_k^t)$. We assume that robots are enumerated counterclockwise with b_0^t chosen arbitrarily.

We define the following protocol based on GO-TO-THE-MIDDLE [27] for boundary-robots. Afterwards, we prove that it is invertible.

Algorithm 5 Boundary-protocol: ε-GO-TO-THE-MIDDLE

$$\varepsilon\text{-GTM}(b_k^t, \mathbf{b}^t) := \varepsilon \cdot \frac{b_{k-1}^t + b_{k+1}^t}{2} + (1-\varepsilon) \cdot b_k^t$$

R The protocol is in general not executable in our model, because robots cannot decide locally, whether they are boundary-robots. In Lemma 3.13, we prove that it is executable in swarms meeting our requirements. In Section 3.3.4 we argue, how the protocol works in swarms not meeting these requirements.

R If not stated otherwise, we assume that $b_k^{t+1} = \varepsilon\text{-GTM}(b_k^t, \mathbf{b}^t)$. In general, the robots on the Connectivity-Boundary may change during the execution, depending on where other robots move. However, in Lemma 3.12, we prove that the set of boundary-robots does not change during the execution of our protocol.

Lemma 3.9 If $\varepsilon \in [0, 0.5)$, ε-GO-TO-THE-MIDDLE is invertible for a global observer.

Proof. ε-GO-TO-THE-MIDDLE can be described as $\mathbf{z}^{t+1} = F(\mathbf{z}^t)$ with

$$F = \begin{pmatrix} 1-\varepsilon & \frac{\varepsilon}{2} & 0 & 0 & 0 & \cdots & 0 & \frac{\varepsilon}{2} \\ \frac{\varepsilon}{2} & 1-\varepsilon & \frac{\varepsilon}{2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{\varepsilon}{2} & 1-\varepsilon & \frac{\varepsilon}{2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \frac{\varepsilon}{2} & 1-\varepsilon & \frac{\varepsilon}{2} & \cdots & 0 & 0 \\ \cdots & & & & & & & \\ \frac{\varepsilon}{2} & 0 & 0 & 0 & 0 & \cdots & \frac{\varepsilon}{2} & 1-\varepsilon \end{pmatrix}.$$

For $\varepsilon < 0.5$, F is strictly diagonally dominant. By [37], F is invertable. ■

Note, that ε-GTM is only invertible because the neighborhood relation is fixed.

The following two geometric properties show, that ε-GTM decreases the surrounded area and stays convex.

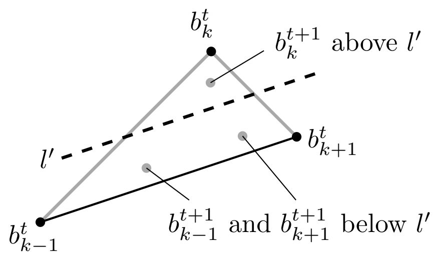


Figure 3.4: Shows that a convex corner in $\texttt{wave}^t$ is convex in $\texttt{wave}^{t+1}$ as well (Lemma 3.10).

Lemma 3.10 If $\mathbf{b}^t$ is convex, $\mathbf{b}^{t+1}$ is convex as well. Let $area(\mathbf{b}^t)$ denote the area enclosed by $\mathbf{b}^t$. Then, $area(\mathbf{b}^{t+1}) \subseteq area(\mathbf{b}^t)$.

Proof. Let us assume $\mathbf{b}^t$ is a convex set. We consider three neighboring robots $b_{k-1}^t, b_k^t, b_{k+1}^t$ in $\mathbf{b}^t$ that are not collinear. They form a triangle. Let τ be the target point of b_k^t executing ε-GTM with $\varepsilon = 1$. The point τ is on the line l between b_{k-1}^t and b_{k+1}^t (black in Figure 3.4). The position of robot b_k^{t+1} is the target point of ε-GTM with $\varepsilon < 0.5$. It can be constructed by moving τ factor ε towards b_k^t which is more than half the way. Therefore, it must lie above l', the parallel line to l move half way towards b_k^t (dotted in Figure 3.4).

Let τ' be the target point of b_{k+1}^t executing ε-GTM with $\varepsilon = 1$. Because $\mathbf{b}^t$ is a convex set, b_{k+2}^t must lie below l. The midpoint between b_k^t and b_{k+2}^t cannot lie above l'. Moving τ' towards b_{k+1}^t cannot move it above l'. Therefore, b_{k+1}^{t+1} and (analogous) b_{k-1}^{t+1} lies below l'. Therefore, the robots b_{k-1}^{t+1}, b_k^{t+1} and b_{k+1}^{t+1} form a convex corner in $\mathbf{b}^{t+1}$.

If $b_{k-1}^t, b_k^t, b_{k+1}^t$ are collinear, they might move onto target points on the same line. But with analog arguments as above, it is easy to see that they cannot move onto positions that form a concave corner.

Therefore, the set $\mathbf{b}^{t+1}$ is still convex.

In the proof above we have shown, that b_k^{t+1} lies inside the triangle $b_{k-1}^t, b_k^t, b_{k+1}^t$. Therefore, the area surrounded by $\mathbf{b}^{t+1}$ is a subset of the area surrounded by $\mathbf{b}^t$.

■

3.3.2 Wave Movement

In this subsection we will define the set of wave-robots and construct their protocol Algorithm 6. In Lemma 3.10 we have shown, that the area enclosed by $\mathbf{b}^{t+1}$ is inside the area enclosed by $\mathbf{b}^t$. The goal of the wave-protocol is, to remove all inner robots that are outside of the area of $\mathbf{b}^{t+1}$ inside that area such that $\mathbf{b}^{t+1}$ is the Connectivity-Boundary in round $t+1$. We first define this area formally. Because this process is similar to a wave front that pushes the robots inwards we use the terminology wave to describe it and call the area *Wave*.

Definition 3.5 — Wave. Let $area(\mathbf{b}^t)$ denote the area enclosed by $\mathbf{b}^t$. We define $\texttt{wave}^t$ as $area(\mathbf{b}^t) \setminus area(\mathbf{b}^{t+1})$ (see Figure 3.5 for a visualization).

Definition 3.6 — Wave-robot. We call robots in $\texttt{wave}^t$ and $\texttt{wave}^{t+1}$ at time t *wave-robots.*

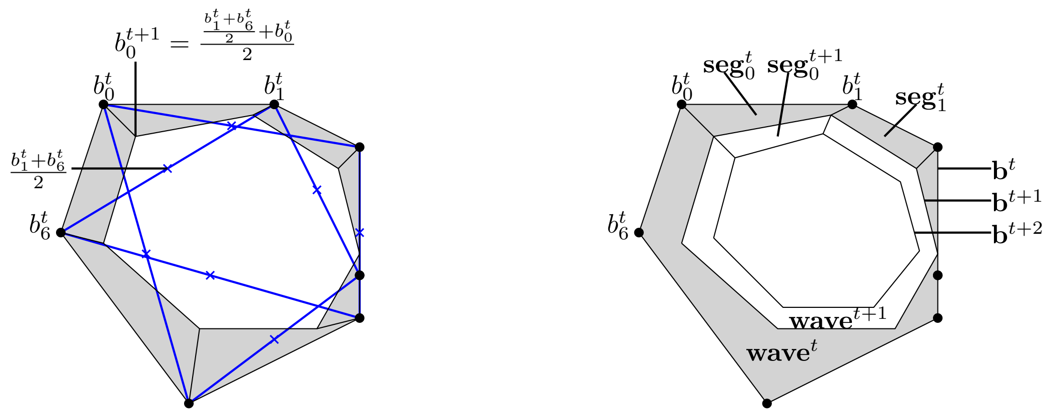


Figure 3.5: (left) shows how the wave segments are computed via the midpoints of neighboring robots (Algorithm 5). For the point b_0^{t+1} the formula is written. (right) shows the naming of the waves (Definition 3.5), its segments (Definition 3.7) and the boundary (Definition 3.4).

Definition 3.7 — Wave-segment. We cut $\texttt{wave}^t$ by cutting from b_k^t to b_k^{t+1} for all $1 \le k \le n$. This yields n *Wave-Segments*. We call the segment with corners $b_k^t, b_{k+1}^t, b_k^{t+1}, b_{k+1}^{t+1}$ the k-th wave-segment of $\texttt{wave}^t$ or $\texttt{seg}_k^t$.

We will design Algorithm 6 such that all robots from $\texttt{seg}_k^t$ move into $\texttt{seg}_k^{t+1}$. The robots in $\texttt{seg}_k^{t+1}$ move inside their segment as well, to prevent collisions with incoming robots.

Preliminary Statements. To use wave-segments as a base for our protocol, we need to make sure that they partition the robots unambiguously. In the following we prove that segments do not overlap and are not twisted. But there are configurations of $\mathbf{b}^t$ (in case of collinear robots in $\mathbf{b}^t$) where the quadrilateral is degenerated, i.e. partly without area or just a line without any area. We prove that a segment in $\texttt{wave}^t$ will not become more degenerated in $\texttt{wave}^{t+1}$.

Corollary 3.1 Let $\mathbf{b}^t$ be a convex set of robots. The segments of $\texttt{wave}^t$ are not twisted quadrilaterals, i.e., two sides do not intersect on a single point, and do not overlap.

Proof. From the arguments of Lemma 3.10 follows, that each position b_k^{t+1} lies in the triangle $b_k^t, \frac{b_k^t+b_{k-1}^t}{2}, \frac{b_k^t+b_{k+1}^t}{2}$. Figure 3.6 shows these triangles with gray dashed lines. It is easy to see, that such quadrilaterals have neither crossing edges nor overlapping areas. ■

Corollary 3.2 We call a quadrilateral where all four corners are collinear *fully degenerated*, and where three corners are collinear *partially degenerated*. Otherwise a quadrilateral has 3 collinear corners and a has 4 collinear corners.

1. If $\texttt{seg}_k^t$ is a non-degenerated quadrilateral, $\texttt{seg}_k^{t+1}$ is also a non-degenerated quadrilateral.
2. If $\texttt{seg}_k^t$ is a partially degenerated segment, $\texttt{seg}_k^{t+1}$ is a non-degenerated segment.

Proof. (1) can be followed from Lemma 3.10.

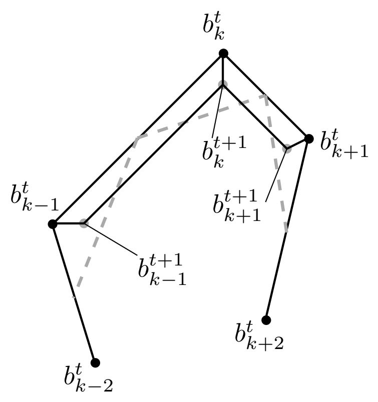


Figure 3.6: Shows that a convex corner in $\mathtt{wave}^t$ is convex in $\mathtt{wave}^{t+1}$ as well (Lemma 3.10).

(2) If b_k^t, b_{k+1}^t and b_{k+2}^t are collinear but b_{k-1}^t is not, $b_{k+1}^{t+1} = b_{k+1}^t$ will not move but $b_k^{t+1} \neq b_k^t$. Therefore, b_k^{t+1}, b_{k+1}^{t+1} and b_{k+2}^{t+1} are not anymore collinear and $\mathtt{seg}_k^{t+1}$ is non-degenerated. ■

Corollary 3.3 Robots in $\mathtt{seg}_k^t$ and $\mathtt{seg}_k^{t+1}$ have a distance of $\leq 1 + \varepsilon^2/2$ to b_k^t and b_{k+1}^t.

Proof. We compute the distances between the corners of $\mathtt{seg}_k^t$ and $\mathtt{seg}_k^{t+1}$.

In the equations below we estimate $|b_k^t - b_{k+1}^t| \leq 1$, $|b_k^t - b_{k+2}^t| \leq 2$ and $\left|b_k^t - b_{k+3}^t\right| \leq 3$.

$$\begin{aligned}
\left|b_k^t - b_{k+1}^{t+1}\right| &= \left|b_k^t - \frac{\varepsilon}{2} b_k^t - \frac{\varepsilon}{2} b_{k+2}^t - (1-\varepsilon) b_{k+1}^t\right| \\
&= \frac{\varepsilon}{2} \left|b_k^t - b_{k+2}^t\right| + (1-\varepsilon)\left|b_k^t + b_{k+1}^t\right| \\
&\leq \varepsilon + (1-\varepsilon) = 1
\end{aligned}$$

Analog is $\left|b_{k+1}^t - b_k^{t+1}\right| \leq 1$. It is clear, that $\left|b_k^t - b_k^{t+1}\right| \leq \varepsilon$ and $\left|b_{k+1}^t - b_{k+1}^{t+1}\right| \leq \varepsilon$.

$$\begin{aligned}
\left|b_k^t - b_{k+1}^{t+2}\right| &= \left|b_k^t - (1-\varepsilon) b_{k+1}^{t+1} - \frac{\varepsilon}{2} b_k^{t+1} - \frac{\varepsilon}{2} b_{k+2}^{t+1}\right| \\
&\leq (1-\varepsilon) \cdot 1 + \frac{\varepsilon}{2} \cdot \varepsilon + \frac{\varepsilon}{2} \left|b_k^t - \frac{\varepsilon}{2} b_{k+1}^t - \frac{\varepsilon}{2} b_{k+3}^t - (1-\varepsilon) b_{k+2}^t\right| \\
&\leq (1-\varepsilon) + \frac{\varepsilon^2}{2} + \frac{\varepsilon}{2}\left(\frac{\varepsilon}{2} + 3\frac{\varepsilon}{2} + 2(1-\varepsilon)\right) \\
&= (1+\varepsilon)(1-\varepsilon) + \frac{3}{2}\varepsilon^2 \\
&= 1 + \frac{\varepsilon^2}{2}
\end{aligned}$$

Analog is $\left|b_{k+1}^t - b_k^{t+2}\right| \leq 1 + \frac{\varepsilon^2}{2}$. It is clear, that $\left|b_k^t - b_k^{t+2}\right| \leq 2\varepsilon$ and $\left|b_{k+1}^t - b_{k+1}^{t+2}\right| \leq 2\varepsilon$.

Any robot inside $\mathtt{seg}_k^t$ or $\mathtt{seg}_k^{t+1}$ as a smaller or equal distance to b_k^t and b_{k+1}^t than the farthest corner of the wave-segments. Therefore, the distance is $\leq 1 + \frac{\varepsilon^2}{2}$. ■

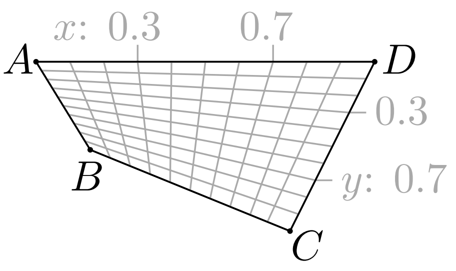


Figure 3.7: Shows a grid inside a segment according to Definition 3.8.

Definition of the Protocol. The protocol uses a bijective mapping from the area $\mathtt{seg}_k^t \cup \mathtt{seg}_k^{t+1}$ onto $\mathtt{seg}_k^{t+1}$. For this mapping we define a normalized two-dimensional coordinate system inside each segment from ranging from $(0,0)$ to $(1,1)$ (Definition 3.8). In rectangular segments these are simple vertical and horizontal lines and the x- and y unit-distance is scaled accordingly, for other shapes the lines are adjusted to follow the boundaries of the shape. Algorithm 6 is a simple mapping between the areas of the segments based on the normalized coordinates. Wave robots from $\mathtt{seg}_k^t$ move inside the outer half of $\mathtt{seg}_k^{t+1}$ while the robots inside $\mathtt{seg}_k^{t+1}$ move into the inner half of $\mathtt{seg}_k^{t+1}$. See Figure 3.8 for an example. In the end we prove that the wave protocol is invertible (Lemma 3.11).

Definition 3.8 — normalized coordinate-system inside a quadrilateral. Let A, B, C, D be the corners of the quadrilateral, we denote the line-segments between the corners by $\overline{AB}$. For $x \in [0,1]$ we define $(x,0) := A + x \cdot (D - A)$ and $(x,1) := B + x \cdot (C - B)$ (in particular $(0,0) = A; (1,0) = B; (1,1) = C$ and $(0,1) = D$). If the quadrilateral is convex, for $y \in [0,1]$ we equally distribute (x,y) on the straight line between $(x,0)$ and $(x,1)$. For non-convex quadrilaterals we assume w.l.o.g. that C is the concave corner. For $x \in [0,1]$ we connect $(x,0)$ and $(x,1)$ with a parallel to $\overline{AB}$ starting at $(x,0)$ and a parallel to $\overline{CD}$ starting at $(x,1)$. For $y \in [0,1]$ we equally distribute (x,y) on this connection. See Figure 3.7 and the figures in Figure 3.8 for an example.

For wave-segment $\mathtt{seg}_k^t$ we define $A = b_k^t, B = b_{k+1}^t, C = b_{k+1}^{t+1}$ and $D = b_k^{t+1}$. To denote the position (x,y) inside $\mathtt{seg}_k^t$ we use the notation $\mathtt{seg}_k^t(x,y)$.

Algorithm 6 Protocol for wave-robots

This protocol gets the positions of boundary-robots $\mathbf{b}^t$ as input. It is used to compute $\mathtt{wave}^t$ and $\mathtt{wave}^{t+1}$. The protocol then determines whether z_i^t is in $\mathtt{seg}_k^t$ or $\mathtt{seg}_k^{t+1}$ for some k and the coordinates x and y according to Definition 3.8.

$$\textsc{Wave-Movement}(z_i^t, \mathbf{b}^t) = \begin{cases} \mathtt{seg}_k^{t+1}(x/2, y), & \text{if } z_i^t = \mathtt{seg}_k^t(x,y) \\ \mathtt{seg}_k^{t+1}(1/2 + x/2, y), & \text{if } z_i^t = \mathtt{seg}_k^{t+1}(x,y) \end{cases}$$

Lemma 3.11 Assuming $\mathbf{b}^t$ and the wave-robots in round t are fixed and known. After executing Algorithm 6 with all wave-robots, we can compute the positions of the wave-robots in round t.

Proof. All robots from $\mathtt{seg}_k^t$ and $\mathtt{seg}_k^{t+1}$ moved inside $\mathtt{seg}_k^{t+1}$ for $0 \le k < |\mathbf{b}^t|$. The segments do not overlap (Corollary 3.1), therefore the movement is unambiguous. If $\texttt{seg}_k^t$ is non-degenerated, then $\texttt{seg}_k^{t+1}$ is also non-degenerated (Corollary 3.2). In both segments, the coordinates are unambiguous. Therefore, Definition 3.8 yields a bijective mapping. If $\texttt{seg}_k^t$ is partly degenerated, there may exist $(x,y) \neq (x',y')$ with $\texttt{seg}_k^t(x,y) = \texttt{seg}_k^t(x',y')$. But $\texttt{seg}_k^{t+1}$ is non-degenerated (Corollary 3.2), therefore no two robots move onto the same positions in $\texttt{seg}_k^{t+1}$. Therefore, the origins for all robots is unambiguous. If $\texttt{seg}_k^{t+1}(x,y)$ is fully-degenerated (is only a line), it follows from Corollary 3.2 that $\texttt{seg}_k^t(x,y)$ must be fully-degenerated as well. But because both segments are only a line without area in this case, the mapping is bijective. ■

3.3.3 Main Protocol

We first define *inner-robots*, that are the remaining robots beside boundary- and wave-robots. Afterwards we state the main protocol formally and prove its correctness as well as that it is symmetry preserving.

Definition 3.9 — inner-robots. We call robots in $\texttt{wave}^q, q > t+1$ at time t *inner-robots*.

Algorithm 7 WAVE-PROTOCOL

Based on $\mathbf{z}^t$ the positions of boundary-robots $\mathbf{b}^t$ can be observed. $\mathbf{b}^t$ is used to compute $\texttt{wave}^t$ and $\texttt{wave}^{t+1}$ to determine wave-robots and inner-robots.

$$f(z_k^t, \mathbf{z}^t) = \begin{cases} \varepsilon\text{-}\textsc{Gtm}(z_k^t, \mathbf{b}^t) & \text{if } z_k^t \text{ is a boundary-robot (Definition 3.4),} \\ \textsc{Wave-Movement}(z_k^t, \mathbf{b}^t) & \text{if } z_k^t \text{ is a wave-robot (Definition 3.6),} \\ z_k^t & \text{if } z_k^t \text{ is an inner-robot (Definition 3.9).} \end{cases}$$

See Figure 3.8 for an example.

R To execute Algorithm 7 as it is written, robots must observe $\mathbf{b}^t$. They are not able to observe $\mathbf{b}^t$ fully because of the limited visibility in out model. But we prove in Lemma 3.13 that they are able to observe the locally relevant part of $\mathbf{b}^t$ to compute Algorithm 7 locally.

Lemma 3.12 If $\mathbf{b}^t$ is convex and the Connectivity-Boundary of a swarm without 2.24-holes that executes Algorithm 7 than

$$\mathbf{b}^{t+1} = \varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t)$$

Proof. From Lemma 3.10 we know, that $\varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t)$ is convex. Neighboring positions in $\varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t)$ have distance ≤ 1. Therefore, $\varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t)$ would be the Connectivity-Boundary of the new configuration, if all other robots are within $area(\varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t))$. We defined $\texttt{wave}^t$ as $area(\mathbf{b}^t) \setminus area(\varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t))$. Algorithm 6 is designed such that all robots from $\texttt{wave}^t$ move into $\texttt{wave}^{t+1}$, therefore moving inside $area(\varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t))$. Only robots from coordinates $\texttt{seg}_k^t(x,0)$ move onto coordinates $\texttt{seg}_k^{t+1}(x,0)$ (see Algorithm 6). On coordinates $\texttt{seg}_k^t(x,0)$ are by definition only boundary-robots. Therefore, $\varepsilon\text{-}\textsc{Gtm}(\mathbf{b}^t)$ is the Connectivity-Boundary of the swarm in round $t+1$. ■

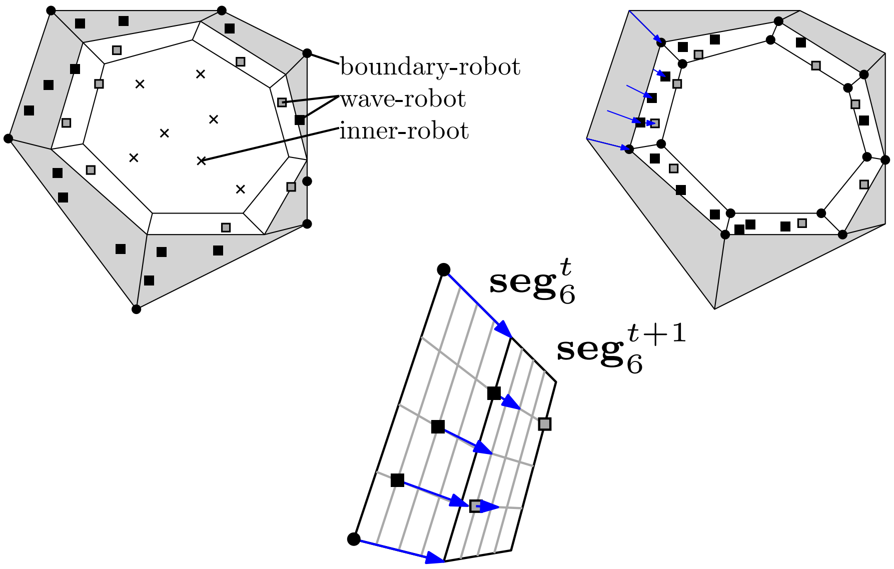


Figure 3.8: (left) shows the partition of robots in boundary-robot (Definition 3.4), wave-robots (Definition 3.6) and inner-robots (Definition 3.9). (right) shows the new robot positions after execution Algorithm 7. The figure in the middle shows a close-up on $\mathtt{seg}_6^t$ and $\mathtt{seg}_6^{t+1}$ that demonstrates the computation of Algorithm 6, based on the coordinate-system defined in Definition 3.8.

Lemma 3.13 In a configuration with a convex Connectivity-Boundary and no 2.24-holes, Algorithm 7 **is executable** for $\mathcal{O}$BLOT-robots with a viewing range of $2+\sqrt{2}$.

Proof. **Decide robot state locally (boundary, wave, inner).** With no 2.24-holes, a robot is on the Connectivity-Boundary if it is on the border of the convex hull of its 2.24-surrounding. This can be determined with a viewing range of 2.24. A robot r inside $\mathtt{seg}_k^t$ or $\mathtt{seg}_k^{t+1}$ has a distance $\leq 1+\varepsilon^2/2 < 1.12$ to b_k^t and b_{k+1}^t (see Corollary 3.3). Therefore, r with a viewing range of $1.12+2.24 < 2+\sqrt{2}$ can determine, whether the robot on b_k^t is a boundary-robot. To compute $\mathtt{seg}_k^t$ and $\mathtt{seg}_k^{t+1}$ robots on $b_{k-2}^t, \cdots, b_{k+3}^t$ must be observed. To b_k^t and b_{k+1}^t these have a maximal distance of up to 2. Because the robot on b_k^t could already be identified, it is known where the outside of the Connectivity-Boundary is. We know that the Connectivity-Boundary is connected with a distance ≤ 1. Therefore, the 1-surrounding around a known boundary-robot is sufficient to identify its next neighbor along the boundary. With a viewing range of $2+\sqrt{2}$ the 1-surrounding of b_k^t and b_{k+1}^t as well as from b_{k-1}^t and b_{k+2}^t is observable. This allows to identify robots on $b_{k-2}^t, \cdots, b_{k+3}^t$. Therefor, each robot in $\mathtt{seg}_k^t$ and $\mathtt{seg}_k^{t+1}$ can identify, that it is in the mentioned segment (wave-robots). All robots r not in these segments can either not find sufficient many boundary-robots or can compute that they are not in segments of $\mathtt{wave}^t$ or $\mathtt{wave}^{t+1}$ adjacent to the observed boundary-robots. Both is sufficient to decide, that r is an inner robot.

Compute the protocol locally. For boundary robots (ε-GO-TO-THE-MIDDLE) and inner robots (do not move) this is trivial. The wave-segments are not overlapping. Target positions are computed based on the segment the robot is in. Therefore, robots not on the borders of a segment can unambiguously compute their target positions. Robots on the

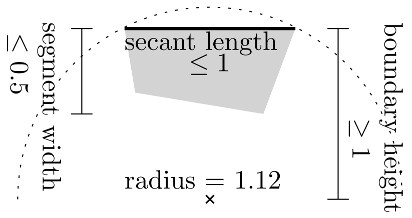


Figure 3.9: Distance between $\mathtt{seg}_k^t$ (gray) and the midpoint of a 2.24-hole.

border of a segment will compute for both segments the exact target position, therefor it is unambiguous as well. The movement distance is ≤ 1, therefore the computed move can be executed. ■

Lemma 3.14 In a configuration with a convex Connectivity-Boundary, Algorithm 7 **does not lead to collisions**.

Proof. We consider round t. Robot within $\mathtt{wave}^{t'}, t' \geq t+2$ (inner-robots) cannot have collisions with each other, because they do not move. They can also have no collisions with other robots, because boundary- and wave-robots move within $\mathtt{wave}^t$ and $\mathtt{wave}^{t+1}$. Boundary robots are essentially wave robots with the special case that they are on coordinates $\mathtt{seg}_k^t(x,0)$. From the proof of Lemma 3.11 follows, that Algorithm 6 is collision-free. ■

Lemma 3.15 In a configuration with convex Connectivity-Boundary and initially no 1-holes, Algorithm 7 **does not create** 2.24**-holes**.

Proof. All robots not inside $\mathtt{wave}^t$ have been inner robots until now and have not moved at all. Therefore, holes that have no overlap with $\mathtt{wave}^t$ must have existed initially and can only have diameter < 1. Let us consider a 2.24-hole that overlaps with $\mathtt{wave}^t$. See the construction of $\mathtt{seg}_k^t$ that overlaps with a 2.24-hole in Figure 3.9. $\mathtt{seg}_k^t$ cannot lie completely within the hole, because that would mean the boundary robots are inside the hole. But the boundary of $\mathtt{seg}_k^t$ can be a secant with length ≤ 1 of the hole. The height of such a secant of a circle with radius 1.12 is ≥ 1. A wave-segment can have a maximal width of 0.5, therefore all points of $\mathtt{seg}_k^t$ have a distance ≥ 0.5 to the midpoint of the hole. Therefore, 1-hole with the same midpoint does not overlap with $\mathtt{wave}^t$ and must have existed initially. ■

Lemma 3.16 In a configuration with a convex Connectivity-Boundary, Algorithm 7 **is locally invertible**.

Proof. Let us assume $\mathbf{b}^t$ is convex and the Connectivity-Boundary of the swarm in round t. For the initial configuration this is true by definition. We will show, that we can compute the swarm in round t from round $t+1$. From Lemma 3.10 we know that $\mathbf{b}^{t+1}$ is convex. From Lemma 3.12 we know, that $\mathbf{b}^{t+1}$ is the Connectivity-Boundary in round $t+1$. The Connectivity-Boundary can easily be identified by a global observer, therefore $\mathbf{b}^{t+1}$ is known. By Lemma 3.9 we know that ε-GTM is invertible. Therefore, we can compute $\mathbf{b}^t$ and $\mathtt{wave}^t$ as well as $\mathtt{wave}^{t+1}$. We know that all robots that were in round t not in $\mathtt{wave}^t$ or $\mathtt{wave}^{t+1}$ are inner-robots and have not moved (Definition 3.9). Therefore, all robots in $\texttt{wave}^{t+1}$ in round $t+1$ must have been wave- or boundary-robots in round t. Because all wave- and boundary-robots in round t move into $\texttt{wave}^{t+1}$ the robots in $\texttt{wave}^{t+1}$ in round $t+1$ are the complete set of wave- and boundary-robots in round $t+1$. The boundary-robots are already identified, therefore we know the set of wave-robots. This allows us to apply Lemma 3.11 to compute the positions of the wave-robots in round t. All other robots are inner robots, therefore they do not move in round t. ■

Theorem 7 — restated. We assume robots according the $\mathcal{O}$BLOT model and $\mathcal{F}$SYNC scheduler with a viewing range of $2+\sqrt{2}$ in a swarm with a convex Connectivity-Boundary (Definition 1.6) and no 1-holes (Definition 1.7). The WAVE-PROTOCOL (Algorithm 7) leads to NEAR-GATHERING and does not change the symmetry.

Proof. The protocol is executable in the assumed model (Lemma 3.13). From [27] we know, that ε-GO-TO-THE-MIDDLE converges towards gathering. From Lemma 3.12 we know, that $\mathbf{b}^{t+1} = \varepsilon\text{-GTM}(\mathbf{b}^t)$. Therefore, $\mathbf{b}^t$ will converge towards gathering for $t \leftarrow \infty$. Because $\mathbf{b}^t$ is the Connectivity-Boundary, all robots are within $area(\mathbf{b}^t)$. At the same time, each robot keeps its own position (Lemma 3.14). Therefore, eventually a NEAR-GATHERING will be reached. Because Algorithm 7 is locally invertible (Lemma 3.16) we can follow from Theorem 5 that the symmetries are not changed. ■

3.3.4 Contracting Waves as a General Near-Gathering Protocol

The WAVE-PROTOCOL presented in this section works for general swarms where the Connectivity-Boundary is not necessarily convex and the swarm may contain (large) holes. A non-convex Connectivity-Boundary and (large) holes changes some properties of the protocol because the detection of the robot role (boundary-, wave- or inner-robot) is not perfect anymore. For example, not all robots on the Connectivity-Boundary can detect locally that they are boundary-robots. However, all robots that are near the global enclosing circle can locally detect whether they are boundary-robots or not and move inside. We use this insight to show analog to the proof in Section 2.2.2, that eventually a Diam-1-Configuration is reached. Note, that the protocol need to be changed slightly, and the viewing range must be doubled which is explained during the proof. However, these changes do only affect swarms with a non-convex Connectivity-Boundary or larger holes; the changed protocol is equivalent to Algorithm 7 if the Connectivity-Boundary is convex and the swarm contains no 1-hole. While a NEAR-GATHERING is always reached as shown below, the symmetry preservation can only be guarantied of the Connectivity-Boundary is convex and the swarm contains no 1-hole.

Lemma 3.17 We assume robots according the $\mathcal{O}$BLOT model and $\mathcal{F}$SYNC scheduler with a viewing range of $4+2\sqrt{2}$ in a connected swarm. Algorithm 7 with slight changes leads to NEAR-GATHERING.

Proof. Not all robots on the Connectivity-Boundary can detect locally that they are boundary-robots. For situations where a robot is only connected to one other robots (e.g. a robot at the end of a line) ε-GTM must be slightly altered such that a robot considers one neighbor as its left and right neighbor at the same time in these situations. A large hole in the swarm may result in robots that locally assume they are boundary-robots but are not on the Connectivity-Boundary. This results in multiple disconnected chains of boundary-robots that perform ε-GTM. Because ε-GTM results never in a loss of connectivity, this is not a harmful behavior. However, it may result in collisions if the wave-robots are do move accordingly.

The detection of wave-robots has the following problem. Whenever two boundary-robots of different chains are close to each other, a robot may be in two different waves. This leads to unambiguous behavior. We can fix that as follows by doubling the viewing range of robots. A boundary-robot uses its regular viewing range $(2+\sqrt{2})$ to look whether it in on the local convex hull, if not it does not count as boundary-robots. This way, a connected chain of boundary-robots always ends with a robot that is boundary-robot in its 1-neighborhood but not in its regular-viewing range and that does not move. For the wave-definition this static robot at the end is a boundary-robot b_k^t that does not move (therefore $b_k^t = b_k^{t+1} = b_k^{t+2}$). Wave robots use the doubled viewing range $(4+2\sqrt{2})$ to check the same for their observed boundary-robots (a boundary-robots b that is relevant for the computation of wave-robots w's wave-segment is within the regular viewing-range of w, the doubled is used to check whether b counts as boundary robot according the new definition). The waves constructed this way cannot overlap, because two robots that are locally detected as boundary-robots but not connected via the Connectivity-Boundary within their viewing range must have a distance $\geq 2+\sqrt{2}$ and a wave-segment has only a diameter of $\leq \sqrt{2}$.

Boundary-robots near the global enclosing circle in circle-cap S_1 (see Section 2.2.2 for the definition) are always boundary-robots according the new criterias. Connected chains of boundary-robots must always end outside S_1. Therefore, all boundary-robots leave $S_{1/2}$ after a finite amount of time. Because the wave-robots stay within the convex hull of their respective boundary-robots, all robots leave $S_{1/2}$ after a finite amount of time. Analog to the proof in Section 2.2.2 we can follow, that the protocol reaches a Diam-1-Configuration after a finite amount of time. ■

While Algorithm 7 is a near-gathering protocol it is inferior to λ-contracting NEAR-GATHERING protocols (Definition 2.3) concerning the running time. λ-contracting NEAR-GATHERING protocols reach a Diam-1-Configuration of n robots after $\mathcal{O}(n^2)$. In the following we show, that there exist configurations where Algorithm 7 needs $\Omega(\log(n) \cdot n^2)$ rounds. The proof is in parts analog to the running time proofs of GO-TO-THE-CENTER and GO-TO-THE-MIDDLE in [18, 24].

Lemma 3.18 Let $\mathbf{z}^0$ be a regular n-gon with side length 1. Algorithm 7 needs $\Omega(\log(n) \cdot n^2)$ rounds to reach a Diam-1-Configuration.

Proof. In the configuration $\mathbf{z}^0$ all robots are boundary robots and perform ε-GTM. The configuration $\mathbf{z}^1$ (and iteratively $\mathbf{z}^t$ for all $t > 0$) is again a regular n-gon, just with a smaller side length. We define the following measures

- $s(t)$ is the side-length of the n gon in round t
- $r(t)$ is the radius of the n-gon in round t
- $\gamma = \pi - {}^{2\pi}/_{n}$ is the inner-angle of the n-gon (values in radian)

In each round, the robots move factor ε towards the midpoint of their neighbors. This is a distance of $\varepsilon \cdot s(t) \cdot \cos(\gamma/2)$. The radius decreases by twice this amount.

$$r(t+1) = r(t) - \varepsilon \cdot s(t) \cdot \cos(\gamma/2).$$

We can approximate the side length from the radius as follows

$$s(t) \leq r(t) \cdot {}^{2\pi}\!/_{n}.$$

The cosine can be approximated with the following formula

$$cos(x) \leq {}^{\pi}\!/_{2} - x \quad \text{for} \quad 0 \leq x \leq {}^{\pi}\!/_{2}$$

This results for $cos({}^{\gamma}\!/_{2})$ with $\gamma = \pi - {}^{2\pi}\!/_{n}$ in

$$cos({}^{\gamma}\!/_{2}) \leq 4{}^{\pi}\!/_{n}.$$

Therefore, we can estimate the radius as follows

$$r(t+1) \leq r(t) - \varepsilon \cdot r(t) \cdot 2{}^{\pi}\!/_{n} \cdot 4{}^{\pi}\!/_{n} = r(t) - \varepsilon\Theta({}^{r(t)}\!/_{n^2})$$

With $\varepsilon < 1$ it needs at least $\Omega(n^2)$ round to halve the radius. Therefore, we need at least $\Omega(\log(n) \cdot n^2)$ rounds to reach a radius ≤ 1. ■

4. Forming Large Patterns from Contracted Swarms

In this chapter we present a protocol that solves ARBITRARY-PATTERN-FORMATION. The previous chapters showed how a swarm could reach a Diam-1-Configuration by NEAR-GATHERING. The following main result of this chapter builds on this by assuming that the initial configuration is a Diam-1-Configuration.

Theorem 8 — restated. A connected pattern P can be formed by $|P|$ disoriented $\mathcal{O}$BLOT robots with limited viewing range in the $\mathcal{F}$SYNC model from a Diam-1-Configuration $\mathbf{z}$ if and only if $\operatorname{sym}(\mathbf{z}) \mid \operatorname{sym}(P)$. The formation takes $O(|P|)$ rounds, which is worst-case optimal.

Note that the restriction on the symmetricity (see Definition 1.2) is necessary in general even for swarm with global visibility as (see Theorem 1). The λ-contracting NEAR-GATHERING protocols from Chapter 2 lead to a Diam-1-Configuration from any strongly connected initial configuration $\mathbf{z}_0$. However, it does not make any guarantee on the symmetricity of the Diam-1-Configuration. This allows only limited usage of the theorem above. In Section 3.3 a protocol is presented that reaches a Diam-1-Configuration with the same symmetricity as $\mathbf{z}_0$ under the assumption that the Connectivity-Boundary is convex and the $\mathbf{z}_0$ contains no 1-holes. The theorem above allows directly to follow that for such $\mathbf{z}_0$, that can have a diameter up to $\Omega(n)$, any connected pattern P with $\operatorname{sym}(\mathbf{z}_0) \mid \operatorname{sym}(P)$ can be formed (see Theorem 9).

Requiring that P is connected (see Section 1.3) is natural for robots of limited viewing range. However, our technique can even be adapted to form disconnected patterns, as long as they contain a connected component of size ≥ 3 (see Section 5.1 for a brief discussion).

In a nutshell, our protocol partitions the input pattern P into $\operatorname{sym}(P)$ rotation-symmetric *components*. Using the initial mutual visibility, we let the robots form one cluster, called *drawing formation*, per component. Such a drawing formation relies on a careful placement of its contained robots to store information about the component it is responsible for and to coordinate its movement. We show how the drawing formation's robots can compute and coordinately move along a deliberately constructed path through the component in order to "draw" the pattern by dropping one robot at each contained coordinate. The content of this chapter was first published in

Forming Large Patterns with Local Robots in the OBLOT Model [39]
Christopher Hahn, Jonas Harbig and Peter Kling
Symposium on Algorithmic Foundations of Dynamic Networks (SAND) 2024

Outline. Section 4.1 contains the major part of our protocol description and its analysis. That section formalizes notions like drawing formations or drawing paths and details how we coordinate the robots that form a drawing formation. At the section's end, we prove Theorem 8 under the assumptions that there is a drawing path that adheres to certain conditions (namely Definition 4.12) and that $\operatorname{sym}(P) < |P|/2$. The construction of such a drawing path is subject of Section 4.2. In Section 4.3 pseudocode for the whole protocol is presented.

4.1 Forming Patterns via Drawing

Given a pattern P of symmetricity $\operatorname{sym}(P) \in \mathbb{N}$, we "draw" P using $\operatorname{sym}(P)$ *drawing formations*. Each drawing formation consists of a carefully arranged subset of *state robots* and is responsible to form one of $\operatorname{sym}(P)$ symmetric subpatterns $P' \subseteq P$. The state robots' careful placement enables them to coordinately move through P' along a specific *drawing path*. While doing so, some state robots are "dropped" at nearby pattern coordinates to form P'.

We start in Section 4.1.1 by formalizing the idea of drawing formations and related concepts. Section 4.1.2 details the legal arrangements of state robots in a drawing formation and shows how we can use this to coordinate state robots. Equipped with this coordination, Section 4.1.3 formalizes the drawing path $\boldsymbol{v}$ and what properties a drawing formation F must have in order to traverse $\boldsymbol{v}$ while suitably dropping robots. Afterward, Section 4.1.4 shows how we can create $\operatorname{sym}(P)$ different drawing formations and use them to draw suitable, symmetric subpatterns of P. Finally, Section 4.1.5 puts everything together to prove Theorem 8. We often define protocols implicitly during the proofs. To better illustrate the protocols, we provide pseudocode in Section 4.3.

4.1.1 Drawing Formations & Movement

We first define the *drawing hull*, representing the general shape of a drawing formation.

Definition 4.1 — Drawing Hull. A *drawing hull* $H = (a, \mathbf{d}, \phi, \Delta)$ consists of an *anchor* $a \in \mathbb{R}^2$, a *direction* $\mathbf{d} \in \mathbb{R}^2$ with $\|\mathbf{d}\|_2 = 1$, a *span* $\phi \in (0, \pi/3]$, and a *diameter* $\Delta \in (0, 1]$.

As illustrated in Figure 4.1a, one should think of a drawing hull $H = (a, \mathbf{d}, \phi, \Delta)$ as the point set $\{\, x \in \mathbb{R}^2 \mid \operatorname{dist}(x, a) \leq \Delta \wedge \angle(\mathbf{d}, x - a) \in [0, \phi) \,\}$.[1] With this in mind, we sometimes abuse notation and identify H with this set to write, e.g., $\operatorname{pos}(r) \in H$ for a robot r.

A *drawing formation* is defined by a drawing hull and all robots contained in it. These robots form a tight cluster whose exact placement inside the hull (the drawing formation's *state*) allows us to coordinate their movement (see Section 4.1.2).

Definition 4.2 — Drawing Formation. A *drawing formation* $F = (H_F, \mathcal{R}_F)$ consists of a drawing hull H_F and the robot set $\mathcal{R}_F := \{\, r \in \mathbf{z} \mid \operatorname{pos}(r) \in H_F \,\}$. We call $r \in \mathcal{R}_F$ a *state robot* of F and $\mathcal{S}_F := \operatorname{pos}(\mathcal{R}_F)$ the *state* of F. The *size* of F is $|\mathcal{R}_F|$.

[1] Note that $\phi \leq \pi/3$ ensures that Δ is indeed the diameter of the point set H.

We sometimes identify a drawing formation with its associated hull, allowing us to, e.g., speak of a drawing formation's anchor or diameter.

A drawing formation F forms a given pattern by "moving" F along a specific *drawing path* (see Section 4.1.3) that visits all pattern coordinates, dropping one state robot per pattern coordinate along the way. The following Definition formalizes such *moves* (see Figure 4.1c for an illustration).

Definition 4.3 — Move. Consider a drawing formation $F = (H_F, \mathcal{R}_F)$ with drawing hull $H_F = (p, \mathbf{d}, \phi)$ in configuration $\mathbf{z}$. Let $\mathbf{z}^+$ denote the configuration in the next round. We say F *moves* from p (in configuration $\mathbf{z}$) to p' (in configuration $\mathbf{z}^+$) if a state robot subset $\mathcal{R}_{F'} \subseteq \mathcal{R}_F$ of F forms a drawing formation $F' = \big((p', \mathbf{d}, \phi), \mathcal{R}_{F'}\big)$ in configuration $\mathbf{z}^+$. We call the robots $\mathcal{R}_F \setminus \mathcal{R}_{F'}$ *dropped* robots.

When moving from one drawing path vertex to the next, the remaining state robots change state (their placement in the drawing formation) to encode the progress on the drawing path. To ensure that a drawing formation can adopt any (reasonable) state after a movement, we restrict its movement distance to $1 - \Delta$ (s.t. each state robot can reach any other location in the resulting drawing formation of diameter Δ).

Observation 4.1 Consider a drawing formation F of diameter Δ that moves from position p to p'. If $\operatorname{dist}(p, p') \leq 1 - \Delta$, the robots that are not dropped can form any state in the resulting drawing formation.

4.1.2 States of a Drawing Formation

Given a target pattern P, our protocol considers only drawing formations F with fixed span $\phi = 2\pi / \operatorname{sym}(P)$ (depending only on P) and fixed diameter Δ (constant). Moving F between vertices of the drawing path (see Section 4.1.3) requires a coordinated movement of F's state robots. To achieve this, any robot must

1. decide whether it is one of F's state robots and, if so,
2. know the current progress on the drawing path.

To achieve (1), we use a careful placement of three *defining robots* that allows any robot r that sees them to deduce the remaining hull parameters (anchor and direction); once all four hull parameters are known, r can compute the hull H_F and decide whether it lies inside H_F or not. To achieve (2), we require that any additional state robots are placed on an ε-grid ($\varepsilon > 0$ fixed, depending only on P) that is aligned with the defining robots; using an arbitrary but fixed enumeration scheme for ℓ robots on such a grid, all state robots can agree on the same ordering of states and use it (in combination with F's size ℓ) to encode the progress on the drawing path.

Legal States. We continue to formalize this idea for a given parameter $\varepsilon > 0$. The placement of the defining robots r_1, r_2, and r_3 of a drawing formation F with anchor a and direction $\mathbf{d}$ is as follows:

1. r_1 is at the anchor a,
2. r_2 is at distance ε in direction $\mathbf{d}$ from anchor a, and
3. r_3 is at distance $\in \{2\varepsilon, 4\varepsilon, \dots\}$ in direction $\mathbf{d}$ from r_2.

Further state robots (if any) must be placed on the non-negative 2ε-grid with origin r_2 and whose x-axis is aligned with $\mathbf{d}$ (see Figure 4.1b).

The robot pair $\{r_1, r_2\}$ can be identified since they are the only state robots with distance ε. And since r_3 at distance $\geq 2\varepsilon$ is closer to r_2 than to r_1, robots can distinguish r_1 from r_2, from which they can infer both the hull's anchor and direction.

We get the following set of potential state robot locations:

Definition 4.4 — ε-Granular Locations. Consider a drawing formation $F = (H_F, \mathcal{R}_F)$ with anchor a and direction $\mathbf{d}$. Let $\mathbf{d}_\perp$ be the unit vector with $\angle(\mathbf{d}, \mathbf{d}_\perp) = \pi/2$. The set of ε-*granular locations* of F is

$$L_F(\varepsilon) := \{\, a, a + (1+2i)\varepsilon \cdot \mathbf{d} + 2j\varepsilon \cdot \mathbf{d}_\perp \mid i, j \in \mathbb{N}_0 \,\} \cap H_F. \tag{4.1}$$

States (i.e., state robot placements) considered legal by our protocol consist of all possible placements on ε-granular locations with the mentioned restrictions on the three defining robots' positions.

Definition 4.5 — ε-Granular States. Consider a drawing formation F with anchor a and direction $\mathbf{d}$. The set of ε-*granular states* of F is

$$A_F(\varepsilon) := \{\, \mathcal{S} \cup \mathcal{T} \mid \mathcal{S} \in \mathcal{P}(L_F), \mathcal{T} \in T(\varepsilon) \,\}.$$

with $T(\varepsilon) := \bigcup_{i=1}^{\lfloor (\Delta/\varepsilon - 1)/2 \rfloor} \{\, \{a, a + \varepsilon \cdot \mathbf{d}, a + (1+2i)\varepsilon \cdot \mathbf{d}\} \,\}$ being the sets of defining robots. For $\ell \in \mathbb{N}$ we define $A_F^\ell(\varepsilon) := \{\, \mathcal{S} \in A_F(\varepsilon) \mid |\mathcal{S}| = \ell \,\}$ as the set of all ε-granular states of F that can be adopted with ℓ state robots.

Our protocol considers only drawing formations that adhere to the above restrictions, leading us to the following Definition:

Definition 4.6 — ε-Granular Drawing Formation. A drawing formation F is ε-*granular* if F is in an ε-granular state and if F's state robots know[a] the fixed parameters ε, Δ, and ϕ.

[a]The parameters are either hard-coded into the protocol or can be computed from the target pattern P.

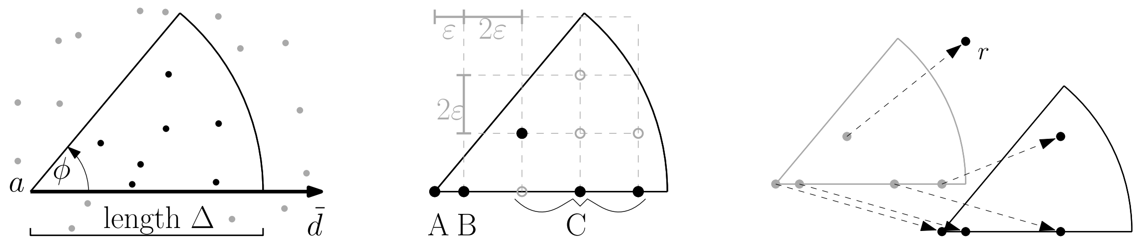
Note that all subsets of robots in the current configuration that fulfill the definition above are ε-granular drawing formations. We require that $r \in F$ knows the parameter ε, Δ and ϕ of F. Therefore $r \in F$ can check all subsets in its viewing range whether they are drawing formations with these parameters. With a viewing range of $1 \geq \Delta$, r observes all robots in F and can compute the drawing hull of F.

Observation 4.2 Let F be a ε-granular drawing formation. All state robots in F can compute the anchor of F and $\mathbf{d}$.

As a final property, we only want non-overlapping drawing formations. If two drawing hulls were overlapping, a robot might be state robot in two drawing formations that move in different directions.

Definition 4.7 — Validity. An ε-granular drawing formation $F = (H_F, \mathcal{R}_F)$ is *valid in configuration* $\mathbf{z} \supseteq \mathcal{R}_F$ if for any other ε-granular drawing formation $F' = (H_{F'}, \mathcal{R}_{F'})$ we have $H_F \cap H_{F'} = \emptyset$.

Counting via States. Given an ε-granular drawing formation F of size ℓ (i.e., consisting of ℓ state robots), we can easily enumerate $A_F^\ell(\varepsilon)$ in a way that depends solely on the relative positions of its locations. In particular, all state robots can use this enumeration and thus agree on the order of states, which basically equips the state robots with a shared



(a) Drawing formation F with drawing hull H_F and state robots $r \in \mathcal{R}_F$ (black) and other robots $r \notin \mathcal{R}_F$ (gray).

(b) ε-granular drawing formation; locations A and B must each contain a robot, the locations on the bottom line at C must in sum contain ≥ 1 robots.

(c) Movement of a drawing formation where robot r is dropped.

Figure 4.1

counter. A concrete implementation is depicted in Section 4.1.6.

Definition 4.8 — *i*-th State. Using an arbitrary, unique state enumeration on $A_F^\ell(\varepsilon)$ that depends solely on the relative position of locations, for $i \in \{1, 2, \ldots, |A_F^\ell(\varepsilon)|\}$ we define the *i-th state* of F as the corresponding state in this enumeration.

We conclude with a lower bound on the number of states that an ε-granular drawing formation may have.

Lemma 4.1 Consider a drawing formation F with hull diameter Δ and span $\phi \in (0, \pi/3]$. We have $|A_F^3(\varepsilon)| = \lfloor(\Delta/\varepsilon - 1)/2\rfloor$ and $|A_F^\ell(\varepsilon)| = \Omega(\Delta^3 \cdot \phi/\varepsilon^3)$ for $4 \leq \ell \leq |L_F(\varepsilon)| - 1$.

Proof. For $\ell = 3$ robots, only the third defining robot r_3 can choose between multiple locations; the first and second defining robots r_1 and r_2 have a fixed location inside H_F. Since r_3 can be placed on any of the locations of the form $a + (1+2i)\varepsilon \cdot \mathbf{d}$ for $i \in \mathbb{N}$ that lies in H_F, we get $|A_F^3| = \lfloor(\Delta/\varepsilon - 1)/2\rfloor$.

For $\ell = 4$, observe that there are $k := |L_F(\varepsilon)| = \Omega(\Delta^2 \cdot \phi/\varepsilon^2)$ ε-granular locations in F, since the 2ε-grid allows for $\Omega(1/\varepsilon^2)$ many locations per unit area and the total area covered by F's hull is $\pi \cdot \Delta^2 \cdot \phi/(2\pi) = \Delta^2 \cdot \phi/2$. Again, the first two defining robots have a fixed location, while the third defining robot may occupy one of $\Omega(\Delta/\varepsilon)$ many locations. The remaining $\ell - 3 \geq 1$ robots can be arranged on the remaining $k - 3$ locations in $\binom{k-3}{\ell-3}$ ways. By the Lemma's restriction on ℓ we have $\ell - 3 \geq 4$ and $\ell - 3 \leq k - 4$, such that $\binom{k-3}{\ell-3} \geq \binom{k-3}{k-4} = \binom{k-3}{1} = \Omega(\Delta^2 \cdot \phi/\varepsilon^2)$. Together, we get the desired bound. ■

4.1.3 Drawing a Pattern via a Drawing Path

This section introduces the *drawing path* of a (sub-) pattern P, which is a path in $\mathbb{R}^2$ that visits all pattern coordinates. This path should allow a drawing formation to move along its vertices while dropping one state robot per pattern coordinate along the way to form P. An instructive illustration of the idea can be found in Section 4.1.6.

A drawing path $\boldsymbol{v}$ has a parameter δ that controls the maximal distance between consecutive vertices as well as between each pattern coordinate and the path. Moreover, $\boldsymbol{v}$ must depend only on P, such that oblivious robots can all recalculate $\boldsymbol{v}$ each LCM-cycle.

Definition 4.9 — Drawing Path. Consider any pattern P. A path $\boldsymbol{v} = (v_j)_{j=1}^k$ of k vertices $v_j \in \mathbb{R}^2$ is a *δ-drawing path* of P if

1. $\boldsymbol{v}$ can be calculated from P,
2. $\forall p \in P\colon \operatorname{dist}(p, \boldsymbol{v}) \leq 1 - \delta$, and
3. $\forall j \in \{1, 2, \ldots, k-1\} : \operatorname{dist}(v_j, v_{j+1}) \leq 1 - \delta$.

Choosing δ equal to the diameter of a drawing formation F enables F to traverse $\boldsymbol{v}$ while forming any state (Observation 4.1). We omit δ if it is irrelevant for the matter at hand.

When traversing $\boldsymbol{v}$, we want a drawing formation to drop a robot at pattern coordinate $p \in P$ when leaving the latest vertex v_j that is close enough to p. This ensures that, at any time after being dropped, the dropped robot has a distance of at least $1 - \delta$ to the drawing formation that dropped it. We say p is *covered* by vertex v_j.

Definition 4.10 — Covered Coordinates. Consider a δ-drawing path $\boldsymbol{v} = (v_j)_{j=1}^k$ of a pattern P. Coordinate $p \in P$ is *covered* by vertex v_j if $j \in \{1, 2, \ldots, k\}$ is the maximal index for which $\operatorname{dist}(p, v_j) \leq 1 - \delta$. Let $\operatorname{cov}(v_j)$ denote the set of all coordinates covered by v_j.

We extend $\operatorname{cov}(\bullet)$ in the natural way to subsequence $\boldsymbol{v}'$ of $\boldsymbol{v}$, such that $\operatorname{cov}(\boldsymbol{v}') = \bigcup_{u \in \boldsymbol{v}'} \operatorname{cov}(u)$. Care must be taken once a drawing formation dropped so many robots that it reached size $\ell = 3$: It must not drop further robots before the path's end, since the remaining two robots would no longer form a drawing formation and could not coordinate (see Section 4.1.2). We capture this (possibly non-existent) path region in the following Definition 4.11.

Definition 4.11 — Tail of a Drawing Path. The *tail* $\operatorname{tail}(\boldsymbol{v})$ of a drawing path $\boldsymbol{v} = (v_j)_{j=1}^k$ is the longest suffix $(v_j)_{j=s}^k$ s.t. $\sum_{j=s}^k |\operatorname{cov}(v_j)| < 4$.

As a final notion, we declare when a drawing formation and a drawing path are *compatible* (i.e., can be used to form the path's pattern). Here, we use $\operatorname{hops}(v_s, v_t)$ to denote the number of edges between two path vertices $v_s, v_t \in \boldsymbol{v}$.

Definition 4.12 — Compatibility. An ε-granular drawing formation F with diameter Δ and span Φ is *compatible* with a δ-drawing path $\boldsymbol{v} = (v_j)_{j=1}^k$ of a pattern P if

1. $\varepsilon < \operatorname{mindist}(P)$ and $\Delta \leq \delta$,
2. $\forall s < t$ s.t. $\bigcup_{j=s}^{t-1} \operatorname{cov}(v_j) = \emptyset\colon \operatorname{hops}(v_s, v_t) \leq |A_F^4(\varepsilon)|$,
3. $|\operatorname{tail}(\boldsymbol{v})| \leq |A_F^3(\varepsilon)|$, and
4. $|\operatorname{cov}(\operatorname{tail}(v))| = 3$ and $\operatorname{cov}(\operatorname{tail}(\boldsymbol{v})) \subseteq \mathcal{B}(v_k, 1)$.

Item 1 ensures that the distance ε (identifying F's defining robots) does not occur in P and that F can traverse $\boldsymbol{v}$ (by Observation 4.1). Item 2 requires that, after dropping a robot, the state space $A_F^\ell \supseteq A_F^4$ of the remaining ℓ robots is large enough to encode the progress towards the next vertex where a robot is dropped. These two properties are used in Lemma 4.2 to prove that F can traverse the non-tail of $\boldsymbol{v}$ while appropriately dropping robots.

Lemma 4.2 Consider a compatible drawing path $\boldsymbol{v} = (v_j)_{j=1}^k$ of a pattern P. Let $\mathbf{z}$ be the configuration formed by a drawing formation F of size $|P|$ in state 1 anchored in v_1 that is compatible with $\boldsymbol{v}$. Then F can traverse $\boldsymbol{v}$ by taking one edge per LCM-cycle while dropping one robot at each coordinate in $\operatorname{cov}(v_j)$ when leaving $v_j \notin \operatorname{tail}(\boldsymbol{v})$.

Proof. **Enumeration of States.** We defined in Definition 4.8 an enumeration of A_F^ℓ, let the i-th state of A_F^ℓ be $state(i,\ell)$. We define the following unique states for the path, using an enumeration that includes different sizes ℓ of drawing formations fitting to the number of not dropped robots at a node.

$$f(v_i) := |\operatorname{cov}((v_j)_{j=i}^k)|$$

$$g(v_i) := \max\left(|(v_j)_{j<i}^{i-1}|\right) \text{ with } \operatorname{cov}((v_j)_{j<i}^{i-1}) = \emptyset$$

$$state(v_i) := state(g(v_i)+1, f(v_i))$$

From (3) and (4) of Definition 4.12 follows directly, that all such states exist.

Induction Proof. Assumption: F with anchor on v_i with $|F| = |\operatorname{cov}((v_j)_{j=i}^k)|$ in state i and dropped robots on $\operatorname{cov}((v_j)_{j=1}^{i-1})$. Let $H_F = (a, \mathbf{d}, \phi, \Delta)$ be the drawing hull of F (Definition 4.1). We define the coordinate system $\mathcal{S}$ as the coordinate system with x direction $\mathbf{d}$ and origin at v_1. Start: At node v_1 this is initially given.

Step: No coordinates $p \in \mathcal{B}(v_i, 1-\delta)$ contain robots $r \notin F$, otherwise $p \in \operatorname{cov}(v_j) \cap \mathcal{B}(v_i, 1-\delta), j < i$ with is a contradiction to Definition 4.10. Therefore F is valid (Definition 4.7). $r \in F$ knows anchor a and direction vector $\mathbf{d}$ of F (see Observation 4.2) and F is unabigous because it is valid (i.e. $r \in F$ is not in another ε-granular drawing formation). Because v is a drawing path (Definition 4.9), r can compute v from P. With the assumption that a (the anchor of F) is at v_i and F in state $state(v_i)$, it can determine the coordinate system $\mathcal{S}$. $dist(v_i, v_{i+1}) \leq 1-\delta$ (by Definition 4.9) and $1-\delta \leq 1-\Delta$ (by Definition 4.12). From Observation 4.1 follows that F can move from v_i to v_{i+1}. F moves such that its new anchor is v_{i+1}, $|\operatorname{cov}(v_i)|$ robots are dropped onto the coordinates $\operatorname{cov}(v_i)$ and its new state $state(v_{i+1})$.

■

Lemma 4.3 uses Item 3 to traverse the tail and Item 4 to drop the final three robots at the tail's end. This final drop is slightly more involved: if the last three coordinates form, e.g., a straight path of edge length 1, our drawing formation cannot drop all robots at once. With the help of Item 4, we handle this via an intermediate step.

Lemma 4.3 Consider a compatible drawing path $\boldsymbol{v} = (v_j)_{j=1}^k$ of a pattern P. Let $\mathbf{z}$ be the configuration that has

1. one robot at each coordinate in $P \setminus \operatorname{cov}(\operatorname{tail}(\boldsymbol{v}))$ and
2. a drawing formation F of size 3 in state $|\operatorname{tail}(\boldsymbol{v})|$ anchored in v_k that is compatible with $\boldsymbol{v}$.

Then F can dissolve within two LCM-cycles while dropping one robot at each coordinate in $\operatorname{cov}(\operatorname{tail}(\boldsymbol{v}))$.

Proof. Lemma 4.3 follows immediately from Observation 4.3 and Lemma 4.4 that are given in the following. ■

We have shown in Lemma 4.2 that on all coordinates outside the $\operatorname{tail}(v)$, F can drop robots while traversing the drawing path v. On the tail it cannot further drop robots before reaching the end; otherwise $|\mathcal{R}_F| \leq 2$, which is not an ε-granular drawing formation anymore. In fact, it can never be a valid drawing formation because two robots can not encode the direction $\mathbf{d}$ in a model without a compass. Therefore, F does not drop robots

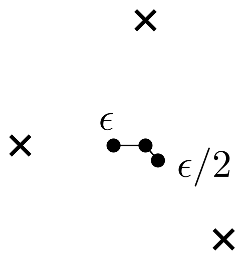


Figure 4.2: A depiction of an ε-intermediate-shape (Definition 4.13). All positions ($\times$) have a distance of 1 to the robot ($\bullet$) in the center.

onto $\text{cov}(tail(v))$ during the traversal. Instead, the drawing formation moves onto v_k, and the robots will move from there onto $\text{cov}(tail(v))$. Because $dist(\text{cov}(tail(v)), v_k) \leq 1$ for a compatible path, the drawing formation is close enough to all remaining positions. However, not all robots in $\mathcal{R}_F$ are on v_k; they can have a distance up to Δ. If $dist(v_k, p) > 1 - \Delta, p \in \text{cov}(tail(v))$ then the drawing formation F can be placed inconveniently such that $dist(r, p) > 1, r \in \mathcal{R}_F$. We will add an intermediate step that reshapes the drawing formation such that all robots have a distance of ≤ 1 to the coordinate they must obtain in the end. Robots may leave the drawing hull H_F, but most properties of a drawing formation must still be fulfilled. This intermediate shape F_{inter} must be valid in the sense that robots in F_{inter} must be able to determine the position of v_k and the direction vector $\mathbf{d}$ to compute the global coordinate system. In the following, we will define the intermediate shape and prove that robots can obtain this information.

Definition 4.13 — ε-intermediate-shape. Let P be a pattern and $v = (v_1, \cdots, v_k)$ be its drawing path with

1. $|\text{cov}(tail(v))| = 3$ and
2. $\mathcal{B}(v_k, 1) \supseteq \text{cov}(tail(v))$

Let $\text{cov}(tail(v)) = \{p_1, p_2, p_3\}$. The intermediate shape F_{inter} definines the following positions for a set of three robots r_1, r_2, r_3.

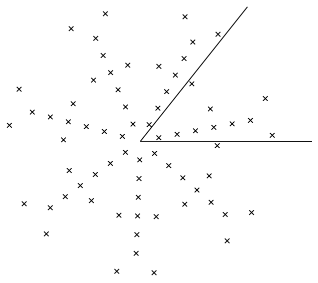
- r_1 is on v_k
- r_2 is distance $\varepsilon/2$ in direction p_2 and
- r_3 is distance $\varepsilon/3$ in direction p_3

See Figure 4.2 for a depiction of an intermediate shape.

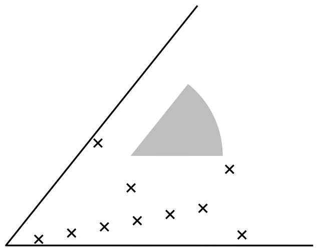
Observation 4.3 Let F_{inter} be an intermediate shape as in Definition 4.13. It is clear, that $dist(r_1, p_1) \leq 1$, $dist(r_2, p_2) \leq 1$ and $dist(r_3, p_3) \leq 1$.

Lemma 4.4 Let P be a pattern and $v = (v_1, \cdots, v_k)$ be a drawing path which is compatible with an ε-ganular drawing formation. Let F_{inter} be an intermediate shape as defined in Definition 4.13 which is on v_k. Let $dist(v_k, \mathbf{z} \setminus F_{inter}) > \varepsilon$. All robots $r \in F_{inter}$ can decide, that they are in F_{inter} and can compute the positions of $tail(v_k)$ in their local coordinate system.

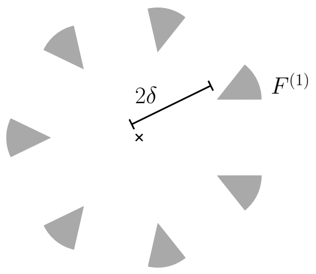
Proof. To decide, whether a robot $r \in F_{inter}$ is in F_{inter} it observes other robots in distance ε. Because the $dist(v_k, \mathbf{z} \setminus F_{inter}) > \varepsilon$ it only finds one triple of robots with distances $\varepsilon/2$ and $\varepsilon/3$. Let $r_1, r_2, r_3, p_1, p_2, p_3$ be as in Definition 4.13. The triangle r_1, r_2, r_3 is never equiliteral (one side has length $\varepsilon/2$ and another $\varepsilon/3$). With chirality, all three robots know,



(a) Pattern P with symmetricity $\mathrm{sym}(P) = 7$ and cone $C^{(1)}$.



(b) Symmetric component $P^{(1)}$ with an aligned drawing formation $F^{(1)}$.



(c) Initial drawing pattern $\mathcal{I}$ for $\mathrm{sym}(P) = 7$.

Figure 4.3

who is r_1, r_2 and r_3. They know the coordinates p_1, p_2 and p_3 as well as the position of v_k in the global coordinate system. The positions of r_1, r_2 and r_3 in the global coordinate system are also defined. Therefore, they can translate/rotate their local coordinate system and compute $tail(v_k)$. ■

4.1.4 Full Pattern via Many Drawing Formations

As shown in Section 4.1.3, we can draw any pattern P *if* we start in a suitable drawing formation F (and have a compatible drawing path). But we must first form such a drawing formation from the initial NEAR-GATHERING, which might have a symmetricity $s > 1$. In that case, since any drawing formation has symmetricity 1, we cannot form F (by Theorem 1). Instead, we show how to form $\mathrm{sym}(P)$ symmetric copies of F that are placed such that they

1. have symmetricity $\mathrm{sym}(P)$ (we have $s \mid \mathrm{sym}(P)$ or we cannot form P, even globally) and
2. do not interfere with each other (if using suitable drawing paths, see Section 4.2).

We start by partitioning the pattern P of symmetricity $\mathrm{sym}(P)$ into $\mathrm{sym}(P)$ *symmetric components*, each of which will be drawn by its own drawing formation.

Definition 4.14 — Cone & Symmetric Component. Let $\mathbf{e}_x := (1,0)$. For a pattern P of symmetricity $\mathrm{sym}(P)$, define the i-th *cone*

$$C^{(i)} := \{\, p \in \mathbb{R}^2 \mid \angle(\mathbf{e}_x, p) \in [(i-1)\cdot 2\pi/\mathrm{sym}(P), i\cdot 2\pi/\mathrm{sym}(P))\,\} \tag{4.2}$$

and the i-th *symmetric component* $P^{(i)} := P \cap C^{(i)}$.

Note that the symmetric components are pairwise disjoint and that $P = \bigcup_{i=1}^{\mathrm{sym}(P)} P^{(i)}$. See Figure 4.3a for an illustration.

To form pattern P, we first form a suitable *initial drawing pattern* of diameter ≤ 1. This initial drawing pattern places each robot $r \in \mathbf{z}$ in one of $\mathrm{sym}(P)$ ε-granular drawing formations $F^{(i)}$ of size $|P|/\mathrm{sym}(P)$ in state 1. If there exists a drawing path $\mathbf{v}^{(1)}$ for $P^{(1)}$ that is compatible with $F^{(1)}$ and starts at the anchor of $F^{(1)}$, we immediately get a corresponding (rotation-symmetric) drawing path for each $F^{(i)}$. Assuming that, additionally, those drawing paths lie "sufficiently inside" their respective cone $C^{(i)}$, we will prove a generalization of Lemmas 4.2 and 4.3, basically showing that the different $F^{(i)}$ can draw their $P^{(i)}$ without interfering with each other. The existence of suitable drawing paths is shown in Section 4.2.

To enable the robots to deduce the symmetric component $P^{(i)}$ they are drawing[2], we require $F^{(i)}$ to be *aligned* with $P^{(i)}$:

Definition 4.15 — Alignment. Fix $i \in \{1, 2, \dots, \mathrm{sym}(P)\}$ and the i-th symmetric component $P^{(i)}$ of a pattern P with symmetricity $\mathrm{sym}(P)$. Let F be a drawing formation with direction $\mathbf{d}$ and span ϕ. Then F is *aligned* with $P^{(i)}$ if $\angle(\mathbf{d}, \mathbf{e}_x) = (i-1)\cdot 2\pi/\mathrm{sym}(P)$ and if $\phi = \min\{2\pi/\mathrm{sym}(P), \pi/3\}$.

Figure 4.3b gives an illustration. With this, we define the *initial drawing pattern* of a pattern P and a suitable drawing formation F as follows (illustrated in Figure 4.3c):

Definition 4.16 — Initial Drawing Pattern. Fix a pattern P of symmetricity $\mathrm{sym}(P)$. An *initial drawing pattern* $\mathcal{I}$ for P is a configuration of $|P|$ robots that consists of $\mathrm{sym}(P)$ drawing formations $\{F^{(i)}\}_{i=1}^{\mathrm{sym}(P)}$ of diameter $\Delta \le 1/6$ in state 1 such that:

1. $F^{(1)}$ is aligned with $P^{(1)}$ and anchored in $(2\Delta, \pi/\mathrm{sym}(P))$.
2. $F^{(i)}$ is a rotation of $F^{(1)}$ by $(i-1)\cdot 2\pi/\mathrm{sym}(P)$.

We say $\mathcal{I}$ is ε-granular if the $F^{(i)}$ are ε-granular.

Note that, by construction, each $F^{(i)}$ is aligned with $P^{(i)}$. Moreover, $\mathcal{I}$ is a pattern with diameter ≤ 1 and has symmetricity $\mathrm{sym}(P)$, such that we can form $\mathcal{I}$ from any Diam-1-Configuration for which the symmetry condition (Theorem 1) holds.

It remains to prove that once the initial drawing pattern is formed, each drawing formation $F^{(i)}$ forms its symmetric component $P^{(i)}$ and does not interfere with the operation of any other drawing formation $F^{(j)}$ with $j \neq i$.

Lemma 4.5 Assume the current configuration is an ε-granular initial drawing pattern $\mathcal{I}$ for a pattern P of symmetricity $\mathrm{sym}(P)$. Consider a drawing path $\boldsymbol{v}^{(1)} = \left(v_j^{(1)}\right)_{j=1}^{k}$ of symmetric component $P^{(1)}$ that is compatible with $F^{(1)}$ such that

1. the path $\boldsymbol{v}^{(1)}$ starts in the anchor of $F^{(1)}$,
2. $\boldsymbol{v}^{(1)}$ lies in the first cone (i.e., $\boldsymbol{v}^{(1)} \subseteq C^{(1)}$) and

$$\mathrm{dist}\left(\boldsymbol{v}^{(1)}, \partial C^{(1)}\right) > \max\{\varepsilon, \Delta \cdot \sin(2\pi/\mathrm{sym}(P))\} \tag{4.3}$$

.

Then P can be formed in $\mathrm{O}(k)$ many LCM-cycles.

Proof. From Lemmas 4.2 and 4.3 we know, $P^{(1)}$ can be formed by $F^{(1)}$ assuming $P^{(1)}$ is the whole pattern. The traversal of this path takes at most k rounds to reach $v_k^{(1)}$ plus 2 rounds for dropping the last robots. With the other symmetric components $P^{(i)}$ next to $P^{(1)}$, $F^{(1)}$ can still form $P^{(1)}$ if $F^{(1)}$ is always valid. We will prove in the following, that $F^{(1)}$ is always valid.

Validity. $F^{(1)} = (\mathcal{R}_F, H_F)$ is valid, if there exists no subset $\mathcal{R}_G \subseteq \mathbf{z}$ which fulfills the criteria of Definition 4.6 such that $G = (\mathcal{R}_G, H_G)$ a ε-granular drawing formation $\mathcal{R}_G \neq \mathcal{R}_F$ and $H_F \cap H_G \neq \emptyset$. $F^{(1)}$ is aligned with $C^{(1)}$ (Definition 4.15). It follows directly, that $F^{(1)} \subseteq C^{(1)}$ and naturally for all symmetric drawing formation $F^{(i)}$ that $F^{(i)} \subseteq C^{(i)}$.

[2]Since robots are disoriented, they cannot deduce which $P^{(i)}$ they are drawing. But they can deduce $P^{(i)}$'s coordinates in their own, local coordinate systems.

Therefore, those drawing formations have disjunct hulls. From the proof of Lemma 4.2 we know that all dropped robots on $p \in P^{(1)}$ have a distance $\geq \Delta$ to H_F, therefore those robots cannot be part of $\mathcal{R}_G$. It is left to show, that $r \in C^{(i)}$ with $i \neq 1$ cannot build a drawing formation G that intersects H_F.

Two dropped robots r_1, r_2 on $p_1, p_2 \in P$ have $dist(r_1, r_2) > \varepsilon$, this follows from the fact that $\varepsilon \leq \text{mindist}(P)$ (Item 1 of Definition 4.12). Let F be anchored on $v_j^{(1)}$. By prerequisit (1) $dist(v_j^{(1)}, \partial\mathbb{C}^{(1)}) > \varepsilon$. Because $F^{(1)}$ is aligned to $C^{(1)}$, $dist(\partial C^{(1)}, \mathcal{R}_F) > \varepsilon$. Therefore $r_1 \in F^{(1)}$ and $r_2 \notin F^{(1)}$ have $dist(r_1, r_2) > \varepsilon$. Therefore, drawing formation G must have a pair of $dist(r_1, r_2) > \varepsilon$ which are part of the same drawing formation $F^{(i)}$. There must exist a third robot $r_3 \notin F^{(i)}$ collinear to r_1, r_2 with $dist(r_3, \{r_1, r_2\}) \leq \Delta$. W.l.o.g $i = 1$. Let $H_F = (a, \mathbf{d}, \Phi, \Delta)$ be the drawing hull of $F^{(1)}$. Because $F^{(1)}$ is aligned to $C^{(1)}$ (Definition 4.15) $\mathbf{d}$ is parallel to one side of its boundary and the line segment $\text{line}(a, -\mathbf{d}, \Delta)$ can never cut this side. $\text{line}(a, -\mathbf{d}, \Delta - \varepsilon)$ cuts the other side of $C^{(1)}$ boundary with angle Φ. The length of $\text{line}(a, -\mathbf{d}, \Delta - \varepsilon)$ must be $\geq \frac{dist(\partial C^{(1)})}{\sin(\Phi)}$. The line segment has a length of Δ. $\Phi = \frac{2\pi}{\text{sym}(P)}$ (Definition 4.15) $dist(a, \partial C^{(1)}) \geq \Delta \cdot \sin\left(\frac{2\pi}{\text{sym}(P)}\right)$ (assumption (2) of this lemma). This resolves to $\Delta - \varepsilon \geq \Delta \cdot \sin\left(\frac{2\pi}{\text{sym}(P)}\right) / \sin\left(\frac{2\pi}{\text{sym}(P)}\right) = \Delta$, which is obviously a contradiction. Therefore, $\text{line}(a, -\mathbf{d}, \Delta - \varepsilon)$ is completely in $C^{(1)}$ and cannot contain $r_3 \notin F^{(1)}$. ■

4.1.5 Putting Everything Together

Section 4.1.4 showed how to form a pattern P assuming we start in a suitable initial drawing pattern $\mathcal{I}$ and if a compatible drawing path $\boldsymbol{v}^{(1)}$ for $P^{(1)}$ exists. We continue by showing the existence of such a path for patterns with symmetricity $\text{sym}(P) := \text{sym}(P) < |P|/2$ (Lemma 4.6); patterns of larger symmetricity can be handled without drawing formations (Lemma 4.7). Afterward, in Lemma 4.8, we prove that each robot can distinguish in which phase of our protocol it is:

1. forming $\mathcal{I}$,
2. being part of a valid ε-granular drawing formation F,
3. dropping the last three robots at the tail's end, or
4. having been dropped at a pattern coordinate.

Putting everything together, we conclude this section with the proof of Theorem 8.

Lemma 4.6 Consider the ε-granular drawing formation $F^{(1)}$ of the initial drawing pattern for a connected pattern P of symmetricity $\text{sym}(P) < |P|/2$. The parameter ε can be chosen such that $F^{(1)}$ has $\left|L_{F^{(1)}}(\varepsilon)\right| \geq 2 + \left|P^{(1)}\right|$ ε-granular locations. Moreover, there exists a drawing path $\boldsymbol{v}^{(1)}$ of symmetric component $P^{(1)}$ that is compatible with $F^{(1)}$ such that

1. the path $\boldsymbol{v}^{(1)}$ starts in the anchor $a = (2\Delta, \pi/\text{sym}(P))$ of $F^{(1)}$,
2. $\boldsymbol{v}^{(1)}$ lies in the first cone (i.e., $\boldsymbol{v}^{(1)} \subseteq C^{(1)}$) and

$$\text{dist}\left(\boldsymbol{v}^{(1)}, \partial C^{(1)}\right) > \max\left\{\varepsilon, \Delta \cdot \sin(2\pi/\text{sym}(P))\right\} \tag{4.4}$$

.

The proof can be found in Section 4.2, where a suitable drawing path is constructed.

Lemma 4.7 Consider a pattern P with symmetricity $\mathrm{sym}(P) \geq |P|/2$. P can be formed.

In the following proof, we give a protocol for patterns with a symmetricity of n or $n/2$. If the diameter of P is ≤ 1, the pattern can be formed in one step, assuming suitable symmetricity. Otherwise, the robots form P scaled down to diameter 1. Then, the robots scale the small pattern back to its original (large) size. We show that this protocol can be computed and executed by the individual robots in our model with limited visibility and without memory.

Proof. For $\mathrm{sym}(P) = |P|/2$, the pattern P has coordinates

$$\cup_{i=0}^{\mathrm{sym}(P)-1}\{(D_1, \alpha_1 + i \cdot 2\pi/\mathrm{sym}(P)), (D_2, \alpha_2 + i \cdot 2\pi/\mathrm{sym}(P))\}$$

with $\alpha_1, \alpha_2 \leq 2\pi/\mathrm{sym}(P)$. $\mathrm{sym}(P) = |P|$ is just a special case with $D_1 = D_2$ and $\alpha_1 = \alpha_2$. Let $pos_t(r_i)$ be the positon of the robot r_i in round t. In a configuration with symmetricity $|P|/2$, the positions are as follows $pos_t(r_i) = (d_1(t), \beta_1(t) + ((i-1)/2) \cdot 2\pi/\mathrm{sym}(P))$ if i uneven, $pos_t(r_i) = (d_2(t), \beta_2(t) + (i/2) \cdot 2\pi/\mathrm{sym}(P))$ if i even. The robots form in the first round of the execution the pattern P scaled to a diameter ≤ 1. From there on, they scale the pattern up. Therefore $\frac{d_1(t)}{d_2(t)} = \frac{D_1}{D_2}$ and $\beta_1(t) = \alpha_1, \beta_2(t) = \alpha_2$ for $t > 1$. It is clear, that the "uneven" (i is uneven) robots on $pos_t(r_i) = (d_1(t), \alpha_1 + ((i-1)/2) \cdot 2\pi/\mathrm{sym}(P))$ can distinguish themself from the "even" robots on $pos_t(r_i) = (d_2(t), \alpha_2 + (i/2) \cdot 2\pi/\mathrm{sym}(P))$ when $\frac{d_1(t)}{d_2(t)} = \frac{D_1}{D_2}$. This is enough to compute the global coordinate system. The robots move onto positions with the same angle and

$$d_1(i+1) = \min\left(d_1(i)+1, d_1(i) \cdot \frac{d_2(i)+1}{d_2(i)}, D_1\right)$$

$$d_2(i+1) = \min\left(d_2(i)+1, d_2(i) \cdot \frac{d_1(i)+1}{d_1(i)}, D_2\right).$$

This reached $d_1(t) = D_1$and $d_2(t) = D_2$ after $\leq \max(D_1, D_2) \leq |P|$ rounds.

■

Lemma 4.8 Let $r \in \mathbf{z}$ be a robot in a configuration described in Lemma 4.6 executing the protocol from Lemma 4.5. Then r can locally distinguish between the following situations:
1. r is in an initial configuration before the initial drawing pattern is formed
2. $r \in H_F$ of a valid ε-granular drawing formation F
3. $r \in F_{inter}$ ε-intermediate shape
4. r has been dropped from a drawing formation

Proof. The proof argues all four situations separately in the following.
1. If a robot $r \in \mathbf{z}$ sees $|P|$ robots, it is in a Diam-1-Configuration and observes all robots $\mathbf{z}$. In that case, r checks whether $\mathbf{z}$ equals the initial drawing pattern $\mathcal{I}$ (Definition 4.16) or any of the execution steps resulting from the protocol described in Lemma 4.5 that are still a Diam-1-Configuration ($\Theta(1)$ many). If not, r can conclude that it is in the initial configuration.

2. Is clear by Observation 4.2.
3. Is shown in Lemma 4.4.
4. When a robot is dropped from a drawing formation it is on $p \in P$. We know, that $\text{mindist}(P) > \varepsilon$, therefore dropped robots can never form an ε-granular drawing formation. Moreover, in Lemma 4.5 we have shown that all ε-granular drawing formations in the configuration are valid. Therefore, no robot $r' \in H_F$ can be part of another ε-granular drawing formation F'. Similar arguments are true for F_{inter}. Hence, a robot observing that it is not in one of the first three situations knows that it has been dropped. ■

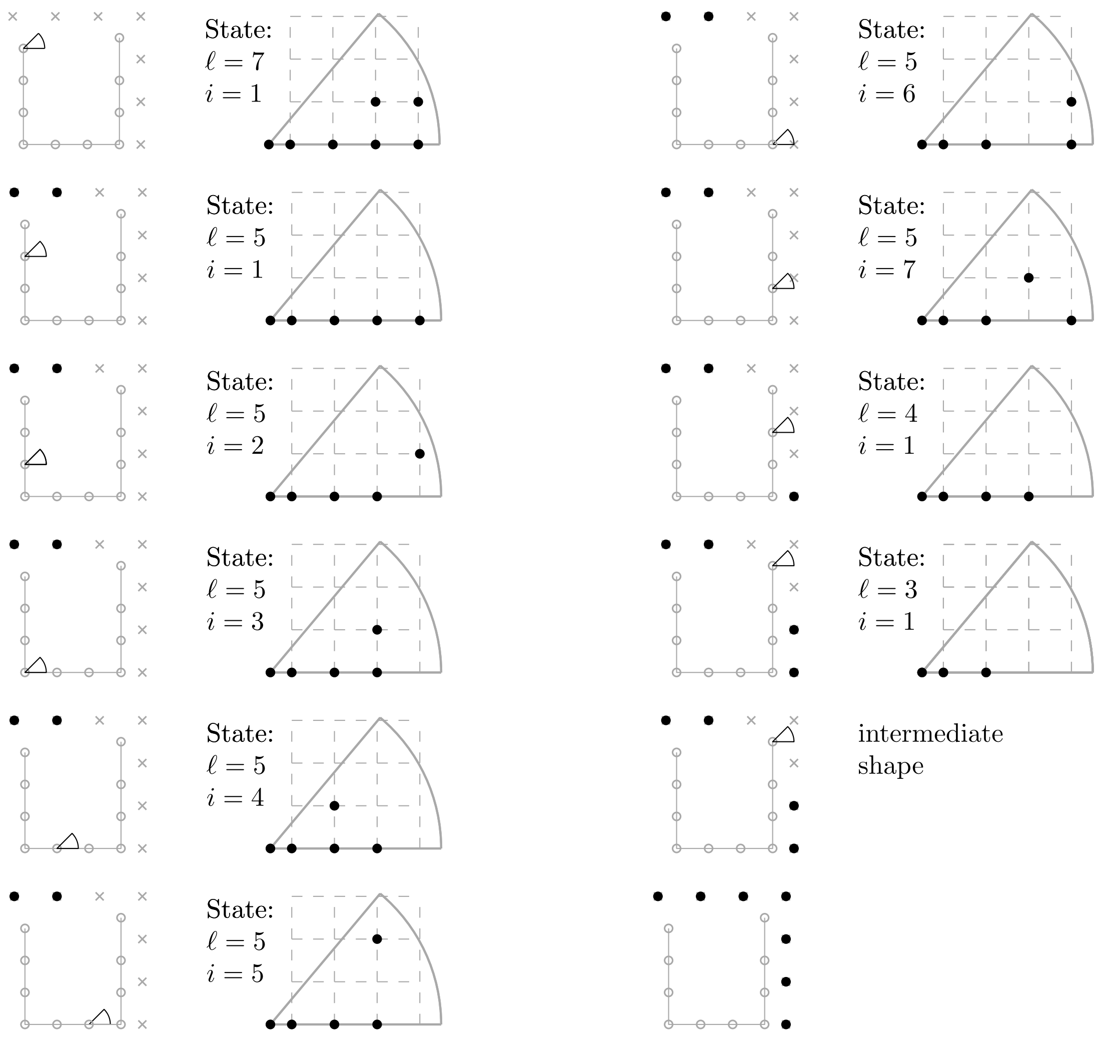
Theorem 8 — restated. A connected pattern P can be formed by $|P|$ disoriented $\mathcal{O}$BLOT robots with limited viewing range in the $\mathcal{F}$SYNC model from a Diam-1-Configuration $\mathbf{z}$ if and only if $\text{sym}(\mathbf{z}) \mid \text{sym}(P)$. The formation takes $\mathrm{O}(|P|)$ rounds, which is worst-case optimal.

Proof. The first direction follows from Theorem 1, since a pattern where $s_I := \text{sym}(I)$ does not divide $\text{sym}(P) := \text{sym}(P)$ cannot be formed.

For the second direction, assume $s_I \mid \text{sym}(P)$. By Lemma 4.7 the pattern can be formed if $\text{sym}(P) \geq |P|/2$, so assume $\text{sym}(P) < |P|/2$. Then we must execute the protocol described in Lemma 4.5, whose prerequisites (esp. the existence of a suitable drawing path) can be fulfilled: By Lemma 4.8 we know, that a robot can locally decide between the phases necessary to start and execute the protocol. If $\mathbf{z}$ is in an initial Diam-1-Configuration before forming the initial drawing pattern $\mathcal{I}$, the robots collectively form $\mathcal{I}$ (which has symmetricity $\text{sym}(P)$ and thus can be formed by Theorem 1). From Lemma 4.6's guarantee on $\left|L_{F^{(1)}}(\varepsilon)\right|$ we get that the drawing formation has enough locations for the number of robots as well as the existence of a suitable drawing path. Thus, we can apply Lemma 4.5 to get that one robot is dropped at each $p \in P$ after at most $\mathrm{O}(|P|)$ rounds. By Lemma 4.8, robots can realize that they have been dropped and remain idle on their respective pattern coordinate. ■

4.1.6 Illustration of Drawing Path Traversal

The following is an illustration how our protocol traverses a drawing path. The drawing formation F has initial size $\ell = 7$ and state $i = 1$. Black dots are robots (left: dropped robots; right: state robots), gray circles are vertices of the drawing path and gray crosses the pattern positions. Note that the drawing path used for the example *not* the outcome of the construction in Section 4.2. However, it is compatible (Definition 4.12) with the pattern. On the left side an image of the current configuration with a symbolic representation of the ε-granular drawing formation F is depicted, while the right side zooms in on F, indicating the state i of the drawing formation F each time (see Definition 4.8). Before the last step, the three state robots form an ε-intermediate shape (Definition 4.13), an example of such a shape is depicted in Figure 4.2.


State: $\ell = 7$ $i = 1$
State: $\ell = 5$ $i = 1$
State: $\ell = 5$ $i = 2$
State: $\ell = 5$ $i = 3$
State: $\ell = 5$ $i = 4$
State: $\ell = 5$ $i = 5$
State: $\ell = 5$ $i = 6$
State: $\ell = 5$ $i = 7$
State: $\ell = 4$ $i = 1$
State: $\ell = 3$ $i = 1$
intermediate shape

4.2 Existence of Suitable Drawing Paths

In the previous section, we showed that a drawing formation $F^{(i)}$ which is placed in $C^{(i)}$ can traverse a drawing path $\boldsymbol{v}$ of $P^{(i)}$ if $F^{(i)}$ is compatibiliy to $\boldsymbol{v}^{(i)}$. We omited all details on how we can create such a drawing path. In this section, we will construct a path and prove that it fulfills all required properties of Lemma 4.6. The final proof of Lemma 4.6 can be found in the end of Section 4.2.2.

Outline. For a symmetric component $P^{(1)}$, we define a tree $T^{(1)}$ with $\mathcal{O}(n)$ nodes such that its nodes cover all points of $P^{(1)}$. The tree's root node will be $(2\Delta, \pi/\operatorname{sym}(P))$, the initial position of the drawing formation $F^{(1)}$ aligned to $P^{(1)}$. It is clear that a simple traversal of $T^{(1)}$ will fulfill the requirement (1) and (4) of compatibility. To additionally fulfill (2) and (3), we construct a tail that fits the requirements and append it to the traversal. To prove that such a tail always exists we show, that it is always possible to rotate P s.t. $P^{(1)}$ contains ≥ 3 connected positions.[3] We use these three positions to append a tail to $T^{(1)}$ that fulfills the requirement of compatibility. An illustration of the construction of a drawing tree can be found in Section 4.2.3.

4.2.1 Constructing a Drawing Tree

Definition 4.17 — Drawing-Tree. Let $P^{(1)}$ be a symmetric component of P. We call $T^{(1)}$ constructed with algorithm Algorithm 8 a *drawing-tree* of $P^{(1)}$.

Algorithm 8 ConstructDrawingTree($P^{(1)}$)

```
$root \leftarrow (2\Delta, \pi/\operatorname{sym}(P))$
$Base_{left} \leftarrow$ linear line $(2\Delta, \pi/\operatorname{sym}(P)) + i \cdot (4\delta, 2\pi/\operatorname{sym}(P))$ for $i \in \{1, \cdots, \infty\}$ starting
  at $root$
$Base_{right} \leftarrow$ linear line $(2\Delta, \pi/\operatorname{sym}(P)) + i \cdot (4\delta, 0)$ for $i \in \{1, \cdots, \infty\}$ starting at $root$
$T^{(1)} \leftarrow root + Base_{left} + Base_{right}$                                    ▷ Base Tree
while $cov(T^{(1)}) \neq P^{(1)}$ do                                   ▷ grow the tree inside the cone
    let $(p,t)$, $p \in P^{(1)} \setminus cov(T^{(1)})$, $t \in T^{(1)}$ be the pair with minimal distance
    if $dist(p,t) < 1 - \delta$ then
        add $p$ to $T^{(1)}$ and connect it to $t$
    else
        $t' \leftarrow (p+t)/2$                                ▷ Intermediate node between $p$ and $t$
        add $t'$ to $T^{(1)}$ and connect it to $t$; add $p$ to $T^{(1)}$ and connect it to $t'$
remove all subtrees of $T^{(1)}$ that do not cover any $p \in P^{(1)}$ and return $T^{(1)}$
```

It is clear that $T^{(1)}$ is computable from P. The nodes cover all positions in $P^{(1)}$. Distances between neighboring nodes are 4δ on the linear lines and at most $1 - \delta$ everywhere else. So $T^{(1)}$ fulfills all requirements for a drawing path for $\delta \leq 0.2$.[4]

[3]While the pattern is connected by definition, the cuts in symmetric components can disconnect parts of the component. E.g. if the pattern is a multi-helix spiral

[4]It holds for $\delta \leq 0.2$ that $4\delta \leq 1 - \delta$

Observation 4.4 A reasonable short and deterministically computable traversal of $T^{(1)}$ as defined in Definition 4.17 is a *drawing path* for $\delta \leq 0.2$. We define $trav(T^{(1)})$ to represent the path of this traversal.

4.2.2 Appending a Tail

$trav(T^{(1)})$ is a drawing path, but it is not necessarily compatible to a drawing formation because the tail does not always fulfill the requirements of Definition 4.12. The tail is the last part of the path, which covers in total ≤ 3 pattern positions (Definition 4.11). For the compatibility, it must have a length that is traversable by a drawing formation of 3 robots, i.e. length $\leq \left|A_F^3(\varepsilon)\right|$. Additionally, all positions covered by the tail must be in the distance ≤ 1 to the last node of the tail. This allows the remaining 3 robots of the drawing formation to reach the last three pattern positions from the last node of the tail. In Lemma 4.9 we first show that there exists suitable start point q_{start} and end point q_{end} for such a tail. q_{end} is a suitable end point, when it has at least 3 positions $p_1, p_2, p_3 \in P^{(1)}$ in its 1-surrounding. q_{start} must have a constant distance to q_{end} and cover at least one additional position $p_4 \in P^{(1)} \setminus \{p_1, p_2, p_3\}$. To prove Lemma 4.6 we construct a compatible drawing path out of q_{start}, q_{end} and $T^{(1)}$. We connect q_{start} and q_{end} with intermediate nodes in a straight line and add possibly up to 3 additional intermediate nodes around q_{end} to cover the positions p_1, p_2 and p_3. q_{start} is connected with a straight line of intermediate nodes to the end of $T^{(1)}$ as well. Because q_{start} and q_{end} have a constant distance and cover, together with the intermediate nodes, at least the positions $p_1, \cdots, p_4$ we know that the tail fulfills the requirements mentioned above.

Lemma 4.9 Let P be a pattern with symmetricity $\mathrm{sym}(P) < |P|/2$. There exist for each symmetric component $P^{(1)}$ with cone $C^{(1)}$ points q_{start} and q_{end} such that

1. $|\mathcal{B}(q_{end}, 1)| \geq 3$
2. if $\left|P^{(i)}\right| > 3$
   a. $|\mathcal{B}(q_{start}, 1-\delta) \cup \mathcal{B}(q_{end}, 1)| \geq 4$
   b. $dist(q_{start}, q_{end}) = \mathcal{O}(1)$
   c. $dist(\{q_{start}, q_{end}\}, \partial C^{(1)}) = \Omega(1/\mathrm{sym}(P) + \mathrm{mindist}(P))$

Proof. **(1)** Consider a finite subset $S \subset \mathbb{R}^2$ of symmetricity s and size $|S| \geq 3s$. Assume the unit disc graph $\mathcal{U}(S)$ is connected. It is a geometric fact that there exists a subset $C \subseteq S$ of size $|C| = 3$ and with $\angle(C) < 2\pi/s$ such that $\mathcal{U}(C)$ is connected. We stated and proved this as an auxiliary result in the end of this section (Lemma 4.10). Our pattern P is such a set and a symmetric component $P^{(1)}$ is a subset with $\angle(C) < 2\pi/s$, therefore there exists a rotation of P such that $p_1, p_2, p_3 \in P^{(1)}$ with $\mathcal{U}(\{p_1, p_2, p_3\})$ connected. W.l.o.g. we assume this is the rotation of P. Then, there exists a point q_{end} with $\mathcal{B}(q_{end}, 1) \supseteq \{p_1, p_2, p_3\}$.

Trivial cases for (a), (b) and (c). If $P^{(1)}$ contains 4 positions with $\mathcal{U}(\{p_1, p_2, p_3, p_4\})$ connected, it is clear that q_{start} and q_{end} can be placed with $\mathcal{B}(q_{start}, 1-\delta) \cup \mathcal{B}(q_{end}, 1) \supseteq \{p_1, p_2, p_3, p_4\}$ (a). If $|\mathcal{B}(q_{end}, 1-\delta)| \leq 3$ (w.l.o.g. $p_4 \notin \mathcal{B}(q_{end}, 1-\delta)$), we place q_{start} in distance $\leq 1-\delta$ to p_4 (b). q_{start} and q_{end} have a constant distance (c). If $\mathcal{B}(q_{end}, 1)$ contains more than three positions that are not all connected, the placement for q_{start} is analog.

(a) and (b) We assume that $\mathcal{B}(q_{end}, 1) = \{p_1, p_2, p_3\}$. Because $\mathcal{U}(P)$ is connected, there exists a path from p_1 to $p' \in P^{(1)} \setminus \{p_1, p_2, p_3\}$ in $\mathcal{U}(P)$. Let $p_1, w_1 \cdots, w_k, p'$ be a shortest path form p_1 to p'. The path can only contain 3 consecutive nodes in one symmetric component (otherwise the component would contain 4 connected positions), therefore there exist $w_j \in P^{(2)}$ with $j \leq 3$ and $w_l \notin P^{(2)}$ with $l \leq j+3$. If $w_l \in P^{(1)}$, we find $p_4 = w_l$ with $dist(p_1, p_3) \leq 6$. If $w_l \in P^{(3)}$, there exist a rotational symmetric point in $P^{(1)}$, let this be p_4. It is clear that $dist(p_1, p_4) \leq dist(p_1, w_l) \leq 6$. We can place q_{start} in the distance $1-\delta$ of p_4 to fulfill (a) and (b).

(c) q_{start} can be placed relatively freely in a radius of $1-\delta$ around p_4, this easily fulfill (c). There exist cases where q_{end} must be placed directly on one of the three connected positions, let this be p_1. From Lemma 4.10 we can follow, that w.l.o.g. p_{end} lies on the bisector of $C^{(1)}$. Let $p'P^{(2)}$ be the rotational symmetric point to p_1. Naturally, $dist(q_{end}, \partial C^{(1)}) = \frac{1}{2} dist(p_1, p') \geq mindist(P)$. ■

Lemma 4.6 — restate. Consider the ε-granular drawing formation $F^{(1)}$ of the initial drawing pattern for a connected pattern P of symmetricity $\mathrm{sym}(P) < |P|/2$. The parameter ε can be chosen such that $F^{(1)}$ has $\left|L_{F^{(1)}}(\varepsilon)\right| \geq 2 + \left|P^{(1)}\right|$ ε-granular locations. Moreover, there exists a drawing path $\boldsymbol{v}^{(1)}$ of symmetric component $P^{(1)}$ that is compatible with $F^{(1)}$ such that

1. the path $\boldsymbol{v}^{(1)}$ starts in the anchor $a = (2\Delta, \pi/\mathrm{sym}(P))$ of $F^{(1)}$,
2. $\boldsymbol{v}^{(1)}$ lies in the first cone (i.e., $\boldsymbol{v}^{(1)} \subseteq C^{(1)}$) and

$$\mathrm{dist}\left(\boldsymbol{v}^{(1)}, \partial C^{(1)}\right) > \max\{\varepsilon, \Delta \cdot \sin(2\pi/\mathrm{sym}(P))\} \tag{4.4}$$

.

Proof. Let $T^{(1)}$ be the drawing tree of $P^{(1)}$ Definition 4.17. $trav(T^{(1)})$ has all properties for a drawing path (Observation 4.4). Let $F^{(1)}$ be a drawing formation with $\varepsilon = \Theta(\min(1/\mathrm{sym}(P), \mathrm{mindist}(P), 1/\sqrt{|P|})$ and $\Delta = 0.1$ and $\Phi = 2\pi/\mathrm{sym}(P)$. To make $trav(T^{(1)})$ compatible with $F^{(1)}$ we append the points q_{start} and q_{end} from Lemma 4.9. We add $b = \mathcal{O}(P)$ itermediate nodes $w_1, \cdots, w_b$ between the end of $trav(T^{(1)})$ and q_{start}. We add $b' = \mathcal{O}(1)$ intermediate nodes $w_{b+1}, \cdots, w_{b+b'}$ between q_{start} and q_{end}. The resulting path is

$$\boldsymbol{v}^{(1)} := trav(T^{(1)}) + (w_i)_{i=0}^{b} + (q_{start}) + (w_i)_{i=b+1}^{b+b'} + (q_{end})$$

We make sure, that

$$\mathrm{cov}((w_i)_{i=b+j}^{b+b'} + (q_{end})) \subseteq \mathcal{B}(q_{end}, 1) \text{ with } |\mathrm{cov}((w_i)_{i=b+j}^{b+b'} + (q_{end}))| \geq 3$$

for $1 \leq j \leq b'$. This is possible by placing up to 3 intermediate nodes in distance $\leq \delta$ to q_{end}. This number of nodes is sufficient to reach q_{start}, respectively q_{end}, with distances $\leq 1-\delta$ between w_i and w_{i+1}, because q_{start} has a distance $\mathcal{O}(|P|)$ from any node of $trav(T^{(1)})$ and q_{end} has a constant distance from q_{start} (see Lemma 4.9). There obviously exist deterministic methods to define the intermediate paths, chose q_{end}, q_{start}, and determine $trav(T^{(1)})$ with $hops(trav(T^{(1)})) = \mathcal{O}(|P|)$.

Compatibiliy We show that conditions (1) - (4) from Definition 4.12 are fullfiled. **(1)** with $\delta = 0.1$ this is fullfiled **(2)** From Lemma 4.9 we know that $|\mathcal{B}(q_{end})| = 3$. From the equation in the beginning of this proof follows $\mathrm{cov}(tail(v^{(1)})) \subseteq \mathcal{B}(q_{end})$, **(3)** If $|P^{(1)}| \geq 4$ than $|\mathcal{B}(q_{start}, 1-\delta) \cup \mathcal{B}(q_{end}, 1)| \geq 4$ (by Lemma 4.9 (2)) Hence, the tail of $v^{(1)}$ must start after q_{start}. $|tail(v^{(1)})| = \mathcal{O}(1)$ in this case (3) is fulfilled (see Lemma 4.1). If $|P^{(1)}| = 3$ we know that $\varepsilon = \mathcal{O}(1/|P|)$. By Lemma 4.1) follows that $\mathcal{A}_F^3(\varepsilon) = \Omega(|P|)$. $tail(v^{(1)}) = v^{(1)}$ with length $\mathcal{O}(|P|)$ in this case. This fulfills (3). **(4):** With $\varepsilon = \mathcal{O}(1/\sqrt{|P|})$ we have $|A_F^4(\varepsilon)| = \Omega(|P|)$ (see Lemma 4.1). We have $|v^{(1)}| = \mathcal{O}(|P|)$. This fulfills (4)

Requirements (1) and (2) of Lemma 4.6. (1) the root node of $T^{(1)}$ has coordinate $(2\Delta, \pi/\operatorname{sym}(P))$ and is the start of $v^{(1)}$. (2) By construction of Definition 4.17 is clear that $dist(t, \partial c(1)) \geq dist((2\Delta, \pi/\operatorname{sym}(P)), \partial c(1)) = \Delta \cdot \sin\left(\frac{2\pi}{\operatorname{sym}(P)}\right)$ for $t \in T^{(1)}$. By Lemma 4.3 we know that

$$dist(z_i, \partial C^{(1)}) = \Omega(\operatorname{mindist}(P)), i \in \{1,2\} \quad \text{and}$$

$$\varepsilon < \min(1/\operatorname{sym}(P), \operatorname{mindist}(P), 1/\sqrt{|P|}) \cdot \Delta.$$

With this choice of ε we can create a ε-granular drawing formation that has more locations that $|P^{(1)}| + 2$.

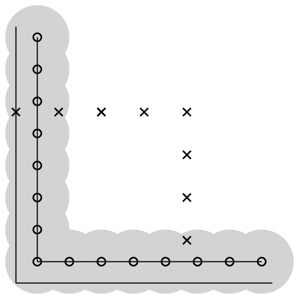
Such that F is an ε-granular drawing formation, the parameter Δ, ϕ and ε must be known. Δ and Φ are given above and computable from P.

$$\varepsilon = \min(1/\operatorname{sym}(P), \operatorname{mindist}(P), 1/\sqrt{|P|}) \cdot c$$

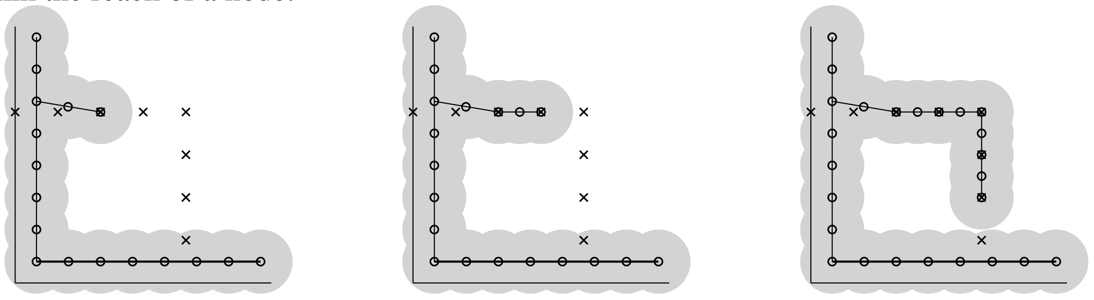
for a constant $c < 1$. The constant can be deterministically determined (but we never write it down). ■

4.2.3 Illustration of the Drawing Path Construction

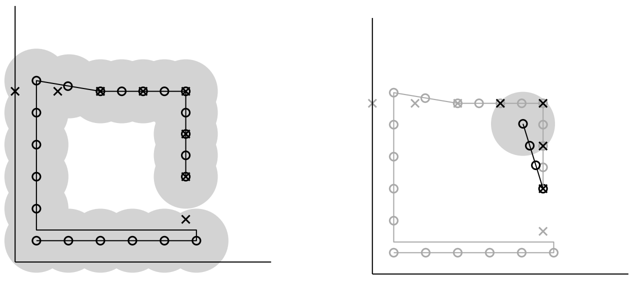
The construction of a drawing path for one symmetric component is illustrated in the following. It considers a square as the pattern P; its symmetricity is $\operatorname{sym}(P) = 4$ and is split into 4 symmetric components. The cone of the symmetric component has an angle of 90°. As a first step, the "Base Tree" along the cone is created. Nodes are depicted as $\circ$ and pattern positions as $\times$. The gray area around a node depicts the distance, that can be reached by robots in a drawing formation on that node.



Next, the tree "grows" iteratively towards the nearest pattern position. Two steps are exemplary illustrated, the third image depicts the final drawing tree after all positions are within the reach of a node.



Afterwards, all subtrees where no nodes are within reach of a pattern position are removed and a traversal of the drawing tree is computed. To get the final drawing path, a fitting tail is computed and appended in the end.



4.2.4 Auxiliary Result

Lemma 4.10 Consider a finite connected subset $S \subset \mathbb{R}^2$ of symmetricity $s := \operatorname{sym}(S) \in \mathbb{N}$ and size $|S| \geq 3s$. There exists a subset $C \subseteq S$ of size $|C| = 3$ and with $\angle(C) < 2\pi/s$ such that C is connected.

Proof. For $s = 1$ the statement is trivial. So assume $s > 1$, such that $2\pi/s \leq \pi$. Consider the points from S in polar coordinates, where we consider angular coordinates on $(-\pi, \pi]$. Since S is connected and $|S| \geq 3$, there are three points $p = (r, \phi), q_1 = (r_1, \phi'_1), q_2 = (r_2, \phi'_2) \in S$ with $\operatorname{dist}(p, q_1) \leq 1$ and $\operatorname{dist}(p, q_2) \leq 1$. Because S has symmetricity s, without loss of generality we can assume $\phi \in [0, 2\pi/s)$ (otherwise we find corresponding rotation-symmetric points for p, q_1, and q_2). Using again the symmetricity, for each $i \in \{1, 2\}$ we find a $p_i = (r_i, \phi_i) \in S$ with $\phi_i \in [0, 2\pi/s)$.

For $i \in \{1, 2\}$ and $k \in K := \{0, 1, \ldots, s-1\}$, define $p_{i,k} := p_i + (0, k \cdot 2\pi/s) =: (r_i, \phi_{i,k})$ and set $S_i := \{p_{i,k} \mid k \in K\} \subseteq S$.[5] In other words, S_i is the set of all points in S that are rotation-symmetric to p_i. In particular, $q_i \in S_i$. For any $p_{i,k} \in S_i$, the distance formula for points in polar coordinates yields

$$\operatorname{dist}(p, p_{i,k})^2 = r^2 + r_i^2 - 2r \cdot r_i \cdot \cos\left(\phi - \phi_i - k \cdot \frac{2\pi}{s}\right). \tag{4.5}$$

By choice of p_i, we have $|\phi - \phi_i| \in [0, 2\pi/s)$. The right-hand side of Equation (4.5) is minimized at the unique $k_i \in K$ for which $|\phi - \phi_i - k_i \cdot 2\pi/s| \in [0, \pi/s)$. More exactly, we have

- $k_i = 0$ if $|\phi - \phi_i| \leq \pi/s$,
- $k_i = 1$ if $\phi - \phi_i \geq 2\pi/s$, and
- $k_i = s-1$ if $\phi - \phi_i \leq -2\pi/s$.

In any case, the choice of k_i yields $\angle(p, p_i) = \left|\phi - \phi_{i,k_i}\right| < \pi/s$ as well as (together with $q_i \in S_i \subseteq S$) $\operatorname{dist}(p, p_{i,k_i}) \leq \operatorname{dist}(p, q_i) \leq 1$.

Now consider the three element set $C := \{p, p_{1,k_1}, p_{2,k_2}\} \subseteq S$. As shown above, $\operatorname{dist}(p, p_{i,k_i}) \leq 1$ for both $i \in \{1, 2\}$, so C is connected. Moreover,

$$\angle(C) \leq \angle(p, p_{1,k_1}) + \angle(p, p_{2,k_2}) < 2\pi/s. \tag{4.6}$$

Thus, we found a set C with the required properties. ■

[5]Remember that we consider angular coordinates modulo the interval $[-\pi, \pi)$, such that $\phi_{i,k} \in [-\pi, \pi)$ for all $i \in \{1, 2\}$ and $k \in K := \{0, 1, \ldots, s-1\}$.

4.3 Description of the Pattern-Formation Protocol

Even though we have proven that a protocol for ARBITRARY-PATTERN-FORMATION exists in this chapter, its description is split throughout many sections and done implicitly during proofs. Some parts of the protocol are proven to be possible without even defining how to do them. In this section we like to present the protocol in a complete and more streamlined version with pseudocode.

The main algorithm of the protocol is PATTERNFORMATION(P, r, neighborhood). It is described in the proof of Theorem 8. Every robot executes this algorithm in each round. For better readability, we pass the executing robot r and its neighborhood (all robots withing viewing range of r) to the algorithm. The algorithm call several subroutines that are presented afterwards. All subroutines are derived from definitions, lemmas, and proofs of this chapter. Note that, for a better readability, we assume P as well as the below computed parameters $\varepsilon, \Delta, \phi$ and δ as global variables and do not pass them to every subroutine.

Algorithm 9 PATTERNFORMATION(P, r, neighborhood)

```
$P^{(1)} \leftarrow$ GETSYMMETRICCOMPONENT($P$)
$\varepsilon, \Delta, \phi, \delta \leftarrow$ COMPUTEPARAMETERS($P$)
$\mathbf{v}^{(1)} \leftarrow$ CONSTRUCTDRAWINGPATH($P^{(1)}$)
$F$, phase $\leftarrow$ COMPUTEPHASEANDDRAWINGFORMATION($r$, neighborhood)
**if** phase = “robot is in Diam-1-Configuration” **then**
    FORMINITIALDRAWINGPATTERN($P^{(1)}, r$, neighborhood)
    execute formation of initial drawing pattern (Definition 4.16)
**else if** phase = ““robot is in drawing formation” **then**
    MOVEDRAWINGFORMATION($\mathbf{v}^{(1)}, F, r$, neighborhood)
**else if** phase = “in intermediate shape” **then**
    EXECUTEMOVEOFLASTTHREEROBOTS($\mathbf{v}^{(1)}, F, r$, neighborhood)
**else if** phase = “robot is on final position” **then**
    do nothing
```

A symmetric component (Definition 4.14) can be computed easily by simply splitting the pattern at its SEC-center in cake-pices. In Lemma 4.10 is state, that there always exist a rotation of P such that $P^{(1)}$ contains three positions where the unit disc graph (regarding unit distance 1) is connected (see also the proof of Lemma 4.9). The following algorithm makes sure, that P is rotated accordingly. Note, that the parameter P has always the exact same value (i.e. the positions are in the same order) for all robots and all rounds. Therefore, the following algorithm always produces the exact same value of $P^{(1)}$.

Algorithm 10 GETSYMMETRICCOMPONENT(P)

```
translate $P$ such that its SEC-center is at the origin
$index \leftarrow 0$
$P^{(1)} \leftarrow$ as defined in Definition 4.14
**while** $P^{(1)}$ contain not three connected positions **do**
    $P \leftarrow$ rotate $P$ such that $p_{index}$ is on the positive $X$-axis
    $index \leftarrow index + 1$
    $P^{(1)} \leftarrow$ as defined in Definition 4.14 in the current rotation of $P$
**return** $P^{(1)}$
```

Our protocol uses an ε-granular drawing formation (Definition 4.6) that traverses a δ-drawing path (Definition 4.9). Such that this traversal can be executed both must be compatible (Definition 4.12) with each other. Lemma 4.6 states, that it is possible to create a compatible pair. To guarantee this, we compute $\varepsilon, \phi, \delta$ and Δ and use these in the computation of the drawing path as well as the computation of the drawing formation. The algorithm below contains the constant factor c that is not further specified. It is composed of the following. The number of states in the drawing formation must be least the length of the drawing path. Note, that this is a relevant requirement for the compatibility because the steps along the path are counted via the states. During the proof of Lemma 4.6 (see Section 4.2.2, paragraph *Compatibility* in the proof) we have shown that the drawing formation has at least $\left|A_F^4(\varepsilon)\right|$ states. In Lemma 4.1 we estimate the sizes of $\left|A_F^4(\varepsilon)\right|$ with $\Omega(\Delta^3 \cdot \phi / \varepsilon^3)$. We know that $\phi \geq 1/\operatorname{sym}(P)$ and Δ is a constant. With $\varepsilon \leq 1/\operatorname{sym}(P)$ and $\varepsilon \leq 1/\sqrt{|P|}$ as chosen below we know that $\left|A_F^4(\varepsilon)\right| \in \Omega(|P|)$. The length of the drawing path that is computed in ConstructDrawingPath($P^{(1)}$) is in $\mathcal{O}(|P|)$. To get a fitting value of c it is sufficient to start with $c = 1$ and count $\left|A_F^4(\varepsilon)\right|$, and the length of the drawing path. If $\left|A_F^4(\varepsilon)\right|$ is to small one can double c and try again. It only needs constant many repeats to reach a fitting c.

Algorithm 11 ComputeParameters(P)

```
$\Delta \leftarrow 0.1$
$\delta \leftarrow 0.5$
$\varepsilon \leftarrow \min(1/\operatorname{sym}(P), \operatorname{mindist}(P), 1/c\sqrt{|P|})$        $\triangleright$ constant $c$ as described above
$\phi \leftarrow \min(2\pi/\operatorname{sym}(P), \pi/3)$
return $\varepsilon, \Delta, \phi$
```

The following subroutine describes the construction of a drawing path as described in Lemma 4.6. It is also illustrated in Section 4.2.3. We do not further specify how to compute the traversal of the drawing tree $T^{(1)}$. As stated in Observation 4.4 any reasonable short traversal (length in $\mathcal{O}(|T^{(1)}|)$) will do as long it is deterministically computable (returns exactly the same path each time).

Algorithm 12 ConstructDrawingPath($P^{(1)}$)

```
$T^{(1)} \leftarrow$ ConstructDrawingTree($P^{(1)}$)        $\triangleright$ Algorithm 8, the drawing tree of $P^{(1)}$
$trav \leftarrow$ traversal of $T^{(1)}$
$tail \leftarrow$ ConstructFittingTail($P^{(1)}, P, trav, \Delta, \varepsilon$)
return $traversal + tail$
```

The construction of the tail is described in the beginning of Section 4.2.2. In GetSymmetricComponent($P^{(1)}$) we made sure, that $P^{(1)}$ contains at least 3 positions that are connected in the unit disc graph. One of these positions, we call it p_{end}, must be connected to the other both. The following algorithm searches p_{end} iteratively to end the tail there. Note, that the proof of Lemma 4.9 requires that p_{end} is on the bisector of $C^{(1)}$ (the cone enclosing $P^{(1)}$, see Definition 4.14). W.l.o.g. we assume, that $P^{(1)}$ is rotated accordingly (this could be easily implemented in GetSymmetricComponent($P^{(i)}$) but is omitted there for readability). q_{start} is the start of the tail. As required in Lemma 4.9 it must have a constant distance to p_{end}, be near a fourth position and sufficiently fare away from the boundary of the cone $C^{(1)}$. We have shown in the proof Lemma 4.9 that there always exists a fourth position $p_{start} \in P^{(1)}$ with distance ≤ 6 to p_{end} (see paragraph *(a) and (b)*). We place q_{start} within distance $1-\delta$ to p_{start} such that it fulfills all the conditions of Lemma 4.9.

Algorithm 13 CONSTRUCTFITTINGTAIL($P^{(1)}$,$trav$)

```
for $p_{end}, p, p' \in P^{(1)}$ do
    if $dist p_{end}, p \leq 1$ and $dist p_{end}, p' \leq 1$ then
        $q_{end} \leftarrow p_{end}$
        $p, p'$ are stored
        end of for-loop
for $p_{end} \in P^{(1)}$ do
    if $dist(p_{start}, p_{end}) \leq 6$ and $p_{start} \neq p_{end}$ and $p_{start} \neq p$ and $p_{start} \neq p_{end}$ then
        $p_{end}$ is stored
        end of for-loop
$d_{max} \leftarrow \max\{\varepsilon, \Delta \cdot \sin(2\pi/\mathrm{sym}(P))\}$      ▷ see Lemma 4.5 for an explanation of the value
if $dist(\{p_{start}\}, \partial C^{(1)}) < d_{max}$ then      ▷ distance to boundary $C^{(1)}$ to small
    if bisector of $C^{(1)}$ is within distance $1/2$ to $p_{start}$ then
        $q_{start} \leftarrow p_{start}$ projected orthogonal on the bisector of $C^{(1)}$
    else$q_{start} \leftarrow p_{start}$ moved $1/2$ towards the bisector of $C^{(1)}$
else
    $q_{start} \leftarrow p_{start}$
$pathToTail \leftarrow$ direct path from last node of $trav$ to $q_{start}$ with nodes in distance $1-\delta$
$tail \leftarrow$ direct path from $q_{start}$ to $q_{end}$ with nodes in distance $1-\delta$
return $pathToTail + tail$
```

We defined in Lemma 4.8 that we can distinguish four situations (called phases) which we can label with "robot is in Diam-1-Configuration", "robot is in drawing formation", "robot in intermediate shape", and "robot is on final position". The following algorithm computes, in which of these phases a robot is. A central part of this computation is the detection of a drawing formation in the neighborhood, this is defined as a separate subroutine below.

Algorithm 14 COMPUTEPHASEANDDRAWINGFORMATION(r, neighborhood)

```
$F$, phase $\leftarrow$ COMPUTEDRAWINGFORMATION($(\varepsilon, \Delta, \phi), r$, neighborhood)
if $|$neighborhood$| = |P|$ and $dist$(neighborhood) $\leq 1$ then
    if neighborhood is a configuration that fits to the state of $F$ then
        pass      ▷ the phase check is continued below
    else
        return $\emptyset$, "robot is in Diam-1-Configuration"
if phase = "robot is in drawing formation" then
    return $F$, "robot is in drawing formation"
else if phase = "robot is in intermediate shape" then
    return $F$, "robot is in intermediate shape"
else
    return $\emptyset$, "robot is on final position"
```

The following algorithm is used such that robots that are part of a drawing formation compute the drawing formation to determine the current drawing formation Observation 4.2. If a robot is not part of a drawing formation, the algorithm will return $\emptyset$, therefore it can be used to check whether a robot is part of a drawing formation.

Algorithm 15 ComputeDrawingFormation(r, neighborhood)

```
$pairs \leftarrow$ detect all pairs $(r_1, r_2)$ with $\mathrm{dist}(r_1, r_2) = \varepsilon$ in neighborhood
found-drawing-formation $\leftarrow$ **None**
**for** all $(r_1, r_2) \in pairs$ **do**
    $r_3 \leftarrow$ find colinear robot in distance $\leq \Delta$ to $r_1$ and $r_2$
    **if** $r_3$ exists **then**
        $a, \mathbf{d} \leftarrow$ compute anchor and direction of the defining robots $r_1, r_2, r_3$ (Section 4.1.2)
        $H \leftarrow$ drawing hull with $a, \mathbf{d}, \Delta, \phi$ (Definition 4.1)
        $\mathcal{R} \leftarrow$ robots within $H$                 $\triangleright$ state robots, seeDefinition 4.2
        $F \leftarrow (\mathcal{R}, H)$               $\triangleright$ $\varepsilon$-granular drawing formation (Definition 4.6)
        **if** $r$ is in $H$ **then**
            **return** $F$, "robot is in drawing formation"
        **else**found-DF $\leftarrow F$
    **else**                  $\triangleright$ this is used for ExecuteMoveOfLastThreeRobots
        find third robot in distance $\varepsilon$
        $F_{inter} \leftarrow$ intermediate shape as defined in Definition 4.13
        **if** $r \in F_{inter}$ **then**
            **return** $F_{inter}$, "robot is in intermediate shape"
        **else**found-DF $\leftarrow F_{inter}$
    **return** found-drawing-formation, "robot is not in drawing formation"
```

The initial drawing pattern I of pattern P is defined in Definition 4.16. It is a pattern with symmetricity $\mathrm{sym}(P)$. Initially, the robots have a global visibility because the diameter of the swarm is smaller than the viewing range. Therefore, the protocol from [69] can be applied to form it. The following algorithm applies this protocol. A central part of this protocol is the computation of a common coordinate system that is independent of the local orientation of robots. A separate subroutine translates the neighborhood is such an independent coordinate system. Note, that GetSymmetricComponent(M) always returns the same $Q^{(1)}$ if the same value M is passed. It is necessary that r is in $Q^{(1)}$. If this is not the case, $Q^{(1)}$ is rotated accordingly by a multiple of $2\pi/\mathrm{sym}(\mathbf{z}^0)$ (see while-loop below).

Algorithm 16 FormInitialDrawingPattern(r, neighborhood)

```
$N \leftarrow$ list of coordinates in neighborhood (arbitrary order, in local coordinate system)
$M, center, rot \leftarrow$ ComputeCommonCoordinateSystem($N$)
$Q^{(1)} \leftarrow$ GetSymmetricComponent($M$)
translate local coordinate system that $center$ is the origin
rotate the local coordinate system by $rot$
**while** $r$ not in $Q^{(1)}$ **do**
    rotate $Q^{(1)}$ by $2\pi/\mathrm{sym}(M)$
$I \leftarrow$ initial drawing pattern as defined in Definition 4.16
```

```
    translate $I$ such that it is centered at the origin
    $I^{(1)} \leftarrow$ cut $I$ with the cone of $Q^{(1)}$
    $num \leftarrow$ index of $p'$ in $Q^{(1)}$ with $p' = r$
    $p \leftarrow$ position with index $num$ in $I^{(1)}$
    $r$ **moves** onto position $p$
```

The following subroutine computes a unified translation, rotation and order of the coordinate list N. This makes the coordinates independent of the local position and orientation of the robot. If N has a symmetricity > 1, there is no unambiguous deterministically computable rotation rot_{min} as described below. However, there are multiple rot_{min} that are rotations of each other by a multiple of $2\pi/\operatorname{sym}(N)$. It is a key property of the symmetricity (Definition 1.2) that all such rot_{min} leads to exactly the same value of N_{rot} below. Note, that the concrete chosen and returned rot_{min} has a relevance for the subroutine above (the while-loop there corrects it).

Algorithm 17 ComputeCommonCoordinateSystem(N)

```
    $center \leftarrow$ the center of the SEC of $N$
    $N_{centered} \leftarrow$ translate $N$ such that $center$ is the origin
    $lex_{min} \leftarrow$ smallest $lex$ of the following loop
    for $p \in N$ do
        $rot \leftarrow$ rotation such that $p$ is on the positive $X$-axis
        $N' \leftarrow N_{centered}$ rotated accordingly, coordinates lexicographical sorted
        $lex \leftarrow$ lexicographical representation $N'$
    $rot_{min} \leftarrow rot$ with $lex = lex_{min}$ of the previous loop
    $N_{rot} \leftarrow$ rotate $N_{centered}$ by $rot_{min}$
    return $N_{rot}, center, rot_{min}$
```

The following algorithm describes how each individual robot inside the drawing formation computes the movement of the whole drawing formation as well as where the individual robot has to move. We defined in Observation 4.1 that a drawing formation F can be moved by mapping state robots of the drawing formation in round t to the state robots in round $t+1$ and pattern positions nearby ("drop-coordinates"). For the last step, robots form an intermediate shape F_{inter} as defined in Definition 4.13 (see subroutine Algorithm 19 for more explanation).

Algorithm 18 MoveDrawingFormation($\boldsymbol{v}^{(1)}, F, r$)

```
    let $F = (\mathcal{R}, a, \mathbf{d}, \Delta, \phi)$          ▷ $a$ is the anchor and $\mathcal{R}$ the set of state robots
    $num \leftarrow$ number of state $\mathcal{R}$     ▷ see "Enumeration of States" in the proof of Lemma 4.2
    $i \leftarrow$ index of robot $r$ in the list of state robots $\mathcal{R}$
    let $\boldsymbol{v}^{(1)} = (v_1, \cdots, v_k)$
    if $num < k$ then                                  ▷ anchor is not on the last node
        $\mathcal{R}^+ \leftarrow$ state with number $num+1$            ▷ state of the next drawing formation
        $move \leftarrow v_{num+1} - v_{num}$
        $a^+ \leftarrow a + move$                           ▷ anchor of the next drawing formation
        $F^+ \leftarrow (\mathcal{R}^+, a^+, \mathbf{d}, \Delta, \phi)$                      ▷ the next drawing formation
        if $v_{num} \notin \operatorname{tail}(\boldsymbol{v}^{(1)})$ then
            drop-coordinates $\leftarrow \{p - v_i$ for $p \in \operatorname{cov}(v_i)\}$
        else
            drop-coordinates ← $\emptyset$
        if $i \leq |\mathcal{R}^+|$ then          ▷ robot $r$ will be part of the next drawing formation
            $r$ moves onto coordinates of state robot with index $num$ in $\mathcal{R}^+$
        else                                   ▷ robot $r$ will move onto one of the drop-coordinates
            $r$ moves onto drop-coordinate number $i - |\mathcal{R}+|$
    else if $num = k$ then                                          ▷ anchor is on the last node
        $F_{inter}$ ← intermediate shape on $a$ as defined in Definition 4.13
        $r$ moves onto the coordinate of index $i$ in $F_{inter}$
```

Reaching the last three positions of a symmetric component is different from the movement onto a “drop-coordinate” as done in the algorithm above. The nodes of the traversal path are always placed near to the positions such that each robot can reach it with a movement of distance ≤ 1. For the last three positions it may happen, that robots need to move a distance of $1+\Delta$, this is not possible in one round. Because robots have no memory, they cannot simply remember how to continue the movement in the next round. We defined in the proof of Lemma 4.3 how the last three positions of a symmetric component can be reached with an intermediate step Definition 4.13 that, similar to a drawing formation state, can be locally recognized by the robots in the next round to continue the movement onto the desired pattern positions (see Lemma 4.4). During TRAVERSEDRAWINGPATH above, the intermediate shape is formed if the drawing formation is on the last node. Note, that the intermediate shape is designed in a way that each of the three robots is nearest to the position where it has to move. The following algorithm uses this property to move the robots onto these positions.

Algorithm 19 EXECUTEMOVEOFLASTTHREEROBOTS($v^{(1)}$, $F_{inter}r$, neighborhood) Lemma 4.3

```
    let $positions$ be the remaining three position
    let $r'$ and $r''$ be the other two robots in $F_{inter}$
    for $p \in positions$ do
        if $dist(p,r) \leq dist(p,r')$ and $dist(p,r) < dist(p,r'')$ then
            $r$ moves onto $p$
```

5. Conclusion and Outlook

In this thesis a (partial) solution of the ARBITRARY-PATTERN-FORMATION for disoriented robots with limited visibility was presented. It was already known that even with global visibility and memory the formation of a pattern is only possible when the symmetricity of the initial configuration divides the symmetricity of the pattern (Theorem 1). A protocol that forms any connected pattern ("large pattern") from any connected configuration ("widespread swarm") with the mentioned condition would be the best possible solution for APF with limited visibility under the known limitations (it is discussed below in Section 5.1 whether it is truly necessary that the configuration or the pattern needs to be connected). These results are fully reached with Theorem 8 for an initial configuration that has a diameter of ≤ 1 ("contracted swarms"). In Observation 1.1 it is shown that, using the same model, APF for a widespread swarm is not possible. Instead, a solution that uses one bit of memory is presented. Any NEAR-GATHERING protocol that does not change the symmetricity of the swarm can be used to solve APF because the robots can remember with one bit of memory, that they already gathered and are now executing the APF protocol above.

NEAR-GATHERING protocols are presented that contract any connected swarm such that its diameter is ≤ 1 (Chapter 2). However, these protocols may change the symmetricity of the configuration. In Section 5.2.1 we discuss the challenges that λ-contracting NEAR-GATHERING protocols face to preserve the symmetricity. Chapter 3 presents a different approach on NEAR-GATHERING protocols that preserve the symmetricity of a configuration under certain conditions: the Connectivity-Boundary of the initial configuration must be convex and must not contain holes (Theorem 7). Under these conditions, ARBITRARY-PATTERN-FORMATION is solved for disoriented robots with limited visibility. In Section 5.2.2 we discuss how the NEAR-GATHERING protocol can be adapted to preserve symmetricity for all connected configurations. A second protocol based on GO-TO-THE-AVERAGE is given that preserves symmetricity for all initial configurations. However, it does not always solve NEAR-GATHERING because robots can lose connectivity and requires some degree of global knowledge. A discussion about why connectivity cannot be preserved and how to reduce the necessity of global knowledge can be found in Section 5.2.3.

This thesis assumes the fully synchronous $\mathcal{F}$SYNC scheduler. Some results on APF

[16] show that patterns can also be formed in $\mathcal{A}$SYNC, even though this requires a stronger condition on the initial configuration. In Section 5.3 we discuss how the results of this thesis can be translated for asynchronous models.

5.1 Limited Visibility and Connectivity

In our theorems, we assume that the unit disc graph of the pattern P and the initial configuration $\mathbf{z}$ are connected. This is a natural assumption for robots with limited visibility because they cannot interact beyond their viewing range. Whenever such pattern formation is used in a real-world application, basically only connected patterns are meaningful (e.g., for creating an ad-hoc-network). $\mathbf{z}$ must always be connected. Otherwise, it cannot be guaranteed that all robots are contracted to a configuration with diameter ≤ 1. However, our protocol is capable of forming patterns with less connectivity. It applies to any pattern P where a compatible drawing path can be created. When we translate Definition 4.12 to a pattern, we get the following condition:

Let $perm(P)$ be a permutation of P. There exists $perm(P) = (p_i)_{i=1}^k$ such that

1. $\text{dist}(p_i, p_{i+1}) \leq \left|A_F^{k-i}(\varepsilon)\right|$ for $1 \leq i \leq k-2$
2. p_{k-2}, p_{k-1} and p_k must have a smallest enclosing circle of radius ≤ 1

Condition (1) follows from (1) and (2) of Definition 4.12, and condition (2) is necessary such that a drawing path can fulfill (4) of Definition 4.12.

Besides the last three robots, the maximal distance between two pattern positions is dependent on $\left|A_F^{k-i}(\varepsilon)\right|$. In the proof of Lemma 4.1 we have shown that $\left|A_F^{k-i}(\varepsilon)\right| = \mathcal{O}\left(\binom{\varepsilon^{-1}}{k-i-3}\right)$. We can choose ε freely to reach any distance necessary.

5.2 NEAR-GATHERING with Symmetricity Preservation

The discussion is split into three parts, each arguing about the possibilities of transforming one of the three NEAR-GATHERING approaches of this thesis (λ-contracting NEAR-GATHERING protocol, WAVE-PROTOCOL and ε-GO-TO-THE-AVERAGE) into a more general symmetricity preserving NEAR-GATHERING protocol.

5.2.1 Symmetricity Preservation with λ-contracting NEAR-GATHERING protocol

In [10], a class of NEAR-GATHERING functions is introduced. In general, these functions do not preserve symmetry. A simple example is an adaptation of the GO-TO-THE-CENTER protocol where robots move to the center of the smallest enclosing circle of their local neighborhood[1]. Let us assume a configuration $\mathbf{z}^t$ with symmetry. After performing GO-TO-THE-CENTER the new configuration $\mathbf{z}^{t+1}$ must also be symmetric. The smallest enclosing circle depends only on robots that reside exactly on the circle. We fix all robots in $\mathbf{z}$ that are on these circles and perturb all other robots so that the configuration is asymmetric. Let this be configuration $\breve{\mathbf{z}}^t$. After performing GO-TO-THE-CENTER the new configuration is $\breve{\mathbf{z}}^{t+1}$. Because the smallest enclosing circles have not changed, the robots in $\breve{\mathbf{z}}^t$ move onto the same positions as the robots in $\mathbf{z}^t$. Therefore, with $\breve{\mathbf{z}}^{t+1} = \mathbf{z}^{t+1}$ we can introduce symmetries. The following open problem remains.

[1] In GO-TO-THE-CENTER as defined in [2, 24] robots move only halfway towards the center.

Open Problem 1 Does a λ-contracting NEAR-GATHERING protocol exist that preserves symmetry?

5.2.2 Generalization of WAVE-PROTOCOL

A central part of our analysis is that the robots on the Connectivity-Boundary perform the same linear function during the execution. This is only the case, if the set of boundary-robots never changes. If we allow for a non-convex Connectivity-Boundary, robots move towards the outside of the swarm in concave sections. So, eventually two robots of the Connectivity-Boundary that initially have a distance > 1 will reach a distance ≤ 1. One of these robots is no longer part of the Connectivity-Boundary in the next round. One could fix this by introducing memory, but with memory symmetry preserving NEAR-GATHERING is already solved [72]. Another possible fix is to not move robots on concave parts of the Connectivity-Boundary. But eventually, these parts become convex and robots start to move according to ε-GTM which, again, changes the linear function.

The second restriction are holes. Robots at the boundary of large holes cannot distinguish their location locally from the Connectivity-Boundary. Therefore, they will start with a NEAR-GATHERING that eventually leads to a configuration, where no hole exists anymore. But this configuration can have another pre-image where the inner robots already had the position of the eliminated hole. Therefore, the protocol is not invertible in this case.

The proofs in Section 3.3 can be generalized for a class of strategies. Our core idea is to split the robots into layers (i.e. boundary, wave and inner). The outermost layer performs a NEAR-GATHERING protocol that is symmetry preserving. All other layers perform protocols, such that they stay inside the outermost layer. The outermost layer is always distinguishable and invertible. The inner layers are distinguishable and invertible, if the outer layers are known. The advantage of this class of protocols is, that you can reduce the problem of an invertible NEAR-GATHERING protocol to a restricted set of robots. However, even for a restricted set of robots like those on the Connectivity-Boundary it turns out to be a major challenge to find a symmetricity preserving protocol that works for all configurations.

5.2.3 Generalization of Averaging

To solve NEAR-GATHERING with the averaging protocol Algorithm 4 the initial configurations must contain evenly spread robots. One example is a square (or triangle or hexagon) that is filled with a regular grid with distance < viewing range. Also, perturbations of such configurations lead to NEAR-GATHERING. However, you can always create configurations that loose connectivity. Place twice $n/2 - 1$ robots in Diam-δ-Configuration with distances $3 - 3\delta$ in between. Place two robots between them, such that the UDG with unit distance $1 - \delta$ is connected. Even for the weight functions, the connectivity will be lost if n is large enough. Therefore, a general NEAR-GATHERING protocol cannot be based on averaging. The following problem remains open.

Open Problem 2 Let us consider disoriented robots according the $\mathcal{O}$BLOT model with limited visibility. Does a NEAR-GATHERING protocol that preserves symmetricity for connected configurations with non-convex Connectivity-Boundary (Definition 1.6) or 1-holes (Definition 1.7) exist?

The averaging protocol uses the global knowledge of the number n of robots in the swarm. In the ARBITRARY-PATTERN-FORMATION problem, this knowledge is implicitly available with the pattern P because it is defined with $|P| = n$. To adapt ε-GO-TO-THE-AVERAGE without any global knowledge, the parameter n in Equation (3.9) can be replaced by $\sum_{j=1}^{n} b(\|z_i - z_j\|^2)$. This does not significantly change the NEAR-GATHERING properties of the protocol; however, the changed evolution function must be proven to be invertible. To adapt the invertibility proof of Lemma 3.6 a new estimation of ε is required that is dependent on the Gersch Gorin radii of the evolution functions Jacobian $DF(\mathbf{z})$ (Lemma 3.8). An attempt to estimate it yields a result that still depends on n. However, the estimation was not sharp. Therefore, different analysis methods may show, that ε can indeed be estimated independent of n. The following problem remains open.

Open Problem 3 Consider disoriented robots according to the $\mathcal{O}$BLOT model with limited visibility. Does a non-trivial protocol exist that used no global knowledge and preserves symmetricity for all configurations?

5.3 Synchronicity

We assume the fully-synchronous $\mathcal{F}$SYNC scheduler for our protocol. In the related work, many papers assume $\mathcal{A}$SYNC. It turns out that APF is significantly harder under $\mathcal{A}$SYNC than under $\mathcal{F}$SYNC. The authors of [16] proved that, for an unlimited viewing range, APF under $\mathcal{A}$SYNC is equivalent to LEADER-ELECTION. This means the limitation from Theorem 1 does not hold for $\mathcal{A}$SYNC, instead the symmetricity of a configuration must be 1 to solve APF. The problem is not only a harder one in therms of solution space, apparently the solutions to the problem are also harder to prove correct. [34] provided a solution for the same problem and stated, that APF is equivalent under $\mathcal{A}$SYNC and $\mathcal{F}$SYNC. However, [16] disproved their result.

An unlimited viewing range makes it much easier to maintain common knowledge in the swarm (like a common coordinate system). For a limited viewing range, our protocol must maintain this information during execution (e.g. using the ε-granular drawing formations). In the $\mathcal{A}$SYNC model, where only a part of the drawing formation might be activated, we would have to ensure that "partially" moving a drawing formation does not destroy the encoded information (e.g., by encoding information redundantly). It is a crucial part of ε-granular drawing formations that their robots can identify them and we would have to maintain this property under partial movements. While it seems challenging (especially considering the failed attempt from [34]), a careful design might be able to solve this.

Open Problem 4 Let us consider disoriented robots according the $\mathcal{O}$BLOT model with limited visibility and $\mathcal{A}$SYNC scheduler. Does a protocol exist that solves APF for any non-trivial combination of initial configuration and pattern?

Furthermore, NEAR-GATHERING must also work in asynchronicity. The λ-contracting NEAR-GATHERING protocols work under $\mathcal{S}$SYNC (see Observations 2.3 and 2.4), but its collision avoidance does not work under $\mathcal{A}$SYNC. The collision avoidance requires that robots can compute all possible target positions of the robots around. When in $\mathcal{A}$SYNC a robot r is observed, which is currently moving, the snapshot that r used to compute its movement cannot be observed. Therefore, its target position cannot be computed, and the collision avoidance method does not work. A different approach to collision avoidance that would work well under $\mathcal{A}$SYNC is to define for each robot an exclusive area where it is allowed to move, but no other robot may enter. This can be done with a function that is only dependent on the current positions of robots, not their target positions (e.g. Voronoi cells). A major problem with this approach is that the movement of robots near the global convex hull may be infinitesimal small, if they are blocked by nearby robots in the inside of the swarm. The system must be carefully designed so that no deadlock is created. The protocol is not λ-contracting because a robot that is at the edge of its local convex hull may observe other robots in a constant distance but nevertheless moves only infinitesimal far. Therefore, the running-time and correctness proof provided in Chapter 2 would not apply for this approach.

Open Problem 5 Does a NEAR-GATHERING protocol for disoriented robots according the $\mathcal{O}$BLOT model with limited visibility and $\mathcal{A}$SYNC scheduler exist?

The method of analyzing the symmetricity change induced by a protocol presented in Chapter 3 requires that the evolution function of a protocol be locally invertible (Theorem 5). The partial activation of robots in $\mathcal{S}$SYNC can be included in the evolution function by adding the binary parameters $a_1, \cdots, a_n$, where a_i determines whether $z_i^+ = z_i$ or $z_i^+ = f(z_i, \mathbf{z})$. However, because of the arbitrary nature of the activation, that makes the evolution function non-invertible. Therefore, our method cannot be adapted for $\mathcal{S}$SYNC and $\mathcal{A}$SYNC. The following problem remains open.

Open Problem 6 What properties must a protocol have to preserve symmetricity under the $\mathcal{S}$SYNC or $\mathcal{A}$SYNC scheduler?

Bibliography

[1] Muhammad Zaki Almuzakki and Bayu Jayawardhana. Cooperative nearest-neighbor control of multi-agent systems: Consensus and formation control problems. *IEEE Control Systems Letters*, 7:1873–1878, 2023.

[2] Hideki Ando, Yoshinobu Oasa, Ichiro Suzuki, and Masafumi Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, 1999.

[3] Subhash Bhagat, Sruti Gan Chaudhuri, and Krishnendu Mukhopadhyaya. Fault-tolerant gathering of asynchronous oblivious mobile robots under one-axis agreement. *Journal of Discrete Algorithms*, 36:50–62, 2016.

[4] Kaustav Bose, Ranendu Adhikary, Manash Kumar Kundu, and Buddhadeb Sau. Arbitrary pattern formation on infinite grid by asynchronous oblivious robots. *Theoretical Computer Science*, 815:213–227, 2020.

[5] Kaustav Bose, Manash Kumar Kundu, Ranendu Adhikary, and Buddhadeb Sau. Arbitrary pattern formation by asynchronous opaque robots with lights. *Theoretical Computer Science*, 849:138–158, 2021.

[6] Michael Braun, Jannik Castenow, and Friedhelm Meyer auf der Heide. Local gathering of mobile robots in three dimensions. In *Proceedings of International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 12156 of *Lecture Notes in Computer Science*, pages 63–79. Springer, 2020.

[7] Jannik Castenow. *Local protocols for contracting and expanding robot formation problems*. PhD thesis, Paderborn University, Germany, 2023.

[8] Jannik Castenow, Matthias Fischer, Jonas Harbig, Daniel Jung, and Friedhelm Meyer auf der Heide. Gathering anonymous, oblivious robots on a grid. *Theoretical Computer Science*, 815:289–309, 2020.

[9] Jannik Castenow, Thorsten Götte, Till Knollmann, and Friedhelm Meyer auf der Heide. The max-line-formation problem - and new insights for gathering and chain-formation. In *Proceedings of International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 13046 of *Lecture Notes in Computer Science*, pages 289–304. Springer, 2021.

[10] Jannik Castenow, Jonas Harbig, Daniel Jung, Peter Kling, Till Knollmann, and Friedhelm Meyer auf der Heide. A unifying approach to efficient (near)-gathering of disoriented robots with limited visibility. In *Proceedings of International Conference on Principles of Distributed Systems (OPODIS)*, volume 253 of *LIPIcs*, pages 15:1–15:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[11] Jannik Castenow, Jonas Harbig, Daniel Jung, Till Knollmann, and Friedhelm Meyer auf der Heide. Gathering a euclidean closed chain of robots in linear time. In *Proceedings of International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)*, volume 12961 of *Lecture Notes in Computer Science*, pages 29–44. Springer, 2021.

[12] Jannik Castenow, Jonas Harbig, Daniel Jung, Till Knollmann, and Friedhelm Meyer auf der Heide. Gathering a euclidean closed chain of robots in linear time and improved algorithms for chain-formation. *Theoretical Computer Science*, 939:261–291, 2023.

[13] Jannik Castenow, Peter Kling, Till Knollmann, and Friedhelm Meyer auf der Heide. A discrete and continuous study of the max-chain-formation problem. *Information and Computation*, 285(Part):104877, 2022.

[14] Yao Chen, Jinhu Lu, Xinghuo Yu, and David J. Hill. Multi-agent systems with dynamical topologies: Consensus and applications. *IEEE Circuits and Systems Magazine*, 13(3):21–34, 2013.

[15] Pascal Chossat and Reiner Lauterbach. *Methods in Equivariant Bifurcations and Dynamical Systems*, volume 15 of *Advanced series in nonlinear dynamics*. World Scientific, 2000.

[16] Serafino Cicerone, Gabriele Di Stefano, and Alfredo Navarra. Asynchronous arbitrary pattern formation: the effects of a rigorous approach. *Distributed Computing*, 32(2):91–132, 2019.

[17] Mark Cieliebak, Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Distributed computing by mobile robots: Gathering. *SIAM Journal on Computing (SICOMP)*, 41(4):829–879, 2012.

[18] Reuven Cohen and David Peleg. Convergence properties of the gravitational algorithm in asynchronous robot systems. *SIAM Journal on Computing (SICOMP)*, 34(6):1516–1528, 2005.

[19] Andreas Cord-Landwehr, Bastian Degener, Matthias Fischer, Martina Hüllmann, Barbara Kempkes, Alexander Klaas, Peter Kling, Sven Kurras, Marcus Märtens, Friedhelm Meyer auf der Heide, Christoph Raupach, Kamil Swierkot, Daniel Warner, Christoph Weddemann, and Daniel Wonisch. A new approach for analyzing convergence algorithms for mobile robots. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, volume 6756 of *Lecture Notes in Computer Science*, pages 650–661. Springer, 2011.

[20] Andreas Cord-Landwehr, Matthias Fischer, Daniel Jung, and Friedhelm Meyer auf der Heide. Asymptotically optimal gathering on a grid. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 301–312. ACM, 2016.

[21] Shantanu Das, Paola Flocchini, Nicola Santoro, and Masafumi Yamashita. Forming sequences of geometric patterns with oblivious mobile robots. *Distributed Computing*, 28(2):131–145, 2015.

[22] Xavier Défago, Maria Potop-Butucaru, and Sébastien Tixeuil. Fault-tolerant mobile robots. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*, pages 234–251. Springer, 2019.

[23] Bastian Degener, Barbara Kempkes, Peter Kling, and Friedhelm Meyer auf der Heide. Linear and competitive strategies for continuous robot formation problems. *ACM Transactions on Parallel Computing (TOPC)*, 2(1):2:1–2:18, 2015.

[24] Bastian Degener, Barbara Kempkes, Tobias Langner, Friedhelm Meyer auf der Heide, Peter Pietrzyk, and Roger Wattenhofer. A tight runtime bound for synchronous gathering of autonomous robots with limited visibility. In *Proceedings of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 139–148. ACM, 2011.

[25] Yoann Dieudonné, Franck Petit, and Vincent Villain. Leader election problem versus pattern formation problem. In *Proceedings of International Symposium on Distributed Computing (DISC)*, volume 6343 of *Lecture Notes in Computer Science*, pages 267–281. Springer, 2010.

[26] Florian Dörfler and Francesco Bullo. Synchronization in complex networks of phase oscillators: A survey. *Automatica*, 50(6):1539–1564, 2014.

[27] Miroslaw Dynia, Jaroslaw Kutylowski, Pawel Lorek, and Friedhelm Meyer auf der Heide. Maintaining communication between an explorer and a base station. In *Proceedings of International Conference on Biologically Inspired Collaborative Computing (IFIP)*, volume 216, pages 137–146. Springer, 2006.

[28] Paola Flocchini, Alfredo Navarra, Debasish Pattanayak, Francesco Piselli, and Nicola Santoro. Oblivious robots under sequential schedulers: universal pattern formation. In *Proceedings of International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 15671 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2025.

[29] Paola Flocchini, Debasish Pattanayak, Francesco Piselli, Nicola Santoro, and Yukiko Yamauchi. Asynchronous separation of unconscious colored robots. In *Proceedings of International Symposium on Computing and Networking Workshops (CANDARW)*, pages 183–189. IEEE, 2024.

[30] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro, editors. *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*. Springer, 2019.

[31] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. Moving and computing models: Robots. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2019.

[32] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Distributed computing by mobile robots: uniform circle formation. *Distributed Computing*, 30(6):413–457, 2017.

[33] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theoretical Computer Science*, 407(1-3):412–447, 2008.

[34] Nao Fujinaga, Yukiko Yamauchi, Hirotaka Ono, Shuji Kijima, and Masafumi Yamashita. Pattern formation by oblivious asynchronous mobile robots. *SIAM Journal on Computing (SICOMP)*, 44(3):740–785, 2015.

[35] Raphael Gerlach, Sören von der Gracht, and Michael Dellnitz. On the dynamical hierarchy in gathering protocols with circulant topologies. In *Proceedings of International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 15671 of *Lecture Notes in Computer Science*, pages 315–332. Springer, 2025.

[36] Raphael Gerlach, Sören von der Gracht, Christopher Hahn, Jonas Harbig, and Peter Kling. Symmetry preservation in swarms of oblivious robots with limited visibility. In *Proceedings of International Conference on Principles of Distributed Systems (OPODIS)*, volume 324 of *LIPIcs*, pages 13:1–13:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[37] Semyon Aronovich Geršgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izvestija Akademii Nauk SSSR, Serija Matematika*, 7(6):749–754, 1931.

[38] Martin Golubitsky, Ian Stewart, and David G. Schaeffer. *Singularities and Groups in Bifurcation Theory*, volume 69 of *Applied mathematical sciences*. Springer, 1988.

[39] Christopher Hahn, Jonas Harbig, and Peter Kling. Forming large patterns with local robots in the OBLOT model. In *Proceedings of Symposium on Algorithmic Foundations of Dynamic Networks (SAND)*, volume 292 of *LIPIcs*, pages 14:1–14:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[40] Parichehr Hassanzadeh, Fatemeh Atyabi, and Rassoul Dinarvand. Creation of Nanorobots: Both State-of-the-Science and State-of-the-Art. *Biomedical Reviews*, 27(0):19, 2017.

[41] Morris W. Hirsch and Stephen Smale. *Differential equations, dynamical systems, and linear algebra*, volume 60 of *Pure and applied mathematics*. Acad. Press, [nachdr.] edition, 1988. Smale, Stephen (VerfasserIn).

[42] M. B. Ignatyev. Necessary and sufficient conditions of nanorobot synthesis. *Doklady Mathematics*, 82(1):671–675, 2010.

[43] Heinrich Jung. Ueber die kleinste kugel, die eine räumliche figur einschliesst. *Journal für die reine und angewandte Mathematik*, 123:241–257, 1901.

[44] Heinrich Jung. Über den kleinsten kreis, der eine ebene figur einschließt. *Journal für die reine und angewandte Mathematik*, 137:310–313, 1910.

[45] Chang-kwon Kang, Farbod Fahimi, Rob Griffin, D Brian Landrum, Bryan Mesmer, Guangsheng Zhang, Taeyoung Lee, Hikaru Aono, Jeremy Pohly, Jesse McCain, et al. Marsbee-swarm of flapping wing flyers for enhanced mars exploration. Technical report, NASA, 2019.

[46] Branislav Katreniak. Convergence with limited visibility by asynchronous mobile robots. In *Proceedings of International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 6796 of *Lecture Notes in Computer Science*, pages 125–137. Springer, 2011.

[47] David G. Kirkpatrick, Irina Kostitsyna, Alfredo Navarra, Giuseppe Prencipe, and Nicola Santoro. Separating bounded and unbounded asynchrony for autonomous robots: Point convergence with limited visibility. In *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC)*, pages 9–19. ACM, 2021.

[48] Sergey Kornienko, Olga Kornienko, and Paul Levi. Minimalistic approach towards communication and perception in microrobotic swarms. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2228–2234. IEEE, 2005.

[49] Geunho Lee and Nak Young Chong. A geometric approach to deploying robot swarms. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):257–280, 2008.

[50] Geunho Lee, Yasuhiro Nishimura, Kazutaka Tatara, and Nak Young Chong. Three dimensional deployment of robot swarms. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5073–5078. IEEE, 2010.

[51] Shouwei Li, Friedhelm Meyer auf der Heide, and Pavel Podlipyan. The impact of the gabriel subgraph of the visibility graph on the gathering of mobile autonomous robots. *Theoretical Computer Science*, 852:29–40, 2021.

[52] Shouwei Li, Christine Markarian, Friedhelm Meyer auf der Heide, and Pavel Podlipyan. A continuous strategy for collisionless gathering. *Theoretical Computer Science*, 852:41–60, 2021.

[53] Ji Lin, A. Stephen Morse, and Brian D. O. Anderson. The multi-agent rendezvous problem. part 1: The synchronous case. *SIAM Journal on Control and Optimization (SICON)*, 46(6):2096–2119, 2007.

[54] Ji Lin, A. Stephen Morse, and Brian D. O. Anderson. The multi-agent rendezvous problem. part 2: The asynchronous case. *SIAM Journal on Control and Optimization (SICON)*, 46(6):2120–2147, 2007.

[55] Giuseppe Antonio Di Luna, Paola Flocchini, Nicola Santoro, and Giovanni Viglietta. Turingmobile: a turing machine of oblivious mobile robots with limited visibility and its applications. *Distributed Computing*, 35(2):105–122, 2022.

[56] Moumita Mondal and Sruti Gan Chaudhuri. Uniform circle formation by swarm robots under limited visibility. In *Proceedings of International Conference on Distributed Computing and Internet Technology (ICDCIT)*, volume 11969 of *Lecture Notes in Computer Science*, pages 420–428. Springer, 2020.

[57] Linda Pagli, Giuseppe Prencipe, and Giovanni Viglietta. Getting close without touching: near-gathering for autonomous mobile robots. *Distributed Computing*, 28(5):333–349, 2015.

[58] Arkadij Pikovskij, Michael Rosenblum, and Jürgen Kurths. *Synchronization*, volume 12 of *Cambridge nonlinear science series*. Cambridge Univ. Press, 1st paperback ed., repr edition, 2003.

[59] Pavan Poudel and Gokarna Sharma. Time-optimal gathering under limited visibility with one-axis agreement. *Information*, 12(11):448, 2021.

[60] Giuseppe Prencipe. Impossibility of gathering by a set of autonomous mobile robots. *Theoretical Computer Science*, 384(2-3):222–231, 2007.

[61] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3293–3298. IEEE, 2012.

[62] Hirokazu Seike and Yukiko Yamauchi. Separation of unconscious colored robots. In *Proceedings of International Symposium on Stabilizing, Safety, and Security of Distributed Systems (SSS)*, volume 14310 of *Lecture Notes in Computer Science*, pages 328–343. Springer, 2023.

[63] Fernando Soto, Jie Wang, Rajib Ahmed, and Utkan Demirci. Medical Micro/-Nanorobots in Precision Medicine. *Advanced Science*, 7(21), 2020.

[64] Ichiro Suzuki and Masafumi Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM Journal on Computing (SICOMP)*, 28(4):1347–1363, 1999.

[65] John Ronald Reuel Tolkien. *The return of the king : being the third part of The Lord of the rings*. George Allen & Unwin, 1955.

[66] U-Blox. MIA-M10 series - U-blox M10 standard precision GNSS SiP modules. https://www.u-blox.com/en/product/mia-m10-series. Accessed: 2025-06-08.

[67] U-Blox. UBX-R52 series - Multi-band LTE-M / NB-IoT / satellite chipset. https://www.u-blox.com/en/product/ubx-r5-series. Accessed: 2025-06-08.

[68] Giovanni Viglietta. Uniform circle formation. In *Distributed Computing by Mobile Entities, Current Research in Moving and Computing*, volume 11340 of *Lecture Notes in Computer Science*, pages 83–108. Springer, 2019.

[69] Masafumi Yamashita and Ichiro Suzuki. Characterizing geometric patterns formable by oblivious anonymous mobile robots. *Theoretical Computer Science*, 411(26-28):2433–2453, 2010.

[70] Yukiko Yamauchi. A survey on pattern formation of autonomous mobile robots: asynchrony, obliviousness and visibility. *Journal of Physics: Conference Series*, 473:012016, 2013.

[71] Yukiko Yamauchi, Taichi Uehara, and Masafumi Yamashita. Brief announcement: Pattern formation problem for synchronous mobile robots in the three dimensional euclidean space. In *Proceedings of ACM Symposium on Principles of Distributed Computing (PODC)*, pages 447–449. ACM, 2016.

[72] Yukiko Yamauchi and Masafumi Yamashita. Pattern formation by mobile robots with limited visibility. In *Proceedings of International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume 8179 of *Lecture Notes in Computer Science*, pages 201–212. Springer, 2013.