Bachelor Thesis

Local methods for swarm navigation and formation

submitted by

Niklaus Houska

June 30, 2019

supervised by

Prof. Dr. Stelian Coros
Computational Robotics Lab, ETH Zürich

Institute for Pervasive Computing
Department of Computer Science
ETH Zürich

Abstract

This paper presents a local method for organizing a swarm of robots into a column formation. The formation spots are defined by the integer coordinates of a two-dimensional formation space known by each swarm member. The members are distributed over these formation places only through local interaction and navigate using optimal reciprocal collision avoidance. The method was successfully tested in simulation with robot numbers between 4 and 200 and evaluated in its convergence time compared to an optimal global solution, with the local method on average 50% slower. A correct end configuration is reached independent of initial configuration and number of robots.

Table of contents

Nomenclature

$\mathbf{o}$	Formation origin
d	Formation spacing
n_c	Number of columns
f_1, f_2	Basis vectors of formation space
V_s	Vision radius short
V_l	Vision radius large
M	Movement strategy
S	Robot status
B_{left}, B_{right},B_{top}	Free space indicators
$\mathbf{p}_f = (i, j)$	Vector in formation space
$\mathbf{p}_f^{clos}$	Closest formation place
$\mathbf{p}_f^{des}$	Desired formation place
$\mathbf{v}_{pref}$, $\mathbf{v}_{cf}$	Preferred and collision-free velocity
d_F, d_U, d_B, d_M	Delays

1 Introduction

Swarm robotic systems are inspired by nature where social animals such as ants, bees or birds collectively accomplish impressive goals. Erol Şahin introduced in the year 2005 following definition:

> Swarm robotics is the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among agents and between the agents and the environment [1].

Applications lie in following task domains: Tasks that cover a large area, tasks that are too dangerous, tasks that scale-up or scale-down in time and tasks that require redundancy. For instance, environmental monitoring, clearing mine fields, containing chemical leakage or creating dynamic communication networks on a battlefield are all suitable tasks [1]. Because of the large number of individuals in a swarm robotic system, collective movement emerges as an important study subject. Flocking algorithms in particular suggest methods that result in swarm motion that resembles that of a flock of birds or a fish school. This thesis on the other hand is concerned with how a swarm of robots can autonomously assemble into a formation, laying the foundation for a more coordinated and structured swarm movement. The aim of the project is to design a local method that efficiently distributes a swarm of robots over predefined formation places. Efficiency is hereby measured by comparing the local method to an optimal global method that is using a centralized control.

Some interesting local methods have been proposed for formation control. The *physicomimetrics* framework [2] has driven a multi-robot system into a desired configuration using virtual physics forces. Summarized it treated the system as a molecular dynamics simulation where each robot corresponded to a physical particle. Results have shown swarm behavior similar to those of solids, liquids and gases and hexagonal or squared pattern formation were achieved. But the outcome depended strongly on the initial configuration. [3] introduced formation control by utilizing elliptical surfaces and limiting functions. The surface on which robots travel has been constructed from artificial potential fields which defined the overall swarm geometry and the individual member spacing. These artificial potential functions together with the limiting functions controlled the formation, orientation and movement of the swarm. The methods offered high flexibility and worked with various formations such as circle, ellipse and wedge. Movement has been realized by a central control constantly sending updated parameters for location and orientation of the formation.

2 Methods

The objective of this thesis is to organize a swarm of homogeneous and autonomous robots into a column formation. Given only the formation definition, each robot finds its place through local communication.

2.1 Formation Definition

The column formation used for the present method is characterized by the number of columns and the member spacing. Formally, it is defined by the tuple $(\mathbf{o}, \mathbf{b}_1, d, n_c)$. $\mathbf{o}$ is the origin giving the location of the first row, $\mathbf{b}_1$ a unit vector in column direction, d the distance between the formation places, and n_c the number of columns.

The row direction $\mathbf{b}_2$ is obtained by rotating $\mathbf{b}_1$ by 270 degrees. These vectors are used to define a new basis $F = \{\mathbf{f}_1 := d\mathbf{b}_1, \mathbf{f}_2 := d\mathbf{b}_2\} \in \mathbb{R}^2$, where all $\mathbf{p}_f \in F$ with integer coordinates describe a potential formation place. For instance, $(2, 0) \in F$ defines the left most formation place of the third row. Let $\mathbf{t} := \mathbf{o} - \frac{n_c-1}{2}\mathbf{f}_2$ be the offset from the origin, such that the origin stays at the center of the columns, and $\boldsymbol{T} := \begin{bmatrix}\mathbf{f}_1 & \mathbf{f}_2\end{bmatrix}$ the transformation matrix. Then the change of basis is calculated as follows:

$$\begin{aligned} \mathbf{p}_w &= \mathbf{p}_f \boldsymbol{T} + \mathbf{t} \\ \mathbf{p}_f &= (\mathbf{p}_w - \mathbf{t})\boldsymbol{T}^{-1} \end{aligned} \tag{2.1}$$

The set of valid formation places is given by all $\mathbf{p}_f = (i, j)$ where $i, j \in \mathbb{N}_0$ and $j \leq n_c - 1$. Furthermore, for any two vectors $\mathbf{p}_f = (i_1, j_1)$ and $\mathbf{q}_f = (i_2, j_2)$; if $i_1 < i_2$ then $\mathbf{p}_f$ is above $\mathbf{q}_f$, respectively if $j_1 < j_2$ then $\mathbf{p}_f$ is left of $\mathbf{q}_f$ – in terms of the formation location.

The choice to treat the formation as a simple change of basis enables the method to efficiently query and communicate information. Moreover, it decouples the robot's location inside the formation with their actual world location. When each robot reached their desired formation place $\mathbf{p}_f^{des}$, an update of either $\mathbf{o}$, $\mathbf{b}_1$ or d will potentially move the whole swarm, but preserve the established structure, as $\mathbf{p}_f^{des}$ will simply translate to a different world location.

2.2 Optimal Solution

The optimal solution acts as baseline to the proposed local method. It assigns a formation place to each robot by calculating a bijection between the n robot locations P and the n first formation places G. The bijection minimizes the cost function that is given by the sum of squared distances

$$f_{\min} = \arg\min_f \sum_{i=1}^{n} ||\mathbf{g}_{f(i)} - \mathbf{p}_i||^2 \tag{2.2}$$

where f is the assignment function, $\mathbf{p}_i$ the location of the robot and $\mathbf{g}_{f(i)}$ its assigned formation place in world coordinates.

The linear assignment problem is a well-known optimization problem, which can be solved by finding a minimum-weight perfect matching through linear programming [4]. Each edge $(\mathbf{p}, \mathbf{g})$ where $\mathbf{p} \in P$ and $\mathbf{g} \in G$ has a weight of $w_{pg} = ||\mathbf{g} - \mathbf{p}||^2$. For each edge exists an indicator variable x_{pg} that is 1 if the edge is contained in the matching and 0 otherwise. This, together with the constraints that ensure an actual perfect matching, leads to following linear program:

$$\begin{aligned} &\textbf{minimize} \quad \sum_{(\mathbf{p},\mathbf{g}) \in P \times G} w_{pg} x_{pg} \\ &\textbf{subject to} \quad \sum_{\mathbf{g} \in G} x_{pg} = 1 \text{ for } \mathbf{p} \in P, \quad \sum_{\mathbf{p} \in P} x_{pg} = 1 \text{ for } \mathbf{g} \in G \\ &0 \leq x_{pg} \leq 1 \text{ for } \mathbf{p}, \mathbf{g} \in P, G \end{aligned} \tag{2.3}$$

The original formulation describes an integer linear program. However, the integrality constraint on the indicator variables can be dropped and the linear program solved using standard methods for continuous linear programs. Then although this formulation allows for fractional variable values, the linear program always has an optimal solution where all variables take integer values. For the present simulations, this linear program was used, thus an optimal assignment was found.

2.3 Local Method

A real-time control defines the behavior of every robot to achieve convergence. At each iteration, the robots exchange information with their neighbors and update their desired formation place $\mathbf{p}_f^{des}$, then choose their preferred velocity towards that goal and modify it to avoid collisions. All variables in the parts below refer to a single iteration of the control methods and belong to a single robot.

2.3.1 Desired Formation Place

This section presents the main part of the control – updating the desired formation place. First, all the concepts used in the method are presented and their relevance to the success argued. Then, the parts are put together to form the complete method.

If the swarm consists of n robots, convergence means that the robots are distributed over the first $n_r := \lceil \frac{n}{n_c} \rceil$ rows of the formation, whereas the first $n_r - 1$ rows are completely filled and the last row possibly only partially. Because an individual robot does not know about the size n of the swarm and subsequently not about n_r, it will prefer formation places higher up, i.e. with lower row index i.

Vision Radius

A larger vision field and thus higher knowledge about a robots surroundings can lead to a more accurate decision making. But more vision also leads to an increased number of possibilities and therefore requires an equally complex decision process – or the introduction of advanced techniques such as machine learning or reinforcement learning, which are not part of this thesis. The goal is therefore to find the minimal vision range that provides enough information to make precise decisions but keeps them simple and general.

A robot has a smaller V_s and a larger V_l vision radius. They are restricted as follows

$$\begin{aligned} V_s &: \quad d < V_s < 1.5 \cdot d \\ V_l &: \quad V_s \leq V_l < 2 \cdot d \end{aligned} \tag{2.4}$$

With V_s a robot sees the next formation place but never more than the neighboring spots of its closest formation place $\mathbf{p}_f^{clos}$. V_l is slightly larger but also prevents a robot of seeing more than the neighboring spots at its location. Note that this formulation restricts the distance d between two formation places by $V_l \leq \mathcal{V}$ where $\mathcal{V}$ is the robots maximal vision range. Both V_s and V_l are visualized in Figure 2.1.

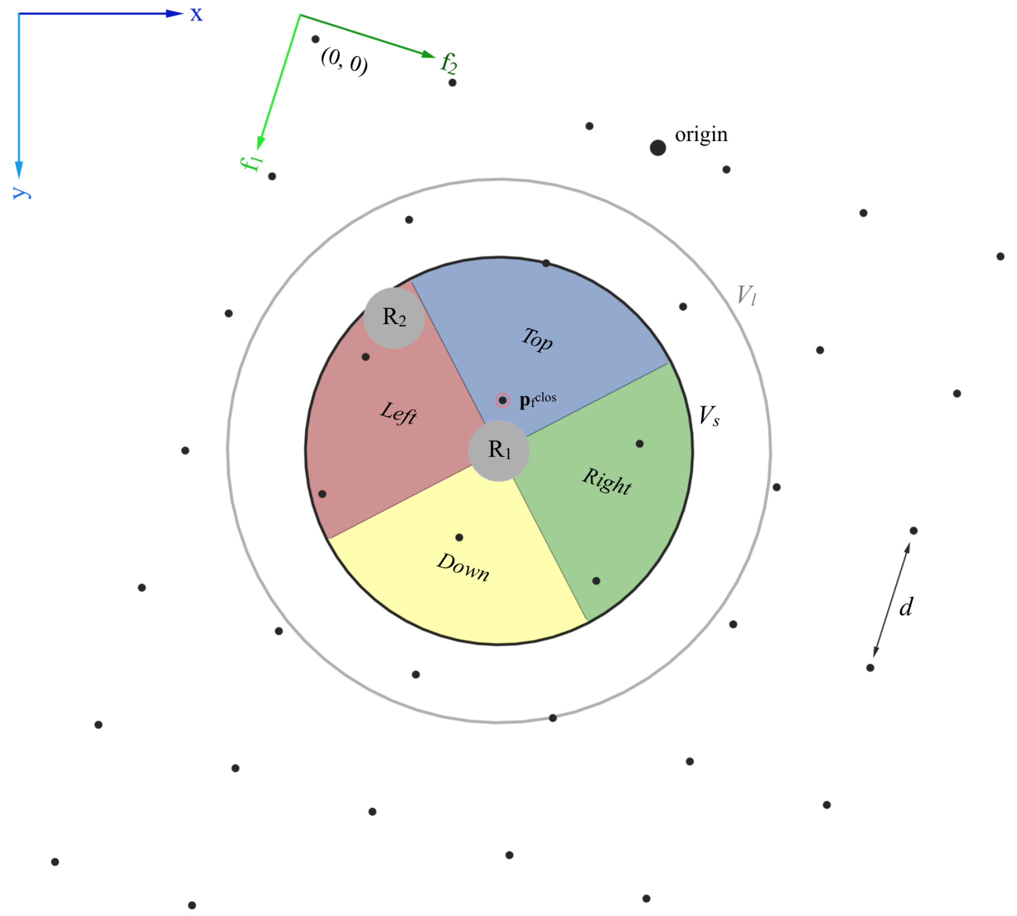

Figure 2.1: Illustration of a formation with $n_c = 6$ and two robots R_1 and R_2. Displayed are V_s, V_l, $\mathbf{p}_f^{clos}$ and the four quartiles of R_1. R_2 is located in the $Left$ quartile of R_1.

Dominance

If any two robots R_1 and R_2 desire the same formation place, a mechanism is needed to resolve this conflict: dominance. $R_1 \prec R_2$ denotes that R_1 is dominated by R_2. The set of dominating robots of R_1 is determined by the column j^{clos} of its closest formation place

$\mathbf{p}_f^{clos} = (i^{clos}, j^{clos})$ with $i^{clos}, j^{clos} \in \mathbb{N}_0$.

$$\begin{aligned}
R_1 \prec_l R_2 \quad &: \quad j^{clos} < \lfloor \frac{n_c - 1}{2} \rfloor \quad \wedge \quad R_2 \text{ in } \textit{Left} \text{ or } \textit{Top} \text{ quartile of } R_1 \\
R_1 \prec_r R_2 \quad &: \quad j^{clos} > \lfloor \frac{n_c - 1}{2} \rfloor \quad \wedge \quad R_2 \text{ in } \textit{Right} \text{ or } \textit{Top} \text{ quartile of } R_1 \\
R_1 \prec_m R_2 \quad &: \quad j^{clos} = \lfloor \frac{n_c - 1}{2} \rfloor \quad \wedge \quad R_2 \text{ in } \textit{Left}, \textit{Right} \text{ or } \textit{Top} \text{ quartile of } R_1
\end{aligned} \tag{2.5}$$

An illustration of the quartiles is given in Figure 2.1.

A robot gives up its desired formation place if a dominating robot inside its V_l desires the same place. This implies that robots can always claim formation spots towards the middle or downwards. Consequently, it implies that robots require cooperation of their dominating neighbors to advance upwards or towards the edge of the formation.

Neighbor Links

A robot forms links with the robots on neighboring formation places. Each robot occupying an non-border formation place can have a *Top*-, $Left$-, *Down*- and *Right*-link. Links are mutual; a robot is the *Right*-link of its $Left$-link and the *Down*-link of its *Top*-link, and vice versa. Meaning that if R_1 tries to set R_2 as a link, but R_2 has a different correspondent symmetric link, R_1 fails. The following algorithm determines if robot R_1 puts R_2 as its l-link, where $l \in \{Top, Left, Down, Right\}$.

Algorithm 1 Try l-link with R_2

```
  Let $o$ be the symmetric link to $l$
**Require:** $R_1$ cleared any link with $R_2$ that is not $l$
**Require:** Equation 2.6
  $R_l \leftarrow$ current $l$-link of $R_1$
  $R_o \leftarrow$ current $o$-link of $R_2$
  **if** $R_l$ and $R_o$ do not exists **then**
    $l$-link is approved
  **else if** $R_o$ exists, $R_l$ does not **then**
    $l$-link is approved if $R_o = R_1$
  **else if** $R_l$ exists, $R_o$ does not **then**
    $l$-link is approved if $R_l = R_2$ **or** $R_2$ is closer to $R_1$ than $R_l$
  **else if** $R_l$ and $R_o$ exist **then**
    $l$-link is approved if $R_l = R_2$ **and** $R_o = R_1$
    $l$-link is approved if $R_l \neq R_2$ **and** $R_2$ is closer to $R_1$ than $R_l$
    $l$-link is cleared if $R_l = R_2$ **and** $R_o \neq R_1$
  **end if**
  **return** whether $l$-link is approved or not
```

Which type of link R_1 assigns R_2 to simply depends on the quartile R_2 is located in, i.e. left quartile leads to a $Left$-link. Robots on a border column consider neighbors on the outside as either *Down*- or *Top*-link depending on who is above who. To prevent cyclic link construction between three or more robots and to only consider actual neighbors, Equation 2.6 needs to hold if R_1 tries l-link with R_2

$$\begin{aligned} l \in \{Left, Right\} &\quad : \quad R_2 \text{ inside } V_s \quad \wedge \quad i_d^{des} = i_{nd}^{clos} \\ l \in \{Top\} &\quad : \quad R_2 \text{ inside } V_s \quad \wedge \quad j_d^{des} = j_{nd}^{clos} \\ l \in \{Down\} &\quad : \quad R_2 \text{ inside } V_s \end{aligned} \tag{2.6}$$

where i_d^{des} (j_d^{des}) is the row (column) of the desired formation place of the dominant robot and i_{nd}^{clos} (j_{nd}^{clos}) the row (column) of the closest formation place of the non-dominant robot. V_s is the small vision radius of R_1. Note that always either $R_1 \prec R_2$ or $R_2 \prec R_1$ holds.

Status

The status S of a robot is used to communicate information about space inside the formation. There are eight statuses:

0 : MOVING	not reached $\mathbf{p}_f^{des}$ yet
1 : FULL	full row
2 : C LEFT	row has free space to the left
3 : C RIGHT	row has free space to the right
4 : C BOTH	row has free space on both sides
5 : U LEFT	unsure if free space on the left
6 : U RIGHT	unsure if free space on the right
7 : U BOTH	unsure if free space on both sides

To collectively fill every row and reach convergence given the limited vision, it is necessary that robots share information about free formation places. A robot that is surrounded by robots with status MOVING cannot doubtlessly tell whether its row is full or has some free space. To prevent triggering false actions based on a speculative information, the U statuses were introduced that cover this case.

While trying to find the best formation place, a robot will naturally observe that a former free side becomes full, i.e. its status changes from a C status to FULL. The opposite case is therefore susceptible and generally implies that its former occupy moved away in favor of another robot that is still outside the vision radius. d_F defines the delay in changing status FULL to anything else except MOVING. Similarly, d_U defines the minimum time a U status must be held before it can change to a C status. Algorithm 2 presents the complete update routine.

Algorithm 2 Update status S of R

```
$S_l \leftarrow$ status of $Left$-link if existent
$S_r \leftarrow$ status of $Right$-link if existent
**if** $R$ reached $\mathbf{p}_f^{des} = (i, j)$ **then**
    $c_l \leftarrow j \neq 0$ **and** $S_l \in$ {C LEFT, C BOTH}
    $c_r \leftarrow j \neq n_c - 1$ **and** $S_r \in$ {C RIGHT, C BOTH}
    $u_l \leftarrow j \neq 0$ **and** $S_l \in$ {MOVING, U LEFT, U BOTH}
    $u_r \leftarrow j \neq n_c - 1$ **and** $S_r \in$ {MOVING, U RIGHT, U BOTH}
    **if** $c_l$ **or** $c_r$ **then**
        $S_{new} \leftarrow$ respective C status
    **else if** $u_l$ **or** $u_r$ **then**
        $S_{new} \leftarrow$ respective U status
    **else**
        $S_{new} \leftarrow$ FULL
    **end if**
**else**
    $S_{new} \leftarrow$ MOVING
**end if**
**if** neither $d_F$ nor $d_U$ are applicable **or** delay uninterruptedly run out **then**
    $S \leftarrow S_{new}$
**end if**
```

Indicators B_{left}, B_{right}, B_{top}

Status S holds verified information shared between robots in place. The binary indicators on the other hand simply tell a robot if there is potentially free space to the left or right in his current row. The indicators for robot R_1 at (i_1, j_1) with closest formation place $\mathbf{p}_{f,1}^{clos} = (i_1^{clos}, j_1^{clos})$ are defined as depicted in Equation 2.7

$$\begin{aligned} B_{left,1} &= j_1^{clos} \neq 0 \quad \wedge \quad \bigwedge_{R_k \in \mathcal{N}} B_{left,k} \wedge j_k < j_1 \\ B_{right,1} &= j_1^{clos} \neq n_c - 1 \quad \wedge \quad \bigwedge_{R_k \in \mathcal{N}} B_{right,k} \wedge j_k \geq j_1 \\ B_{top,1} &= i_1^{clos} \neq 0 \quad \wedge \quad (B_{top,Top\text{-link}} \vee R_1 \text{ has no robot inside its } Top \text{ quartile}) \end{aligned} \tag{2.7}$$

where $\mathcal{N}$ is the set of all robots R_k at (i_k, j_k) with $i_1^{clos} = i_k^{clos}$ that are inside $V_{s,1}$ plus those inside $V_{l,1}$ with status MOVING. Therefore, B_{left} and B_{right} are only false if there is a vision chain from the left most place to the right most. A short delay d_B requires a robot to wait a certain amount of time before changing either B_{left} or B_{right} from false back to true.

Movement Strategy

A robot updating its desired formation place has one possibility to go left or right. The order in which those options are checked matter if both are possible. The movement strategy $M \in$ {Left, Default, Right} determines which direction is preferred. Robots start with the default strategy that prefers movement towards the outside. Then the strategy is simply set to the last action taken. A delay d_M requires the robots to wait a certain amount of time before choosing a formation place against the strategy.

Improve

This section describes how a robot R at position $\mathbf{p}_f = (i, j)$ chooses an improved formation place $\mathbf{p}_f^{impr}$ to reach a higher row, or help another robot doing so.

As soon as the distance between $\mathbf{p}_f$ and $\mathbf{p}_f^{des}$ is lower than D_I, robot R will check three spots based on his closest formation place $\mathbf{p}_f^{clos} = (i^{clos}, j^{clos})$.

$$\begin{aligned}\mathbf{p}_f^{top} &= (i^{clos} - 1, j^{clos})\\ \mathbf{p}_f^{left} &= (i^{clos}, j^{clos} - 1)\\ \mathbf{p}_f^{right} &= (i^{clos}, j^{clos} + 1)\end{aligned} \tag{2.8}$$

If B_{top}, then $\mathbf{p}_f^{impr} = \mathbf{p}_f^{top}$, otherwise the left and right spots are checked according to the movement strategy M, the first allowed location is chosen. R moves horizontally if either B_{left} or B_{right} holds and it has a *Down*-Link with a C status. Assuming R is located in the left half, i.e. $j^{clos} < \lfloor \frac{n_c - 1}{2} \rfloor$, then it is dominated by its $Left$-Link and dominates its *Right*-Link. Therefore moving to the right is possible, whereas moving to the left requires cooperation. This leads to the following additional rule: R moves to the dominating side if there is a robot in its non-dominating quartile and B_{left} holds, respectively B_{right} (whichever is the dominating side).

Backup

If both $\mathbf{p}_f^{des}$ and $\mathbf{p}_f^{clos}$ are claimed by dominating robots, R needs to find a backup place. Similar to finding an improved formation place, R checks three formation spots:

$$\begin{aligned}\mathbf{p}_f^{left} &= (i^{clos}, j^{clos} - 1)\\ \mathbf{p}_f^{right} &= (i^{clos}, j^{clos} + 1)\\ \mathbf{p}_f^{down} &= (i^{clos} + 1, j^{clos})\end{aligned} \tag{2.9}$$

If B_{left} and B_{right} hold, the position $\mathbf{p}_f^{left}$ or $\mathbf{p}_f^{right}$ is chosen according to the movement strategy M. $\mathbf{p}_f^{down}$ is only chosen when both are false.

Complete Method

The complete method is depicted in Algorithm 3. It is repeatedly run with a frequency of f_{update}. Each robot starts with $\mathbf{p}_f^{des} = \mathbf{p}_f^{clos}$.

Algorithm 3 Update goal location

```
Require: Global formation definition $(\mathbf{o}, \mathbf{b}_1, d, n_c)$,
  - internal parameters: $V_l$, $V_s$, $D_I$, $d_F$, $d_U$, $d_M$, $d_B$
  - internal variables: $\mathbf{p}_w$, $\mathbf{p}_f^{des}$, $S$, $M$, neighbor links, $B_{left}$, $B_{right}$, $B_{top}$
  - external from each robot $R_k$: $\mathbf{p}_{w,k}$, $\mathbf{p}_{f,k}^{des}$, $S_k$, neighbor links, $B_{left,k}$, $B_{right,k}$, $B_{top,k}$
  Clear all linked robots that are located outside $V_s$
  $\mathcal{D} \leftarrow$ container for dominant robots
  for all $R_k$ inside $V_l$ do
    Determine potential link $l$ based on quartile
    Add $R_k$ to $\mathcal{D}$ if dominant according to Equation 2.5
    if Algorithm 1 is successful then
      Update $l$-link with $R_k$
    end if
  end for
  Update status $S$, i.e. run Algorithm 2
  Update $B_{left}$, $B_{right}$ and $B_{top}$ according to Equation 2.7
  $\mathbf{p}_f^{impr} \leftarrow$ desired improved formation place (Equation 2.8)
  if $\mathbf{p}_f^{impr}$ exists and not contested based on Algorithm 4 then
    Update $\mathbf{p}_f^{des}$ with $\mathbf{p}_f^{impr}$
  else if $\mathbf{p}_f^{des}$ not contested based on Algorithm 4 then
    Keep $\mathbf{p}_f^{des}$
  else if $\mathbf{p}_f^{clos}$ not contested based on Algorithm 4 then
    Update $\mathbf{p}_f^{des}$ with $\mathbf{p}_f^{clos}$
  else
    $\mathbf{p}_f^{back} \leftarrow$ backup formation place (Equation 2.9)
    Update $\mathbf{p}_f^{des}$ with $\mathbf{p}_f^{back}$
  end if
```

The routine depicted in Algorithm 4 makes use of the dominance mechanism to determine if input formation place $\mathbf{p}_f$ is contested and thus cannot be chosen. Moreover, the location is rejected if it crosses the desired formation place of a dominant neighbor link.

Algorithm 4 Check if $\mathbf{p}_f = (i, j)$ is contested

```
  for all $R_k$ in $\mathcal{D}$ do
    return true if $R_k$'s desired formation place is $\mathbf{p}_f$
  end for
  $(i_{top}, j_{top}) \leftarrow$ desired formation place of $Top$-link
  $(i_{left}, j_{left}) \leftarrow$ desired formation place of $Left$-link
  $(i_{right}, j_{right}) \leftarrow$ desired formation place of $Right$-link
  return true if $i_{top} \geq i$
  return true if $Left$-link $\in \mathcal{D}$ and $j_{left} > j$
  return true if $Right$-link $\in \mathcal{D}$ and $j_{right} < j$
```

2.3.2 Movement

A robots preferred velocity towards its desired formation place is simply given by a proportional controller presented in [5].

$$\mathbf{v}_{pref} = V \min\left(1, \frac{||\mathbf{p}_w^{des} - \mathbf{p}_w||}{K}\right) \frac{\mathbf{p}_w^{des} - \mathbf{p}_w}{||\mathbf{p}_w^{des} - \mathbf{p}_w||} \tag{2.10}$$

where $V > 0$ is the default speed of the robots and slightly lower than v_{max}. The constant $K > 0$ is the distance to the goal from which the velocity is reduced linearly. $\mathbf{p}_w^{des}$ and $\mathbf{p}_w$ are the desired formation location and robot location in world coordinates. Note that $\mathbf{v}_{pref}$ is not the actual velocity of the robot due to its non-holonomicity.

The controls $(\mathbf{v}_{NH}, \omega)$ given a holonomic velocity $\mathbf{v}_H$ are chosen such that they minimize their difference and achieve the correct orientation θ towards the movement direction in a fixed time T. If this is impossible due to the constraint on the angular velocity $\omega = \min\left(\frac{\theta}{T}, \omega_{max}\right)$, the robot turns in place by rotating at maximum speed – making $\mathbf{v}_{NH} = \mathbf{0}$. Otherwise, the linear velocity is given by

$$\mathbf{v}_{NH} = \mathbf{v}_H \frac{\theta \sin(\theta)}{2(1 - \cos(\theta))} \tag{2.11}$$

The controls are based on [6] that presents collision avoidance given the kinematic constraints of non-holonomic robots.

2.3.3 Local Collision Avoidance

Each robot R_i given its preferred velocity $\mathbf{v}_{pref}$ and the velocities of its neighbors computes a collision free velocity $\mathbf{v}_{cf}$ and applies the controls from Equation 2.11. The method implements local reciprocal collision avoidance that guarantees smooth trajectories and optimal velocities. The method is based on Velocity Obstacles [7] in velocity space and the fact that each neighbor runs the same procedure. The chosen method assumes holonomic robots [8] and is shortly summarized here. For the more precise method that includes the constraints of non-holonomic robots refer to NH-ORCA [6]. The simpler ORCA method was taken over NH-ORCA because it is less restrictive with $\mathbf{v}_{cf}$. The latter might lead to occasional collisions, which have no influence depending on the set up.

The Velocity Obstacle is calculated for a horizon τ, meaning that no collisions occur for the next τ seconds assuming all velocities stay the same. The set of collision-free velocities $ORCA_i^\tau$ for robot R_i is

$$ORCA_i^\tau = P \cap \bigcap_{j=1, j \neq i}^{n} ORCA_{i|j}^\tau \tag{2.12}$$

where $ORCA_{i|j}^\tau$ is the set of collision-free velocities for robot R_i at $\mathbf{p}_i$ with radius r_i and velocity $\mathbf{v}_i$ with respect to robot R_j at $\mathbf{p}_j$ with radius r_j and velocity $\mathbf{v}_j$. n being the number of robots in the neighborhood. It is a half plane derived from the general Velocity Obstacle. P is the circle of all possible holonomic velocities of R_i with radius v_{max}, the maximal linear speed. The key difference between ORCA and NH-ORCA is that P_{NH} would be the convex approximation of the set of all non-holonomic velocities that can be tracked within a fixed error.

The collision-free velocity for R_i is then selected as

$$\mathbf{v}_{cf} = \underset{\mathbf{v} \in ORCA_i^{\tau}}{\arg\min} ||\mathbf{v} - \mathbf{v}_{pref}|| \tag{2.13}$$

The collision avoidance method helps preventing the issue of stuck robots. A particular problem especially if the spacing d between two formation places is too narrow to allow a robot to pass in between. In the event of two robots being on collision course, both will contribute half to the avoidance by changing their preferred velocity, thus also robots that already reached their desired formation space will move and make way for a passing robot.

Table 2.1: Parameters and constants of the method, the virtual robot and ORCA

Symbol	Value	Unit	Description
V_s	$1.4 \cdot d$	cm	Short vision range
V_l	$1.9 \cdot d$	cm	Large vision range
D_I	$0.9 \cdot d$	cm	Minimum distance from $\mathbf{p}_w^{des}$ to choose $\mathbf{p}_f^{impr}$
d_F	2.8	s	Delay to change status FULL
d_U	1.2	s	Required time on U status before changing to C status
d_M	0.1	s	Delay to change movement strategy M
d_B	0.2	s	Delay to change B_{left} or B_{right} from False to True
f_{update}	0.1	s	Frequency of goal update (Algorithm 3)
$f_{sensing}$	0.1	s	Frequency of robot sensing
r	60	cm	Diameter of the robot
v_{max}	110	cm/s	Maximum linear speed
ω_{max}	3.96	rad/s	Maximum angular speed
τ	1	s	Time to collision horizon
T	0.2	s	Time to achieve orientation
V	100	cm/s	Default preferred speed
K	20	cm	Linear speed reduction distance

2.4 Simulation Setup

The method was implemented using the Unreal Engine 4.14 [9], exploiting the physics of the game engine to create an accurate simulation environment. Gurobi's linear solver was used to calculate the optimal goal assignment for the optimal solution [10]. The robot object was modeled in the 3D creation software Blender [11] inspired by the Footbots used in the Swarmanoid Project [12]. The engine handles collisions between robots during the simulation and provides a framework to specify sensing and movement for each robot. Local communication was implemented by giving each robot a sensing component that scans the environment at frequency $f_{sensing}$ and detects other robots in a specified vision range. The routine that updates $\mathbf{p}_f^{des}$ (Algorithm 3) – run at frequency f_{update} – then accesses the relevant information of all sensed robots. On the other hand, the robot's location and orientation are updated according to the controls $(\mathbf{v}_{cf}, \omega)$ every frame with a default frame rate of 120 frames per second. The selected parameters and constants used in the simulation are found in Table 2.1.

3 Results

The decentralized method and optimal solution were tested in simulation and compared to one another. Five test setups were constructed to evaluate the performance with different initial configurations on the same formation. The tests consisted of robots mainly joining the formation from the sides, from the top, from the bottom, robots distributed vertically on one side and an overcrowded scenario. Figure 3.1 shows the test setups. In another test series a varying number of robots were randomly placed to empirically assess the general performance.

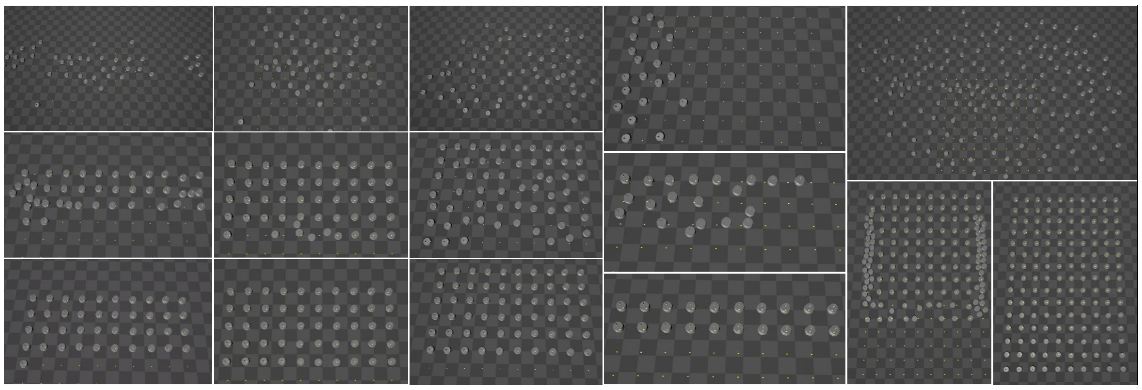

Figure 3.1: Test levels 1 to 5 (left to right): Images in top row show initial configuration. Images in middle row show progress of decentralized method at convergence time of optimal solution. Images in bottom row show end configuration. $n_c = 10$, $d = 140$ cm.

Each of the five test was first run with the optimal solution to get a reference value, then 30 times using the local method. The exact timings of the update functions vary from run to run and introduce noise that make each execution different, thus multiple runs were done to evaluate the robustness of the method. The other test setup randomly placed robots and measured the convergence time once for the optimal solution and once for the local method capturing the ratio between them. The number of robots per test varied such that at minimum 1 and at maximum 20 rows were filled. The robots were randomly placed within a circle centered at the middle of the second row with a radius between 5 and 20 meters.

The results for the five test levels and the random testing are presented in Table 3.1 respectively Table 3.2. All of the over 600 test runs were successful in terms of reaching an acceptable end configuration. There were no out-liners showing significantly worse performance.

The results in table Table 3.1 show an average convergence time that is below twice the optimal time. Only the last test level with 150 robots exceeds the ratio of 2. Its progress at convergence time of the optimal solution (15.02 seconds) is illustrated in the second image from the bottom right in Figure 3.1. The method fails to utilize the full width of the formation as the remaining moving robots are only traveling down the border columns.

Table 3.1: Results of the test levels. 30 runs, $n_c = 10$, $d = 140\text{cm}$

Optimal [s]	Avg (SD) [s]	Ratio	Description
8.08	12.71 (± 1.16)	1.57	41 robots joining mainly sideways
5.83	7.91 (± 1.11)	1.36	50 robots joining mainly from top
5.68	10.73 (± 1.48)	1.89	60 robots joining mainly from bottom
12.92	18.17 (± 1.34)	1.41	20 robots vertically distributed on the left
15.02	32.98 (± 1.20)	2.20	150 robots mainly joining from above

The standard deviation stays consistent at around 1.25 seconds for all test levels. From observation it is assumed that the largest contribution to the deviation happens at the end of the convergence, when the last row is filled; i.e. robots falsely moving down a row and eventually moving up again quickly add 2-3 seconds to the convergence time. The consistent deviation times support this observation.

Table 3.2: Results random testing. The numbers report the ratio between the time of optimal solution and local method. $d = 140\text{cm}$, first half: $n_c = 10$, second half: $n_c = 4$.

No. robots	# tests	10pth	25pth	Median	75pth	90pth	Avg
up to 40	66	1.07	1.14	1.29	1.51	1.73	1.35
up to 100	238	1.12	1.24	1.38	1.58	1.82	1.43
up to 200	304	1.14	1.26	1.46	1.71	1.99	1.52
100 to 200	66	1.35	1.57	1.79	2.05	2.27	1.81
up to 16	91	1.02	1.08	1.25	1.44	1.67	1.31
up to 40	144	1.03	1.16	1.41	1.74	2.07	1.49
up to 80	191	1.04	1.25	1.48	1.69	1.94	1.51
40 to 80	47	1.36	1.44	1.53	1.63	1.73	1.54

Approximately 300 random tests were run with 10 columns. 90 percent of them converged in a time below twice the optimal time. On average it took the method 52 percent longer than the optimal solution. The ratios increase with an increasing number of robots, where tests with robot numbers between 100 and 200 took significantly longer than tests with 10 to 40 robots. The about 200 tests run with 4 columns show very similar results. Although having a lower number of robots per test, the number of rows stays equivalent and therefore requires the same vertical movement distance. The bad ratio for tests with 10 columns and a high number of robots stems from the same issue observed on test level 5 seen in Figure 3.1; only the border columns are used to extend the formation downwards. This issue is mitigated by a lower number of columns and is consequently not observed as serious on the equivalent tests with 4 columns. Summarized the majority of the ratios lie between 1.0 and 2.0 with an average in the middle.

4 Discussion

The results testify to the success of the local method. A correct end configuration is reached independent of initial configuration and the number of robots. Furthermore, the convergence times show consistent and solid numbers, especially for small swarm sizes, where the median ratio is around 1.25. Thus, the method distributes the robots efficiently over the formation space and implements correct rules on how a robot chooses its desired formation place. The simulation environment created in the Unreal Engine adds certain hidden factors to the performance of the local method, such as the handling of collisions and the order in which Algorithm 3 is called for each individual robot. For this reason, the environment is well suited to test the robustness and adaptability of the method. Latter properties are also demonstrated as there are no substantial performance differences observed between any test runs based on the initial configuration. Only the size of the swarm shows a clear influence over the performance.

It lies in the nature of a local, decentralized system that it performs worse with a crowded configuration. Inferring a crowded environment is a hard task for an individual agent given its limited vision. Responding correctly to this information is challenging, as the right action is often defined by factors outside visibility. Generally, specific actions designed to respond to specific situations require an unambiguous interpretation of the available information. If this is impossible, these specific rules intended to lead to a more precise behavior might actually derogate the robustness of the method. Ultimately, a local method is limited by a sensible complexity before it loses its justification as a reasonable alternative to a global solution. Thus, the strength of a local method should come from its simplicity and the general rules that work with any configuration. To illustrate this observation, the crowded situation in test level 5 seen in the second image from bottom-right in Figure 3.1 can be analyzed. The robots on the outside columns travel downwards because each row is already filled. Hence, a better convergence time would be achieved if some robots pushed towards the center regardless of whether the row is filled, forcing robots on all columns to move down. The solution to the problem therefore requires a different policy to choose $\mathbf{p}_f^{impr}$ or $\mathbf{p}_f^{backup}$ when there is crowded space – a specific action responding to a specific situation. As previously argued such a design is not desirable.

The drop in performance with increasing number of robots is hereby partly explained, but nevertheless constitutes a shortcoming of the presented method. It is also noteworthy, that although the reported ratios can assess the general performance and robustness, they are heavily influenced by small changes in the initial configuration. In essence, the convergence time of the optimal solution is defined by the robot with the longest path to its formation place; a single robot far away prolongs the convergence time and subsequently lowers the ratio. The test levels were designed to account for this phenomenon by limiting the maximum distance to the formation origin based on the swarm size.

5 Conclusion and Outlook

The thesis introduces a local method that successfully organizes a swarm of robots into a column formation. The method is divided in two conceptual parts. Firstly, in the processing and communication of information, secondly in the determination of actions based on the former. Vision range V_s and V_l, the neighbor links, status S and the indicator B_{left}, B_{right} and B_{top} define the first part. The design of the dominance relation, the movement strategy and how to choose an improved or backup location make up the second part. All these parts are determined in formation space that translates to world space according to Equation 2.1. In future research, the concept of formation space can be extended to allow different formation types simply by changing the mapping from formation space to world space. Moreover, each swarm member keeps references to its neighbors and knows its formation place explicitly, this structural information can be used to model coordinated swarm movement or fast reorganization into a different formation.

References

[1] E. Sahin, S. Girgin, L. Bayindir, and A. Turgut, *Swarm Robotics*, 2008, pp. 87–100.

[2] W. Spears and D. Spears, *Physicomimetics: Physics-Based Swarm Intelligence*, 2012.

[3] L. E. Barnes, M. A. Fields, and K. P. Valavanis, "Swarm formation control utilizing elliptical surfaces and limiting functions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, pp. 1434–1445, 2009.

[4] J. Munkres, "Algorithms for the assignment and transportation problems," 1957.

[5] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Multi-robot system for artistic pattern formation," in *2011 IEEE International Conference on Robotics and Automation : (ICRA 2011) ; Shanghai, China, 9 - 13 May 2011*. IEEE, 2011, pp. 4512 – 4517.

[6] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Y. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed autonomous robotic systems : the 10th International Symposium*, ser. Springer Tracts in Advanced Robotics, A. Martinoli, F. Mondada, N. Correll, G. Mermoud, M. Egerstedt, M. A. Hsieh, L. E. Parker, and K. Støy, Eds., vol. 83. Springer, 2013, pp. 203 – 216.

[7] E. Owen and L. Montano, "Motion planning in dynamic environments using the velocity space," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 2833–2838.

[8] J. van den Berg, S. Guy, M. Lin, and D. Manocha, *Reciprocal n-Body Collision Avoidance*, 2011, pp. 3–19.

[9] Epic Games. (2019) Unreal Engine 4.14 Webpage. [Online]. Available: https://www.unrealengine.com/en-US/release-notes/414-release-notes-overview [Accessed 23.06.2019].

[10] Gurobi Optimization. (2019) Gurobi Webpage. [Online]. Available: http://www.gurobi.com [Accessed 25.04.2019].

[11] Blender Foundation. (2019) Blender Webpage. [Online]. Available: https://www.blender.org [Accessed 25.04.2019].

[12] Swarmanoid Project. (2019) Swarmanoid Project Webpage. [Online]. Available: http://www.swarmanoid.org [Accessed 14.03.2019].

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

Local methods for swarm navigation and formation

Verfasst von (in Druckschrift):

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Houska	Niklaus

Ich bestätige mit meiner Unterschrift:

- Ich habe keine im Merkblatt „Zitier-Knigge" beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Erlen, 30.06.2019	[illegible]

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.