NTNU – Trondheim
Norwegian University of
Science and Technology

Cooperative Behavioural Control for Omni-Wheeled Robots

Experiments and simulations for formation control with obstacle- and collision avoidance

Kristian Klausen

Master of Science in Engineering Cybernetics
Submission date: June 2013
Supervisor: Thor Inge Fossen, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology,
Mathematics and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name:	Kristian Klausen
Department:	Engineering Cybernetics
Thesis title (Norwegian):	Samarbeidende reguleringssystemer for roboter med omni-hjul
Thesis title (English):	Cooperative Behavioral Control for Omni-Wheeled Robots

Thesis Description: The purpose of the thesis is to implement and simulate coordinated and cooperative control systems for small robotic vehicles.

The following items should be considered:

1. Literature overview on cooperative control and synchronization.
2. Investigate methods for obstacle and collision avoidance using behavioral control, and conduct simulations.
3. Design the robot control systems and demonstrate features such as cooperation and formation control. Test the control laws using simulated data in Simulink based on a mathematical model of the robots.
4. Combine cooperative formation control with obstacle and collision avoidance from behavioral control. Test the control laws by experiments.
5. Simulate a search and rescue mission with multiple agents forming different formations depending on mission state, while avoiding collisions and obstacles.
6. Conclude your findings in a report.

Start date:	2013-01-21
Due date:	2013-06-17

Thesis performed at:	Department of Engineering Cybernetics, NTNU
Supervisor:	Professor Thor I. Fossen, Dept. of Eng. Cybernetics, NTNU

Abstract

This thesis considers the formation and behavioural control problem of a multi-robot system. A mathematical model of the mobile vehicles us presented, followed by an introduction to behavioural control. The Null-Space based Behavioural (NSB) control [Antonelli, Arrichiello, and Chiaverini, 2005] is presented, and is used to create task functions for obstacle- and collision avoidance. The formation problem is solved by a passivity-based approach presented in [Arcak, 2007]. The two controllers are combined to create a complete controller capable of maintaining group formations, while avoiding obstacles and inter-agent collisions. The results are verified by simulations and experiments on custom built robots.

Sammendrag

(Norwegian translation of the abstract)

Denne masteroppgaven tar for seg formasjon- og oppførselsregulering av et multi-robot system. En matematisk modell av robotene blir presentert, etterfult av en introduksjon til *behavioural control*. Deretter presenteres *Null-Space based Behavioural (NSB) control* [Antonelli, Arrichiello, and Chiaverini, 2005]. Denne teknikken blir brukt til å lage reguleringssløyfer for kollisjonsunngåelse. Formasjonsstyring løses ved en passitivitets-teknikk presentert i [Arcak, 2007]. De to kontrollstrukturene blir derretter slått sammen for å lage en komplett formasjons- og oppførselskontroller, som er i stand til å få en gruppe roboter til å innta en predefinert formasjon, uten å kollidere i hverandre eller andre objekter. Systemet blir testet i simuleringer og eksperimenter på spesialdesignede roboter.

Preface

Cybernetics, for me, is about robotics, control and software. During this thesis, I've had the pleasure of working with not just one, but a team of custom robots specifically designed and built for my work. I've been able to design and implement every aspect of the system from scratch, which has been a process of much anger and enjoyment. I am proud to say that my work has resulted in a team of agile, smart vehicles capable of some pretty cool things.

The robots built during my work with this thesis is a part of an ongoing project at the Department of Engineering Cybernetics, NTNU. The project was started by myself and Adam Leon Kleppe during the autumn of 2012, to create a team of mobile robots for demonstrative purposes. My work would not have been possible without the financial support of the department, and I hope they find good use of the results. Additional funds were acquired through a scholarship program from *Norsk Forening for Automatisering* (NFA), which greatly increased our chances of success.

I would also like to thank my supervisor Thor I. Fossen for guidance and overall support for this project.

Kristian Klausen

Contents

List of Figures

Chapter 1

Introduction

1.1 Background and motivation

Two topics will be covered in this thesis, namely behavioural control and cooperative control. The latter is a term used for systems which consists of multiple agents working together, while behavioural control tries to control how the agent interacts with the environment.

For example, imagine a vessel that is on-route to a target position, but finds an obstacle ahead. Behavioural control then dictates how the vessel should deviate from it's course to avoid collision. This is an on-line controller, and often consists of multiple task functions. The job of the behavioural control is then to control the output flow of each of these tasks.

Cooperative control is an umbrella-term for several types of problems. *The formation problem* occurs when multiple agents needs to keep a prescribed distance to each other. This can either be to minimize drag, or some other mission objective like the loading or transportation of cargo. Further, in the *agreement problem*, the goal is for the agents to reach an agreement over a common variable of interest. This can be a heading, phase, etc. This is also referred to as *synchronization*. The formation problem is usually thought of as a shifted agreement problem, but it can also be solved by a master-slave architecture where an agent follows a group leader via trajectory tracking.

The results of this thesis will be used to create a Multi-robot system for demonstration and research purposes at the Department of Engineering Cybernetics, in collaboration with several other students. See [Kleppe, 2013]. The construction and implementation of the robots has been done during the work of this thesis.

1.2 Previous work

Cooperative Control has been an active field of research for many years. Both [Cao et al., 2013] and [R. M. Murray, 2007] provide an excellent overview over the last decade of focus in multi-agent systems. Concerning formation control, [R. M. Murray, 2007] emphasises the work of [Dunbar and R. M. Murray, 2006] who considers the problem using receding horizon optimal control. Further, [Olfati-Saber and R. Murray, 2002] uses potential functions obtained from structural constraints of a desired formations, which results in a distributed bounded state-feedback for each vehicle. The use of LQR-based optimal design for linear agents are addressed in [Zhang et al., 2011].

Formation of underactuated marine vehicles is studied in [Børhaug, Pavlov, Panteley, et al., 2011], [Børhaug, Pavlov, and Pettersen, 2007] and [Børhaug, Pavlov, and Pettersen, 2006]. [Fax and R. Murray, 2004] uses the Nyquist theorem to prove formation stability using graph theory. In [Qu et al., 2008], a framework based on matrix theory is proposed to design cooperative control laws.

Distributed formation control of unicycle robots with non-holonomic constrains are discussed and addressed in [Sadowska and Kostic, 2012]. The proposed controller utilizes a virtual leader-approach to make the group of agents follow a desired trajectory.

In 2007, Murat Arcak wrote a paper about using passivity as a design tool for cooperative control [Arcak, 2007]. It described how one can incorporate communication topology and feedback of the interconnected system into a set of cascaded passive systems, and thus being able to prove stability by lyapunov analysis. The work ended up as a framework which could be used for a wide range of cooperative control problems, where the most important concerns the agreement problem and synchronization. For instance, [Ihle et al., 2007] uses this to make three boats synchronize along a path. Further, the passivity property proved to be quite useful for developing adaptive feedback laws. For instance, in [Bai et al., 2007], the authors simulates a multi-agent network where only one agent has information about the mission velocity. The rest of the agents uses adaptive control to converge to the correct path.

The framework can be used on a wide range of models and systems, as long as a passive controller can be made. This is trivially accomplished for Newtonian systems, but somewhat more involved for more complex systems. In [Skjetne et al., 2004], back-stepping is used for a Lagrangian system, and similar approaches can be used for Hamiltonian systems.

[Bai et al., 2011] summarizes much of this earlier work. It contains updated research and reference of use for [Arcak, 2007], and incorporates examples and several simulations and useful discussions. This book have been my main material on this

topic throughout this thesis.

A great introduction to behavioural control, it's history and connection to AI robotics is given in [Murphy, 2000]. Here, the author describes the transition from *hierarchical* paradigms, via the pure *reactive* paradigm to the current *hybrid* architectures. The term behavioural control is often said to be the reactive part of such architectures, where current sensory information is sensed to calculate the next move. The behavioural control laws presented in [Murphy, 2000] are mostly focused on potential vector fields.

A small breakthrough on behavioural control was made during the PhD thesis of Filippo Arrichiello, [Arrichiello, 2006]. In stead of using classical motorscheme or layerscheme to control the behaviour of mobile robots (for instance, how to avoid obstacles), he laid the groundwork for NSB; Null-space behavioural control. This works by defining convex task functions, and by forcing lower priority tasks to operate in *the Null-space* of higher priority tasks.

NSB makes it easy to define several tasks, and has been proved to be very useful in many situations. It has for example been used to solve the flocking problem [Antonelli, Arrichiello, and Chiaverini, 2008b], control of swarms of Unmanned Aerial Vehicles [Shiyou et al., 2011], and been tested on marine vessels for caging and transporting purposes [Arrichiello et al., 2011].

1.3 Contribution and scope of this thesis

The goal of this thesis is to investigate how to properly combine single-agent behavioural control with multiple-agent cooperative control; thus creating a set of agents capable of working together in a complex environment. This would enable the use of advanced *hybrid* architectures (involved planning, sensor redundancy and fault-handling) to be integrated with cooperative formation algorithms. Although these architectures are not studied further in this thesis, the formulation of formation control as a behaviour is an important first step (also, see [Verret, 2005]). The goals of this thesis can be divided into four parts:

Collision avoidance To maintain integrity of the robot, the agents must preserve a minimum distance to each other.

Obstacle avoidance The environment has several static obstacles, which must be kept at a safe distance. The robots must be able to handle a limited sensing range for the obstacles.

Formation control The agents should converge to a given formation, and hold it as long as it doesn't violate collision or obstacle constraints.

Target The agents, as a unit, should be able to solve simple problems, like moving to a target location.

The proposed control laws will be investigated in both simulations and on tests on actual robots.

To limit the scope of this thesis, the following assumptions are made:

- The agents are equipped with bi-directional communication links.
- The communication topology is known and constant. (This is not difficult to overcome, but in this thesis, it simplifies the amount of on-line calculations)
- Minimal or zero time lag in inter-agent communication. This is a rather hard constraint, but can be overcome by lowering the sampletime and protocoll effectiveness.
- The agents position in a common frame is measurable and known.

1.4 Organization of this thesis

The robots used in this project is described in Chapter 2. The robots have been constructed to have pure holonomic dynamics, and a mathematical model have been derived. The model developed here will be used in simulations throughout this thesis.

In Chapter 3, an introduction to behavioural control is given. The Null-Space based Behavioural (NSB) controller is introduced, and several useful properties is shown. A stability analysis proves the asymptotic stability of the global task. The task-functions needed for this thesis is chosen, and the controller is tested in a simulation.

An introduction to Cooperative Control is given in Chapter 4, followed by a problem statement. The formation problem is solved by a passivity based approach, giving high flexibility to the framework. The most important results from [Bai et al., 2011] is repeated, and two simulations and one experiment are conducted to illustrate the effects of varying the communication pattern.

The behavioural controller is combined with the solution to the formation problem in Chapter 5. Also, collision avoidance is added, and the overall system is deemed stable by a qualitative proof. Two approaches is given to account for both obstacle and collision avoidance. In the first direct approach, the resulting controller is suitable for kinematic movement and is tested in simulations and experiments. The second approach is introduced in Chapter 6, by utilizing a leader to generate a suitable mission velocity to avoid obstacle collisions. This procedure is tested

in a Search-and-Rescue mission, where five agents cooperate to move, search and enclose a target of interest.

Chapter 7 briefly summarizes the results in the preceding chapters, and generally concludes the thesis.

The digital appendix included with this thesis contains the Matlab and Simulink code used in the simulations and experiments, the source-code for the camera application and framework software written for the robot' embedded computer. It also includes several videos of the experiments. For further information on the contents, read the accompanied file `readme.txt`.

1.5 Nomenclature and notation

- The set of real numbers is denoted $\mathbb{R}$. The notation $\mathbb{R}_{\geq 0}$ denotes the set of all real non-negative numbers.
- All vectors in this thesis are column vectors. The set of p by 1 real vectors are denoted $\mathbb{R}^p$, while the set of matrices with p rows and q columns are denoted $\mathbb{R}^{p\times q}$.
- The transpose of a matrix $A \in \mathbb{R}^{p\times q}$ are denoted $A^{\mathrm{T}} \in \mathbb{R}^{q\times p}$.
- $\mathbf{I}_p$ is the identity matrix of dimension $\mathbb{R}^{p\times p}$.
- The $p \times q$ zero-matrix is denoted $\mathbf{0}_{p\times q}$, while the $p \times p$ zero matrix is denoted $\mathbf{0}_p$. Further, the null-vector and unity-vector of size $N \times 1$ is denoted 0_N and 1_N, respectively.
- Labeling of coordinate-systems are done by a letter in curly brackets, e.g. $\{n\}$. Vectors represented in that coordinate system is displayed with that letter in superscript, i.e. v^n.
- The superscript d represents a desired value. E.g. v^d. The superscript f represents a formation specification, e.g. x_i^f, where i is the agent index.
- The Kronecker product of matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$ is defined as

$$A \otimes B := \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nq} \tag{1.1}$$

  and satisfies the properties

$$(A \otimes B)^{\mathrm{T}} = A^{\mathrm{T}} \otimes B^{\mathrm{T}} \tag{1.2}$$

$$(A \otimes \mathbf{I}_p)(C \otimes \mathbf{I}_p) = (AC) \otimes \mathbf{I}_p \tag{1.3}$$

- The notation $\texttt{diag}\{K_1, K_2, \cdots, K_n\}$ represents the block-diagonal matrix

$$\begin{bmatrix} K_1 & \mathbf{0}_{p\times q} & \cdots & \mathbf{0}_{p\times q} \\ \mathbf{0}_{p\times q} & K_2 & \cdots & \mathbf{0}_{p\times q} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{p\times q} & \mathbf{0}_{p\times q} & \cdots & K_n \end{bmatrix} \tag{1.4}$$

  where $K_i \in \mathbb{R}^{p\times q},\ i = 1, \cdots n$.

- The cartesian product of the two sets $\mathscr{A}_1$ and $\mathscr{A}_2$ is denoted as $\mathscr{A}_1 \times \mathscr{A}_2$. The resulting set contains all possible combinations of the elements in $\mathscr{A}_1$ and $\mathscr{A}_2$.

- The notation $K = K^{\mathrm{T}} > 0$ means that K is a symmetric positive definite matrix, while $k > 0$ implies k is a positive scalar.

- $\mathscr{N}(A)$ and $\mathscr{R}(A)$ are the null space (kernel) and the range space of a matrix $A \in \mathbb{R}^{p\times q}$, respectively. The vector $x \in \mathscr{N}(A)$ if $Ax = 0_p$. Likewise, $y \in \mathscr{R}(A)$ if there exists a vector $v \neq 0_p$ s.t. $Ay = v$.

- The notation $\mathrm{atan2}(y, x)$ represents the four-quadrant *arctangent* of the real parts of the elements of x and y satisfying

$$-\pi \leq \mathrm{atan2}(y, x) \leq \pi \tag{1.5}$$

- The matrix A is denoted Hurwitz if every eigenvalue λ_i of A satisfies

$$\mathrm{Re}\,[\lambda_i] < 0 \tag{1.6}$$

  where $\mathrm{Re}\,[x]$ is the real part of x.

- A function is said to be C^k if its partial derivatives exist and are continuous up to order k

- Given a C^2 function $P : \mathbb{R}^p \to \mathbb{R}$ we denote by ∇P its gradient vector, and by $\nabla^2 P$ its Hessian matrix.

If not otherwise stated, all formation specifications are given in meters.

Chapter 2

The Omniwheeled Robot System

The robots that the results of this reasearch are beeing used on, is a rather special robot. It has a triangular shape, with three wheels called omniwheels. (Figure 2.2) These wheels can provide force in one direction, while still being able to have transverse movement. This allows the robot to have true planar movement, without non-holonomic constraints. This effect could also be achieved by other shapes (e.g. by a square shape using four wheels instead of three), but this would also increase the cost of the robot. Figure 2.1 displays the first prototype. The robots described here have been built as a part of the work with this thesis.

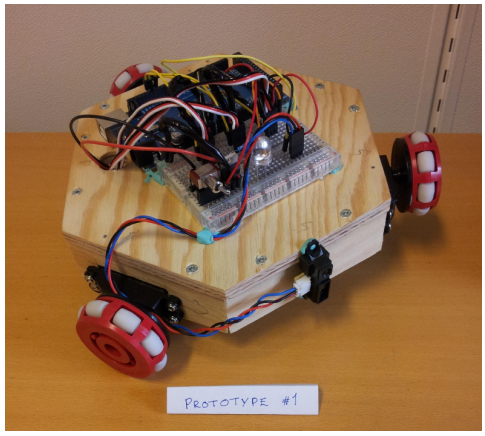


Figure 2.1: Picture of the prototype robot.

Figure 2.2: Image of an Omni-wheel (sometimes called a trans-wheel). *Image courtesy kornylak.com*

In this chapter, a brief introduction to the hardware on the robots are discussed, before a mathematical model of it's dynamics are derived. Further, the visual system is introduced.

2.1 Overview

As can be seen in Figure 2.3, the complete system consists of three distinct parts. The first is the actual robots and their individual computers. Each robot is equipped with a low-cost computer called a BeagleBone, which is described in more detail in Section 2.2. This computer contains all code needed to run a single robot.

The second part is a desktop PC running the visual system. The PC utilizes a camera to capture the location of each robot in the field, which is then transmitted to the robots using a network. More details of the visual system can be seen in Section 2.3. Please note that the PC doesn't do any control computation, every control algorithm is processed in a distributed fashion on each robot.

This network is the third part of the overall system, and can be any type of wired or wireless network. But, for simplicity in this project, the data is transmitted over a simple wired ethernet network. This made the developement and testing phase of the project much less troublesome, and would be easy to switch over to a wireless WiFi network later on.

All controllers presented in this project are developed in a Matlab/Simulink environment. By using a toolbox called Simulink Coder, these controllers can be exported to pure C++ code that can be interfaced by the software on the BeagleBone. This makes the developement cycle from simulation and testing to deployment on the actual robots very easy. Further, the BeagleBone logs all incomming and outgoing data from the controller application to a memory card. This memory card is also shared over the network, enabeling the developer to quickly analyse the

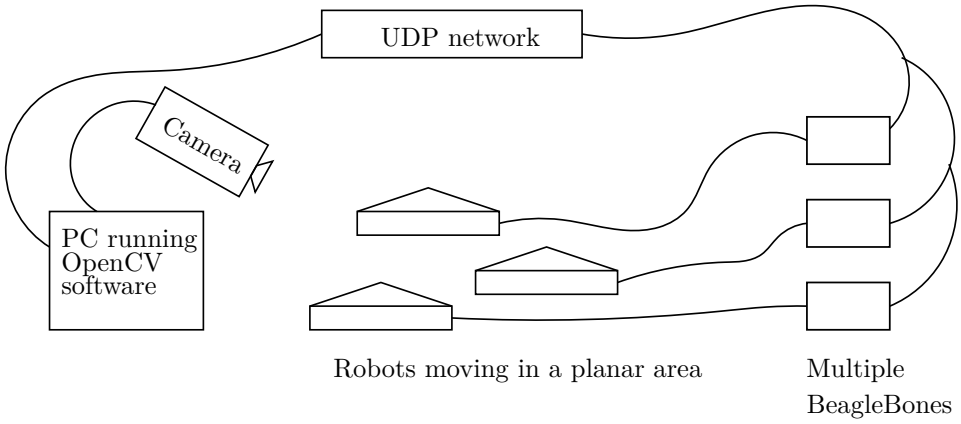


Figure 2.3: An overview of the complete system. The PC distributes updated robot location to each robot.

logs after each test. An application has been created in Matlab which processes the logs and enables advanced plotting capabilities.

2.2 Hardware specification

The heart of the robots is a low-cost computer called a BeagleBone [Beagleboard][1]. The BeagleBone is a small unix-based machine, equipped with a 720 MHz ARM Cortex-A8 processor. This allows us to use hardware like motors and sensors usually designed for smaller embedded processors, with the advantages of a full-fledged operating system. An image of the BeagleBone is seen in Figure 2.4.

This platform was chosen for a number of reasons. Firstly, compared to many other solutions this is really cheap with each computer costing only $89. Secondly, another variant of this card has been used with success on similar projects in the past [Skøien and Vermeer, 2010]. Here, the authors also goes into detail on the availible operating systems, and goes far to recomend the Angstorm distribution. This is a popular linux-variant in the community surrounding linux-based embedded systems, and can be installed on a wide array of computers (among others, both the Raspberry PI [RaspberryPI][2] and BeagleBone are supported).

The operating system gives the developer easy access to hardware support systems like serial communication, interfaces like i2c and spi, and PWM[3] output.

[1] See organization homepage at http://beagleboard.org.
[2] See organization homepage at http://www.raspberrypi.org/.
[3] Pulse-Width Modulated signal. This can be used to control the speed of attached motors.

Figure 2.4: The BeagleBone hardware platform. *Image courtesy of beagleboard.org*

2.3 Visual system

To be able to track the position of each individual robot, a simple visual system was created. It consists of a single color-camera, connected to a PC. The pc utilizes an open source visual backend called OpenCV [OpenCV]. This framework allows easy developement of advanced camera applications in C++. We have chosen to track each robot using a predefined coloured piece of paper attached to each robot.

The image is originally captured in the ordinary colour format RGB. Each pixel of the image is stored in three separate channels, each describing how much Red, Green and Yellow that pixel contains. Before image-processing teqniques are used on the image, it must first be transformed to another format called HSV.

HSV stands for Hue, Saturation and Value. Instead of separating the colors in the three channels, the color is stored in the single channel Hue. This channel is usually represented by a value ranging from 0°to 360°. Saturation and Value describes how much black and white are represented, respectively. An image describing the color-model can be found in Figure C.1. This image representation makes it easy to filter out certain colors, since we can apply ranges to each color. Green for instance, is located around 120°.

2.3.1 Locating the robots in the image

Getting the location of the robot is a multiple-step process. First, a range for each color must be decided. As noted, Green is located at around 120°, so a range from 90°to 160°seems usefull. Further, we require a high value for saturation to only include highly saturated pure colors. The result is a binary image, where the pixel are white if that particullar pixel in the original image satisifed the range, otherwise black.

Figure 2.5 illustrates this first step. In Figure 2.5(a), the original image from the camera is given. Our goal is to find the location of the green square.

(a) Image taken from camera **(b)** Isolated green colors from the original image.

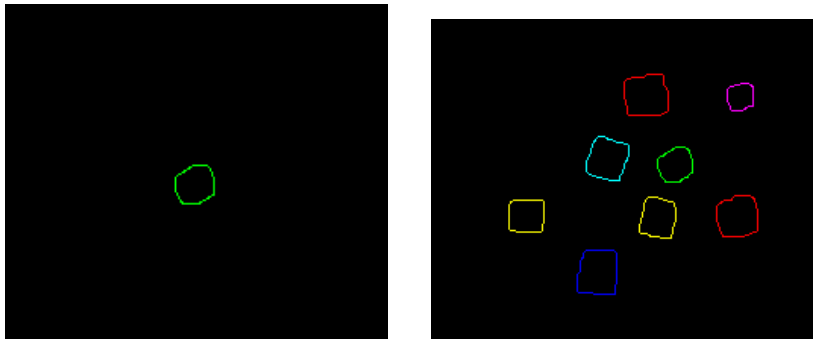
Figure 2.5: Colour localization using OpenCV.

As seen in Figure 2.5(b), the green square is clearly visible, but the image also contains some noise. This can be removed by a process called *erosion*. In simple terms, what this does is shrink the white parts of the image. This leaves the clear white square in the middle.

The next step is to find the edges of the square. The application uses an algorithm called the Canny Algorithm [Canny, 1986]. The implementation is a part of the OpenCV framework, so I will not go into details about it here. Figure 2.6(a) shows a zoomed-in image of the result.

Finally, now that the edges have been found (also called the contours of the shape), every distinct contour must be located. OpenCV has implemented an algorithm adapted from [Suzuki and Abe, 1985] which accomplishes this. The resulting output is the location of each distinct shape, including its area. The whole process is repeated for each of the six colours blue, red, green, cyan, yellow and magenta. An example image where all the colours are located can be seen in Figure 2.6(b).

The complete algorithm is run as fast as possible on the PC, with a resulting varying sample-time of about 3-4 Hz. To make the robots handle this correctly, an observer with the ability to do dead-reckoning when no new data is available is introduced in Section 2.6.



(a) Location of a single green marker is done.

(b) Multiple colour-markers have been found, together with their location.

Figure 2.6: Outline of the located colour markers.

2.4 Kinematics

In this section, the kinetics of the omniwheeled robot is derived. The notation and nomenclature used is similar to [Fossen, 2011].

The system is limited to planar movement, thus has three degrees of freedom (3 DOF) with movement in surge, sway and yaw. The geographic reference frame used in this thesis is denoted $\{n\}$, and the vector describing the vehicle' position and attitude in this reference frame is given by

$$\boldsymbol{\eta} := [x^n, y^n, \psi]^{\mathrm{T}} \tag{2.1}$$

where ψ is the attitude of the body-axis, and x^n, y^n is the North and East positions, respectively. The frame $\{n\}$ is thus named NED (North-East-Down).

Further, let the body-fixed frame, denoted $\{b\}$, be described by a principal rotation ψ about the positive downwards z-axis (see Figure 2.8). By defining

$$\boldsymbol{\nu} := [\dot{x}^b, \dot{y}^b, r]^{\mathrm{T}} \tag{2.2}$$

where $r := \dot{\psi}$, the conversion from NED to Body is given by

$$\dot{\boldsymbol{\eta}} = \mathbf{R}_b^n(\psi)\boldsymbol{\nu} \tag{2.3}$$

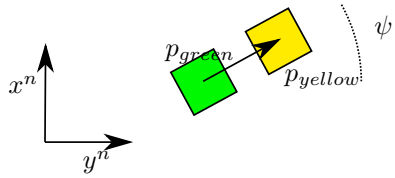
where $\mathbf{R}_b^n(\psi) \in SO(3)$ ([Fossen, 2011]), is a rotation matrix from $\{b\}$ to $\{n\}$ and is given by:

$$\mathbf{R}_b^n(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

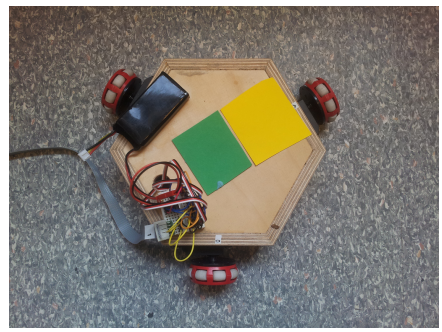
2.4.1 State measurement using camera data

The goal of the visual system is to capture the complete state of the robot, $\boldsymbol{\eta}$. The position $[x^n\ y^n]$ can be found by locating the appropriate colour-marker. Each robot has a distinct color in the middle of the robot body, to locate it's position. (e.g. the green marker in Figure 2.7(b)). But, in addition to the position, this state also includes the rotation angle ψ. To capture this, every robot is marked with two colour-markers. One denotes the position, while the other can be used to calculate the current angle of the robot. This is illustrated in Figure 2.7, and ψ can be calculated by:

$$\psi_{measured} = \text{atan2}(p_{yellow_y} - p_{green_y}, p_{yellow_x} - p_{green_x}) \tag{2.5}$$



(a) Illustration of the two color markers and how to calculate the measured robot angle ψ.



(b) The two color markers (green and yellow) on the robot.

Figure 2.7: Illustration and image of the color markers used on the robot. In this case, the green is used to measure position, while the yellow is used to calculate the current angle of robot.

2.5 Control allocation

In this section, the relationship between the motor inputs and the vehicle dynamics are derived. This is done to show that with this particular robot, $\boldsymbol{\nu}$ is directly manipulable.

Figure 2.8 gives the structural overview of the robot. It's equipped with three motors, marked A, B and C. The motors are bi-directional, and are directly manipulable. The relationship between the velocity contribution and shaft speed ω_i of one wheel is given by:

$$v_i = \omega_i r_{wheel},\ i \in \{A, B, C\} \tag{2.6}$$

where r_{wheel} is the radius of the wheel, and the positive direction of the shaft rotation is given by Figure 2.9.

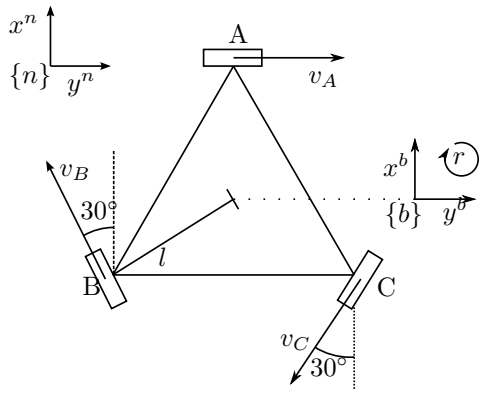


Figure 2.8: Figure describing the omniwheel

For notational simplicity, let

$$v_x := \dot{x}^b \tag{2.7}$$

$$v_y := \dot{y}^b \tag{2.8}$$

By examining the contribution velocities of each motor,

$$v_x = \frac{1}{3}(v_B \cos 30^\circ - v_C \cos 30^\circ) \tag{2.9}$$

$$v_y = \frac{1}{3}(v_A - v_B \sin 30^\circ - v_C \sin 30^\circ) \tag{2.10}$$

$$r = \frac{1}{3l}(v_A + v_B + v_C) \tag{2.11}$$

(l is, as seen in Figure 2.8, the length from the motor to the robot' center).

By writing on vector-form, and using (2.6)

$$\underbrace{\begin{bmatrix} v_x \\ v_y \\ r \end{bmatrix}}_{\boldsymbol{\nu}} = \underbrace{\frac{r_{wheel}}{3}\begin{bmatrix} 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{l} & \frac{1}{l} & \frac{1}{l} \end{bmatrix}}_{:=\mathbf{T}} \underbrace{\begin{bmatrix} \omega_A \\ \omega_B \\ \omega_C \end{bmatrix}}_{:=\tau} \tag{2.12}$$

gives

$$\boldsymbol{\nu} = \mathbf{T}\boldsymbol{\tau} \tag{2.13}$$

$\mathbf{T} \in \mathbb{R}^{3\times 3}$ is invertible, and by supplying a desired body-velocity vector $\boldsymbol{\nu}_d$, the control input vector $\boldsymbol{\tau}$ is given by

$$\boldsymbol{\tau} = \mathbf{T}^{-1}\boldsymbol{\nu}_d \tag{2.14}$$

The robot has 3 DOF and 3 control inputs, and by (2.14), the robot is *fully actuated*.

2.5.1 Motor dynamics and speed restrictions

The motors used on the robots are often called Continuous Servos. They provide a complete boxed-solution with internal gears, and an easy interface for velocity setpoint. They are precise, but have a rather slow max rotational speed. The actual robot will therefore have a maximum travel speed of about 85 mm/s. This is accounted for in all the tests done on the robots, but most of the simulations was done before the robots were actually constructed. Thus, for the simulations, an arbitrary maximum velocity of 1 m/s was used.

Please note that the motors will introduce a small time-constant, but this is assumed negligible.

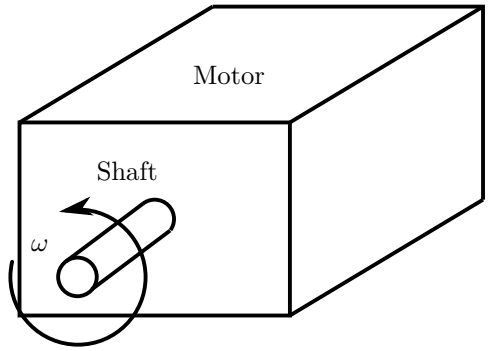


Figure 2.9: Motor and positive shaft direction

2.5.2 Control input to rotation speed

The control-input to the motors must be translated from the desired shaft speed given by the controllers. A simple test-setup where multiple control inputs were set, and the resulting shaft speed recorded was conducted. The results can be seen in Figure 2.10. It is seen that the motors have a maximum shaft-speed of about 4.8 rad/s. Further, the relationship is not linear, and a slack is present at zero speed (that is, at least an input of 4 is needed for the motor to turn). From this data, a non-linear lookup table was created, using linear regression between the datapoints. This relationship is used to convert desired shaft speeds to control inputs for the rest of this thesis.

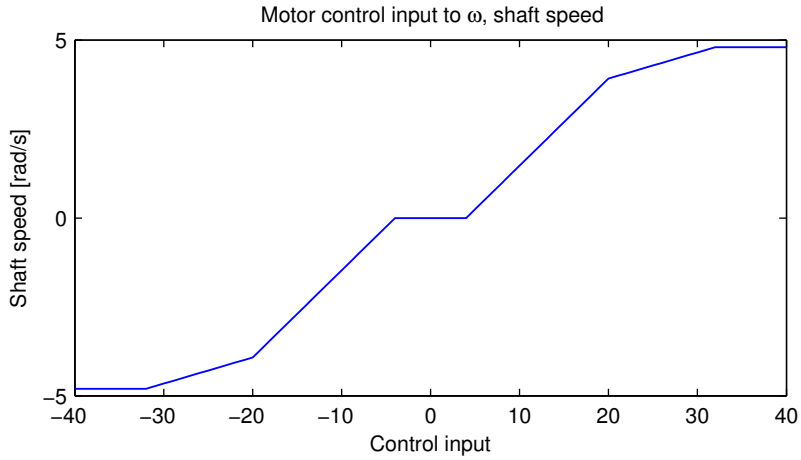


Figure 2.10: Relationship between motor control input and resulting shaft speed

2.6 Observer design

To be able to cope with error-prone measurements and capability for dead-reckoning, an observer is created. The system equations are repeated for clearence:

$$\dot{\boldsymbol{\eta}} = \mathbf{R}_b^n(\psi)\boldsymbol{\nu} \tag{2.15}$$
$$\mathbf{y} = \boldsymbol{\eta} \tag{2.16}$$

where $\mathbf{y} \in \mathbb{R}^3$ are the system measurements. These measurements are received from the camera, where in the example illustrated in Figure 2.7, the green marker is used to measure x^n and y^n, while the angle ψ is calculated from (2.5). The observer equations can be found by copying the dynamics from (2.15) and (2.16):

$$\dot{\hat{\boldsymbol{\eta}}} = \mathbf{R}_b^n(\psi)\boldsymbol{\nu} + \mathbf{K}_{pos}\tilde{\mathbf{y}} \tag{2.17}$$
$$\hat{\mathbf{y}} = \hat{\boldsymbol{\eta}} \tag{2.18}$$

where $\tilde{\mathbf{y}} := \mathbf{y} - \hat{\mathbf{y}}$ and $\mathbf{K}_{pos} = \mathbf{K}_{pos}^{\mathrm{T}} > 0, \in \mathbb{R}^{3\times 3}$ is a matrix of gains. Note that the control input $\boldsymbol{\nu} = \mathbf{T}\boldsymbol{\tau}$ is known. By defining $\tilde{\boldsymbol{\eta}} := \boldsymbol{\eta} - \hat{\boldsymbol{\eta}}$, the error dynamics becomes

$$\dot{\tilde{\boldsymbol{\eta}}} = \dot{\boldsymbol{\eta}} - \dot{\hat{\boldsymbol{\eta}}} \tag{2.19}$$
$$= \mathbf{R}_b^n(\psi)\boldsymbol{\nu} - \mathbf{R}_b^n(\psi)\boldsymbol{\nu} - K_{pos}\tilde{\mathbf{y}} \tag{2.20}$$
$$= -K_{pos}\tilde{\boldsymbol{\eta}} \tag{2.21}$$

which makes the equilibrium point $\tilde{\boldsymbol{\eta}} = 0$ globally asymptotically stable (GAS) by [Khalil, 2002, Theorem 4.5] since $-K_{pos}$ is Hurwitz.

To be able to handle dead-reckoning, a corrector-predictor representation of the non-linear observer are used [Fossen, 2011, Section 11.3.4]. By using Euler integration, the discrete time corrector-predictor formulation becomes:

$$\text{Corrector}: \quad \hat{\boldsymbol{\eta}}(k) = \bar{\boldsymbol{\eta}}(k) + h\mathbf{K}_{pos}\left[\mathbf{y}(k) - \bar{\mathbf{y}}\right] \tag{2.22}$$
$$\text{Predictor}: \quad \bar{\boldsymbol{\eta}}(k+1) = \hat{\boldsymbol{\eta}}(k) + h\mathbf{R}_b^n(\psi)\boldsymbol{\nu} \tag{2.23}$$

where h is the sampling time. This structure allows us to disable the corrector when new measurement are unavailable.

Program 2.1: Pseudo-code for observer

```
h = 0.1 sampling time
η̂ = η_0   initial state vector

while estimating

K_pos = h diag{k_pos, k_pos, k_heading}
if no new measurement
        K_pos = 0_{3×3}
endif

y = camera measurement
η̂ = η̄ + K_pos [y − η̄]
ν = Control system update
η̄ = η̂ + h R_b^n(ψ) ν
```

2.7 The robot as a base for kinematic control

As we have seen in the previous sections, the dynamics of the robot are very simple. This can be advantageous, as it is now possible to create advanced cooperative control laws, that operate on the kinematics of the robot. To see this, assume now that $\boldsymbol{\eta}$, the robot' position and attitude in $\{n\}$, is measurable and known. By keeping $r = \psi = 0$, the rotation matrix $\mathbf{R}_b^n(\psi)$ in (2.3) reduces to $\mathbf{I}_3$. Even with uncertainties in $\mathbf{T}$, this can be accomplished with a controller. Then, the following is valid:

$$\dot{x}^n = v_x \tag{2.24}$$

$$\dot{y}^n = v_y \tag{2.25}$$

This means that we are able to develop pure kinematic controllers, and still be able to test them in experiments. Since the purpose of this paper is to look at kinematic controllers, these robots provides an excellent platform for experiments and simulations.

Chapter 3

Agent-behavioural Control

This chapter will present the concept of behavioural control, and will introduce the method of Null-Space-based Behavioural (NSB) control. The concepts developed here will be used as a framework to combine the cooperative control objectives presented in Chapter 4 with other objectives like collision avoidance. As the main focus of this thesis is on the multi-robot system, the presentation of this topic may be somewhat brief. The reader is encouraged to consult [Arrichiello, 2006] or [Antonelli, Arrichiello, and Chiaverini, 2005] for a more detailed perspective.

3.1 Behavioural Control

Most robots today must be designed to work in complex environments, and to be able to process the incoming sensor data in real time. These environments can be complex to describe mathematically, and thus it is difficult to create a single control law that encompasses all available information at once. It would seem like a wise idea to try to divide the overall objectives of the robot into several sub-problems or tasks. For instance, instead of an objective like "Move to the desired target without crashing into anything", one could split it into two objectives describing each goal separately. In our case, the first objective could be keep a minimum distance to an obstacle, while the second could be to move towards a target location. Figure 3.1 gives a simple overview of a generic behavioural controller.

The concept of behavioural control concerns how these individual tasks should be combined to create a single motion output to the robot. In general, one can distinguish between two classes of behavioural control, which are *competitive* and *cooperative*. In a typical competitive scheme, a simple supervisor selects a single

task that is active at a given time (dependent on sensor information), and let that task control the motion of the robot. An example of such a competitive scheme is the *layered* control system [Brooks, 1986].

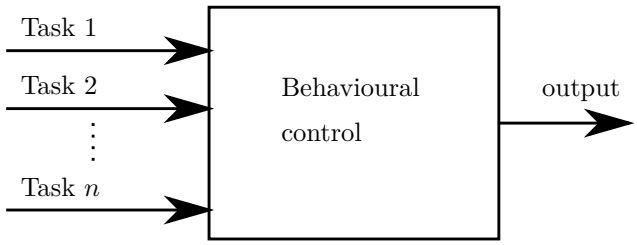


Figure 3.1: A schematic overview of a generic behavioural controller. The controller specifies how the output from each task should be combined to form a single motion output to the robot.

In a cooperative scheme, a supervisor uses a weighted sum of the task output. These weights are typically constants, and can be changed from different sets depending on the current robot situation. It thus consists of a linear combination of the output from each task. At a given moment, no task is completely achieved, but a compromise solution is found [Arrichiello, 2006]. The motor schema is an example of a cooperative behavioural system, and is discussed in more detail in [Emery and Balch, 2001].

The reader should note that when discussing behavioural control types classified as *cooperative*, it does not have a direct link to the term *cooperative control*, which in this thesis is reserved for control algorithms used to control multi-robot systems. Also, the terms task and objective will be used interchangeably to describe a desired sub-goal for the system. A task is usually represented by a potential function, and the overall goal is to get the output to converge to a desired value.

3.2 Introduction to NSB

NSB is an abbreviation for Null-Space Behavioural control. Like the name suggests, it utilizes the null-space of the task functions to provide a framework for task priority. The theory is based on [Arrichiello, 2006], and examples of use can be found in [Antonelli, Arrichiello, and Chiaverini, 2008a], [Antonelli, Arrichiello, Chiaverini, and Cassino, 2009], [Shiyou et al., 2011] and [Arrichiello et al., 2011].

This framework works by controlling the output of a potential function $\boldsymbol{\sigma} = \mathbf{f}(\mathbf{p})$, where $\mathbf{p} \in \mathbb{R}^p$ position information, $\boldsymbol{\sigma} \in \mathbb{R}^m$ is the task-variable to be controlled, and $\mathbf{f}(\mathbf{p}) : \mathbb{R}^p \rightarrow \mathbb{R}^m$. In other use-cases, $\mathbf{p}$ can be some other configuration

parameter. In this thesis, the NSB framework is used to control the position $\mathbf{p}$ of a single agent moving in an environment. Each task consists of a separate potential function.

The desired value of $\boldsymbol{\sigma}$ is $\boldsymbol{\sigma}^d(t) \in \mathbb{R}^m$, and our goal is now to find trajectories of $\mathbf{p}(t)$ which leads to $\boldsymbol{\sigma} - \boldsymbol{\sigma}^d = 0$. To find the dynamics of $\boldsymbol{\sigma}$, we differentiate:

$$\dot{\sigma} = \frac{\partial \mathbf{f}(\mathbf{p})}{\partial \mathbf{p}} \mathbf{v} := \mathbf{J}(\mathbf{p})\mathbf{v} \tag{3.1}$$

where $\mathbf{v} = \dot{\mathbf{p}}$ and

$$\mathbf{J} = \frac{\partial \mathbf{f}(\mathbf{p})}{\partial \mathbf{p}} \in \mathbb{R}^{m \times n} \tag{3.2}$$

is the output Jacobian matrix. An effective way to generate desired motion references $\mathbf{p}^d(t)$ from the desired potential function output $\boldsymbol{\sigma}^d(t)$ is to act at the differential level by inverting the mapping in (3.1). To extract a desired agent velocity $\mathbf{v}^d$ from this, one can seek the minimum-norm velocity:

$$\mathbf{v}^d = \mathbf{J}^\dagger \dot{\boldsymbol{\sigma}}^d \tag{3.3}$$

where

$$\mathbf{J}^\dagger := \mathbf{J}^{\mathrm{T}}(\mathbf{J}\mathbf{J}^{\mathrm{T}})^{-1} \tag{3.4}$$

To avoid numerical drift in the integration, [Arrichiello, 2006] suggests a closed-loop version of the algorithm. By defining $\tilde{\boldsymbol{\sigma}} := \boldsymbol{\sigma}^d - \boldsymbol{\sigma}$, the proposed velocity reference $\mathbf{v}^d$ is now given by:

$$\mathbf{v}^d = \mathbf{J}^\dagger(\dot{\boldsymbol{\sigma}}^d + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}) \tag{3.5}$$

where $\boldsymbol{\Lambda}$ is a suitable constant positive definite matrix of gains, and $\boldsymbol{\sigma}^d$ is the desired task-variable value.

3.3 Multiple tasks

For a single task, (3.5) will make the agent tend to the minimum of the potential function $\mathbf{f}(\mathbf{p})$. For multiple tasks, a merging of different task output velocities is needed. As discussed earlier, several schemes have been used with success in the past, including the competitive layered-scheme and the cooperative motor-scheme. NSB is a cooperative scheme that lets the designer set priorities on tasks,

where the lower-priority tasks only can move in the null-space of those with higher priority.

First, choose the i'th task velocity as

$$\mathbf{v}_i = \mathbf{J}_i^{\dagger}(\dot{\boldsymbol{\sigma}}_{i,d} + \boldsymbol{\Lambda}_i \tilde{\boldsymbol{\sigma}}_i) \tag{3.6}$$

For illustrative purposes, lets assume that task i represents a task with priority i, i.e. task $i = 1$ represents the highest priority task. By defining $\mathcal{N}_i \in \mathbb{R}^{p \times p}$, the nullspace of task i as

$$\mathcal{N}_i := \mathbf{I} - \mathbf{J}_i^{\dagger}\mathbf{J}_i \tag{3.7}$$

Multiple tasks can be merged together by letting lower priority tasks move in the null-space of those with higher priority by, e.g. this 3-task example:

$$\mathbf{v}^d = \mathbf{v}_1 + \mathcal{N}_1[\mathbf{v}_2 + \mathcal{N}_2\mathbf{v}_3] \tag{3.8}$$

A geometrical interpretation of a two-task example is given in Figure 3.2. Here, the two task outputs $\mathbf{v}_1$ and $\mathbf{v}_2$ are combined to create the NSB output $\mathbf{v}^d$ (blue vector). This is done by projecting the lower-priority task $\mathbf{v}_2$ onto the null-space of $\mathbf{v}_1$, resulting in the net output $\mathbf{v}^d = \mathbf{v}_1 + \mathcal{N}_1\mathbf{v}_2$.

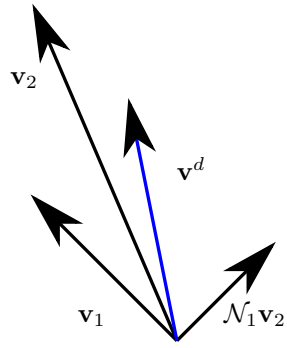


Figure 3.2: A geometrical interpretation of a two-task example, and the corresponding output from the NSB controller. $\mathbf{v}_2$ is projected onto the null-space of $\mathbf{v}_1$, and the net output is given by the blue vector $\mathbf{v}^d$.

To control which task should be active at a given time, the term *task manager* is often used in the litterature. In [Arrichiello, 2006] it is used to describe a mechanism that sets the priority of the tasks dynamically, and is also able to deactivate a task. In this thesis, a slightly different scheme is introduced. That is, rather than having

a single task manager, each task is supplied with it's own activation mechanism that is able to activate or deactivate that task. The result is much the same as with the single task manager, but a higher form of modularity is achieved. Figure 3.3 shows a structure overview of a 3-task example. The task activator decides the state of the switch depending on the input velocities and optionally other sensory information. A concrete example of a task activator is given for the obstacle avoidance task in Section 3.5.1.

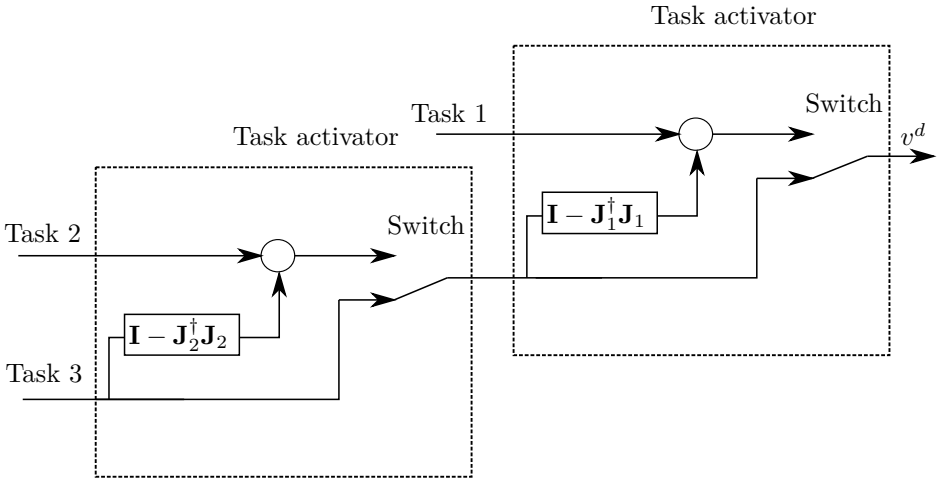


Figure 3.3: An overview of a typical NSB controller. Here there are three tasks present, and their output velocity is labeled Task 1-3. The task activators use the switch to set its own state as active or not. Active tasks are propagated up the chain, and the final output velocity is given in the figure as v^d.

3.4 Stability analysis

To analyse the convergence of the combination of all tasks, one can look at the convergence of each task individually. By definition, if every task reaches its desired value, the global task is completed. By examining the differential relationship between $\boldsymbol{\sigma}_1$, the highest-priority task, and the supplied output $\mathbf{v}^d$ in (3.8), we get from (3.1):

$$\dot{\boldsymbol{\sigma}}_1 = \mathbf{J}_1 \mathbf{v}^d \tag{3.9}$$

$$= \mathbf{J}_1 \mathbf{v}_1 \tag{3.10}$$

since $J_i \mathcal{N}_i = 0$. By further inserting the task velocity $\mathbf{v}_1$ from (3.6):

$$\dot{\boldsymbol{\sigma}}_1 = \dot{\boldsymbol{\sigma}}_{1,d} + \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 \tag{3.11}$$

the equation can be reformulated as (remember, $\tilde{\boldsymbol{\sigma}}_1 = \boldsymbol{\sigma}_{1,d} - \boldsymbol{\sigma}_1$)

$$\dot{\tilde{\boldsymbol{\sigma}}}_1 = -\boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 \tag{3.12}$$

which proves exponential stability of the highest priority task. [Arrichiello, 2006] further proves the stability of the lower priority task, which holds *as long as they do not conflict with higher priority tasks.* It is for this reason the task-manager is employed, as it is sometimes necessary to deactivate a task if it is no longer needed, to make sure a lower priority objective is fulfilled.

3.5 Choosing task functions

The robots in this thesis should be designed to move in an environment with static obstacles. Further, there is need for a simple "move to target location" task. In the following sections, these task-functions will be specified.

3.5.1 Obstacle avoidance

The ability to avoid obstacles is of crucial importance to maintain the integrity of the vehicle. Thus, this task is usually ran with the highest priority. This task should be activated when an obstacle is present near the agent, and the agent is moving towards the obstacle. The goal of the task is to keep the agent at a safe distance $d_o \in \mathbb{R}_{>0}$ from the obstacle located at $\mathbf{p}_0 \in \mathbb{R}^p$, thus creating a circle around the obstacle location. The task potential function can be chosen as:

$$\sigma_o = ||\mathbf{p} - \mathbf{p}_0|| \in \mathbb{R} \tag{3.13}$$

with the corresponding desired value

$$\sigma_{o,d} = d_o \tag{3.14}$$

The Jacobian of this task can be calculated according to (3.1)

$$\begin{aligned}\mathbf{J}_o &= \frac{\partial}{\partial \mathbf{p}}||\mathbf{p}-\mathbf{p}_0|| \\ &= \frac{(\mathbf{p}-\mathbf{p}_0)^{\mathrm{T}}}{||\mathbf{p}-\mathbf{p}_0||} \in \mathbb{R}^{1\times p}\end{aligned} \tag{3.15}$$

According to (3.5), the task-velocity output can be calculated as

$$\mathbf{v}_o = \mathbf{J}_o^{\dagger}\lambda_o(d_o - ||\mathbf{p}-\mathbf{p}_0||) \tag{3.16}$$

where λ_o is a suitable positive gain.

It is worth noticing that the agent-to-obstacle vector $\hat{\mathbf{r}}$ can be defined from the Jacobian as $\hat{\mathbf{r}} := \mathbf{J}_o^{\mathrm{T}}$. Then, one can geometrically conclude that the null-space of this task function will be movement along the tangential line of the circle (with radius d_o) centered at $\mathbf{p}_o$. Thus,

$$\mathcal{N}_o = \mathbf{I} - \mathbf{J}_o^{\dagger}\mathbf{J}_o = \mathbf{I} - \hat{\mathbf{r}}\hat{\mathbf{r}}^{\mathrm{T}} \tag{3.17}$$

3.5.2 Reach target location

Given a target location $\mathbf{p}_t \in \mathbb{R}^p$, one can simply choose ([Arrichiello, 2006]):

$$\boldsymbol{\sigma}_t = \mathbf{p} \in \mathbb{R}^p \tag{3.18}$$

$$\boldsymbol{\sigma}_{t,d} = \mathbf{p}_t \tag{3.19}$$

The resulting Jacobian becomes

$$\mathbf{J} = \mathbf{I} \in \mathbb{R}^{p\times p} \tag{3.20}$$

As before, the task-velocity is, by (3.5):

$$\mathbf{v}_t = \Lambda_t(\mathbf{p}_t - \mathbf{p}) \tag{3.21}$$

where $\Lambda_t \in \mathbb{R}^{p\times p}$ is a positive definite diagonal matrix of gains.

3.5.3 Implementational aspects and task activation

The direct and simple formulation of the obstacle avoidance task has several practical drawbacks that must be addressed. These are somewhat mentioned in [Arrichiello, 2006], and are repeated here with a proposed solution.

First, if the obstacle avoidance task had been active all the time, it would force the robot to linger on the circle around the obstacle. Since this task has the higher priority, and has a natural conflict with the second task, some rules for activation must be set. An intuitive point of view is that the task should only be active if a collision is actually imminent. This can be formulated by checking if the current velocity vector coincides with the obstacle circle. Figure 3.4 shows an illustration of a robot located at $\mathbf{p}$, and its current velocity (output from lower priority task) is towards the target at $\mathbf{p}_t$. By calculating the angle θ between the velocity vector and the obstacle center, $\hat{d}_c$ can be calculated by simple trigonometry. The activation rules can thus be expressed mathematically as:

The obstacle avoidance task is only active if

$$\hat{d}_o < d_o, \text{ and} \tag{3.22}$$

$$|\theta| \in \left[0, \frac{\pi}{2}\right) \tag{3.23}$$

Further, the obstacle-avoidance task is only activated if the robot is within a given range of the obstacle (to model limited sensor range). If not otherwise specified, the sensingrange is assumed to be 1.5 times the radii of the obstacle for the rest of this thesis.

Lastly, a problem occurs if the current velocity is pointed directly at the obstacle center. At this point, the Null-space of the obstacle is orthogonal to the velocity, and the resulting output is zero. In the real world, natural sensor noise would prevent such a situation to occur. Therefore, this problem is not considered nor discussed in this thesis. In either case, it would be easy to check for this special condition and simply push the robot slightly in the orthogonal direction to fix the problem.

A small comment must be made to the move to target-task. As can be seen from (3.21), this is a simple proportional controller. If the target is far away from the robot, the output velocity would be equally large. But, most robots have a natural speed restriction. Thus, if not otherwise noted, the output of this task function is saturated to the speed of the robots in use. For this thesis, as discussed in Chapter 2, this restriction is set to $1m/s$ in simulations and $85mm/s$ for the actual robots.

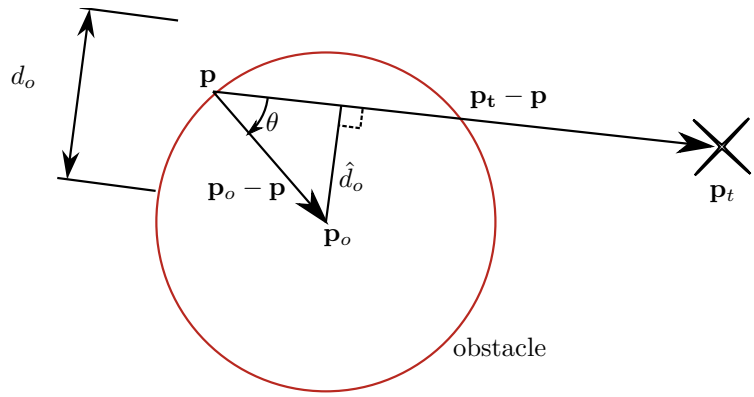


Figure 3.4: An overview of the calculations needed to check if the obstacle-task should be active. By looking at the current velocity vector of the robot ($\mathbf{p}_t - \mathbf{p}$ in the figure), it can be checked if a collision is imminent.

3.6 Single-agent simulation

To illustrate the capabilities of the NSB behavioural control, a simple test environment is simulated. Consider the vehicle modelled in Chapter 2. This is a pure kinematic system, and the resulting first-order dynamics is

$$\dot{\mathbf{x}} = \mathbf{u} \tag{3.24}$$

where $\mathbf{u} \in \mathbb{R}^2$ is the control (velocity) input, and $\mathbf{x} \in \mathbb{R}^2$ is the vehicle position. The simulated environment consists of a single static obstacle, located at $\mathbf{p}_o = [1.3, -0.3]^{\mathrm{T}}$. The initial position of the robot is $x(0) = [0,\ 0]^{\mathrm{T}}$. The target position is $\mathbf{p}_t = [3, 0]^{\mathrm{T}}$.

The goal is to reach the target position while avoiding the obstacle. Since it is vital to keep the robot' integrity, the obstacle avoidance task is chosen as task 1 with the highest priority. To reach the target location is the second task. The task activator is implemented according to the discussions in Section 3.5.3. This makes the obstacle avoidance task active only if a collision is imminent. An overview of the controller is given in Figure 3.5.

Further, the following controller parameters were used for optimal performance, found by trial and error:

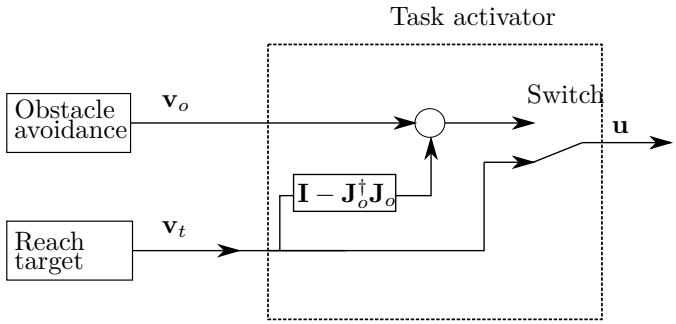


Figure 3.5: An overview of the controller used in the simulation example. The NSB controller consists of two tasks; obstacle avoidance and reach target location, where the first is of highest priority.

$$d_o = 0.5$$
$$\lambda_o = 1$$
$$\Lambda_t = \texttt{diag}\{8, 8\}$$

3.6.1 Results

The results of the simulation can be seen in Figure 3.7. The blue line illustrates the trajectory of the robot. The line is dashed when the obstacle avoidance task is deactivated, and is solid when it is active. One can see that the vehicle starts to turn away from the obstacle when it's getting close. Further, when the obstacle is no longer in the way, the obstacle-task is deactivated and the vehicle moves directly towards the target.

Figure 3.6 illustrates how the controller works when the robot reaches the obstacle. The length of the vectors are drawn for illustrative purposes, and vary in length depending on the control parameters for each task. In the figure, $\mathbf{x}$ is the current robot position. It is seen that the output from the goto-target task ($\mathbf{v}_t$) points inside the obstacle circle, and the obstacle task is thus active. The obstacle avoidance output $\mathbf{v}_o$ points towards the obstacle surface. $\mathbf{v}_t$ is then projected onto the null-space of $\mathbf{v}_o$, and the net output $\mathbf{u} = \mathbf{v}_o + (\mathbf{I}_2 - \mathbf{J}_o^\dagger \mathbf{J}_o)\mathbf{v}_t$ is illustrated by the blue arrow.

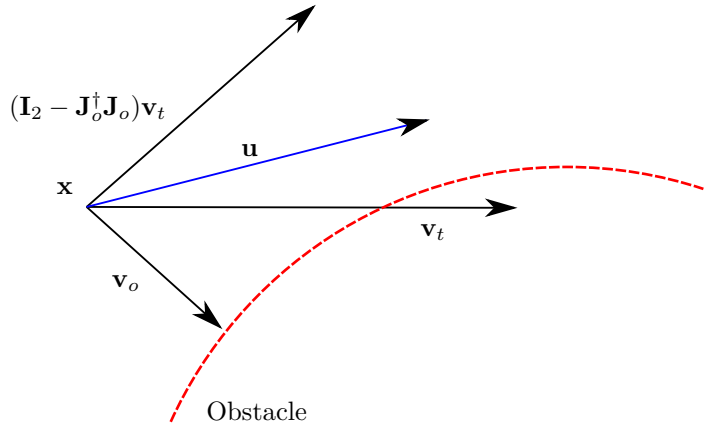


Figure 3.6: An illustration of the NSB control output. In this figure, the robot located at $\mathbf{x}$ must avoid the obstacle marked with a dashed red line. $\mathbf{v}_t$ is projected onto the null-space of $\mathbf{v}_o$, and the net output $\mathbf{u}$ steers the robot around the obstacle.

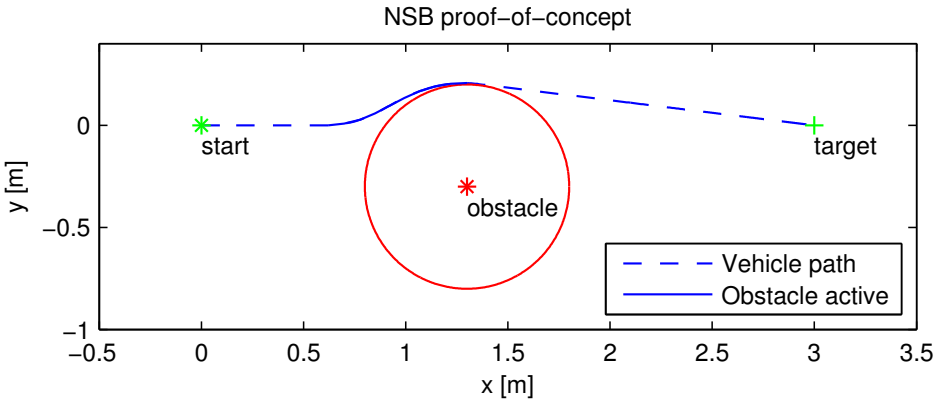


Figure 3.7: Illustration of the path taken by the vehicle in the simulation when avoiding an obstacle. The blue line illustrates the vehicle trajectory. The line is solid when the obstacle-avoidance task is active, dashed otherwise.

3.6.2 Discussion

As this proof-of-concept has shown, the NSB framework allows tasks to be fulfilled quite well. But, for each new task that is implemented, care must be taken when designing the activator for that task. Failure to do so will result in unexpected behaviour, and it can sometimes be hard to test for every possible case that a certain exception might occur. Multiple simulations should be set up to test these cases. The task activator used in this simulation have been tested under various conditions during development to ensure that the requirements are met.

The reader should note that the path taken around the obstacle is continuous, there is no oscillary behaviour. This is an important aspect concerning real-life usages. Further, it is seen that the path is not necessary optimal with respect to distance traversed. But it can be argued that if the sensor range was set to infinite (that is, the obstacle avoidance task can be active regardless of distance to that particular obstacle), the controller can be tuned to achieve nearly optimal paths. Further investigations of optimality concerning path length is beyond the scope of this thesis, as the main motivation for usage in this project is more focused on the usability of the NSB framework it selves.

3.7 Single-agent experiment

To further verify the proposed solution, the test is repeated with an actual robot. In this experiment, we consider a single robot operating in a static environment. The goal is for the robot to reach a predefined desired location, while avoiding a known obstacle. The camera continuously sends location data for both the robot and the obstacle position. Internally on the robot, the observer introduced in Section 2.6 is used to smooth the data, while a simple moving average over the last 5 samples are used on the obstacle location data.

The behavioural controller is illustrated in Figure 3.5. Here, obstacle avoidance is chosen as the task with the highest priority, while reaching the target location is secondary. Further, the angle is regulated to zero by a simple P-controller:

$$r^d = -k_p \psi \tag{3.25}$$

where $k_p \in \mathbb{R}_{>0}$. In this experiment, $k_p = 5$. Also note that before $\boldsymbol{\tau} = [\omega_A \ \omega_B \ \omega_C]^{\mathrm{T}}$ is applied to the motors, it is transformed to motor speeds as described in Section 2.5.2. The rotation matrix $R_n^b(\psi)$ is used to transform the velocities $\mathbf{u}$ in the NED-frame to the body-frame. This makes the behavioural controller consistent with the one used in the simulations. The controller runs at 10 Hz.

The rest of the set-up are done to match the simulation in the previous section as close as possible. Some parameters have been changed for the best performance:

$$d_o = 0.3 \tag{3.26}$$
$$\lambda_o = 1 \tag{3.27}$$
$$\Lambda_t = \texttt{diag}\{0.5, 0.5\} \tag{3.28}$$

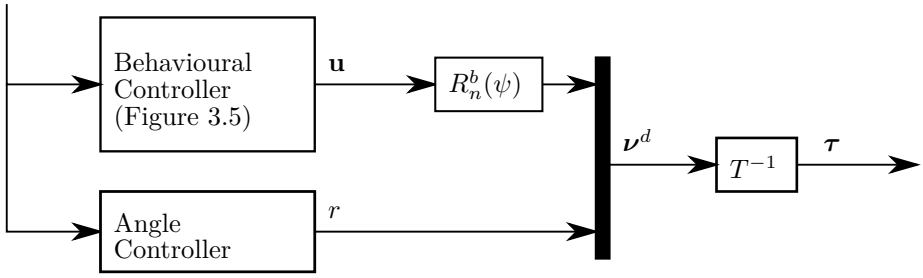


Figure 3.8: Illustration of the controller scheme used in the one-robot example. The black bar represents a vector concatenation.

3.7.1 Results

The results of the test can be viewed in Figure 3.9, where the blue line illustrates the path taken by the robot. Further, the solid line shows where the obstacle avoidance task is activated. The task is activated when the robot approaches the obstacle, and starts to avoid it. It is seen that when the robot has a clear path to its target, the task is deactivated. The robot reaches its desired position.

By comparing these test results with the one obtained through simulations Figure 3.5, it is seen that the behaviour of the robot is pretty similar to that of the simulation. The main difference is the way the robot approaches the obstacle. In the simulations, the approach is more smooth, while in the test the robot starts avoiding the obstacle more directly. There are multiple reasons for this, the main ones are different control parameters and the fact that the simulation was run with a smaller sampling time (100 Hz for simulation). The reader should also note that the simulations are not designed to be numerically comparable with the experiments, but to compare overall behaviour.

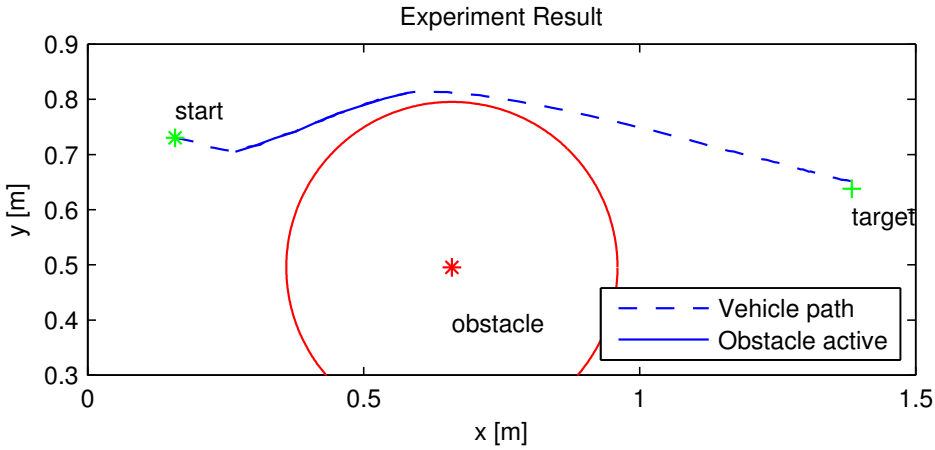


Figure 3.9: The results from the experiment. The trajectory (blue line) is shown as the vehicle avoids the obstacle. The line is solid when the obstacle-avoidance task is active, dashed otherwise.

3.7.2 Discussion

This test has proven that the use of the NSB control framework is feasible to use on actual robots. It has shown that the code generated from Simulink can be used to create such controllers. Further, it has shown that the whole interconnected system with the camera and BeagleBone hardware works. The robot was able to circumvent the obstacle, as it was supposed to. The ability to log data and directly import it to Matlab for plotting of results have proven to be a very effective workflow.

3.8 Summary

In this chapter we have looked at the properties of a behavioural control system, and the concept of Null-Space-based Behavioural (NSB) control was introduced. We have specified and investigated two interesting task functions, namely obstacle avoidance and a task to reach a desired location. The stability of the overall framework was proved, and the properties of the task activator was discussed.

To illustrate the properties of the framework, a simple simulation was presented. It contained a single robot moving in an environment with a single static obstacle. By using the described controller, the robot was able to circumvent the obstacle

and reach the target location. This test was repeated on an actual robot, which also was able to complete its task.

The framework developed here will provide a basis for the behaviour of each individual robot in the multi-robot system introduced in the next chapter. The system is put together in Chapter 5.

Chapter 4

Cooperative Control: The Formation Problem

Cooperative control is an emerging field in advanced robotics. In essence, it's the task of making multiple agents (e.g. several robotic arms, vehicles, etc) work together towards a single goal. By working together, multiple agents can often complete a task faster (like mapping a seabed), or the fact that the group can function even though one robot breaks down. Also, multiple cheap agents could outperform a single expensive agent.

Multiple types of cooperative systems can be formulated. The easiest to visualize would be the formation problem; where multiple agents should stay in a relative formation to reduce drag forces or cover a larger ground area. Other types of objectives could be target escorting, flocking, guarding and cooperative load transport.

Depending on the system, it is often desired to create distributed feedback laws. That is, the system should not be dependent on a master agent with global information. The agents should only have to rely on sensory information gathered by it selves and neighbour agents.

The proposed solution to the formation problem is a passivity based approach, presented in [Bai et al., 2011]. The authors presents a framework that can be utilized in a range of cooperative control problems. This includes, but is not limited to, agreement problems and formation problems. Also, in [Ihle et al., 2007], the framework is used for synchronized path-following for marine vessels.

4.1 Problem statement

This section is based on [Bai et al., 2011].

Consider a group of N agents, labeled $i = 1 \dots N$. Each agent consist of a vector $x_i \in \mathbb{R}^p$ to be coordinated with the rest of the group. The agents are able to communicate with each other, and the communication-topology is modelled by an undirected graph G. Thus, all communicationlines are bi-directional. Further, it is assumed that the G is connected and has ℓ links. The analysis is simplified by assigning an orientation to G, by considering one of the nodes to be at the negative end of the link. A brief introduction to some basic graph-theory is given in Appendix A.

The objective is to develop feedback-laws that guarantee the following behaviour: [Bai et al., 2011, A1), A2)]

A1) Each agent achieves in the limit a common velocity vector $v(t) \in \mathbb{R}^p$ prescribed for the group; that is

$$\lim_{t\to\infty} |\dot{x}_i - v(t)| = 0, \qquad i = 1, \cdots, N \tag{4.1}$$

A2) If agents i and j are connected by link k, then the difference variable z_k

$$z_{\mathrm{k}} := \sum_{l=1}^{\ell} d_{lk} x_l = \begin{cases} x_i - x_j \text{ if } k \in \mathscr{L}_i^+ \\ x_j - x_i \text{ if } k \in \mathscr{L}_i^- \end{cases} \tag{4.2}$$

converges to a prescribed compact set $\mathscr{A}_k \subset \mathbb{R}^p$, $k = 1, \cdots, \ell$, where d_{ik} is defined as in (A.1)

The target set $\mathscr{A}_k$ will have different forms depending on the application. If every agent has some x_i that should reach agreement, $\mathscr{A}_k$ will be the origin (for example when every agent should match phases). $\mathscr{A}_k$ can also be designed to make the agents move to specified relative distances, hence creating a formation.

It is important that the target set $\mathscr{A}_k$ is feasible. To develop a condition to this, some concatenated vectors are introduced:

$$\begin{aligned} \mathbf{x} &:= [x_1^{\mathrm{T}}, \cdots, x_N^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{pN} \\ \mathbf{z} &:= [z_1^{\mathrm{T}}, \cdots, z_\ell^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{p\ell} \end{aligned} \tag{4.3}$$

D can be partitioned in terms of its column vectors:

$$D = [D_1 | \cdots | D_\ell] \tag{4.4}$$

Further, the definition (4.2) can be rewritten as:

$$z_k = (D_k^{\mathrm{T}} \otimes \mathbf{I}_p)\mathbf{x} \tag{4.5}$$

$$\mathbf{z} = (D^{\mathrm{T}} \otimes \mathbf{I}_p)\mathbf{x} \tag{4.6}$$

which means that $\mathbf{z}$ is restricted to be in the range space $\mathscr{R}(D^{\mathrm{T}} \otimes \mathbf{I}_p)$ of $\mathbf{x}$ [Bai et al., 2011]. Thus, for the objective A2 to be feasible, the target sets $\mathscr{A}_k$ must satisfy

$$\{\mathscr{A}_1 \times \cdots \times \mathscr{A}_\ell\} \cap \mathscr{R}(D^{\mathrm{T}} \otimes \mathbf{I}_p) \neq \emptyset \tag{4.7}$$

4.2 The design steps

[Bai et al., 2011] suggests a two-step process, in which the goal is to render the set $\mathscr{A}_k$ in (4.2) asymptotically stable. For each step, the theory is first stated, followed by applying that step to the system used in this thesis.

4.2.1 Step 1: Internal feedback

Suppose you got a system on the form:

$$x_i = \mathscr{H}_i^o\{\tau_i\} \tag{4.8}$$

The system $\mathscr{H}_i^o$ describes the input-output dynamics with x_i as the output and τ_i as the input vector. The goal is to make the system passive from an external feedback signal u_i (to be designed in step 2) to the velocity vector:

$$y_i := \dot{x}_i - v(t) \tag{4.9}$$

where $v(t)$ is a common mission velocity to be achieved by the group, as described in (4.1).

In practice, this reduces to creating an internal feedback controller τ_i. The system with the feedback can often be described by a transfer function, and can be checked for positive realness which proves passivity ([Khalil, 2002]).

The system should now be on the form

$$\dot{x} = \mathscr{H}_i\{u_i\} + v(t) \tag{4.10}$$

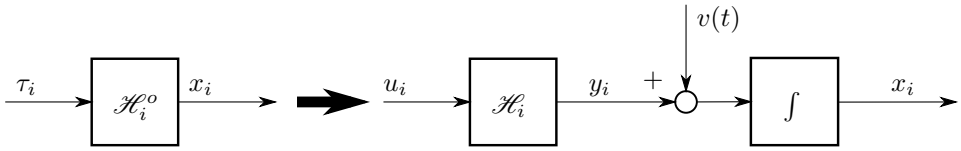


Figure 4.1: Description of Step 1. The system $\mathscr{H}_i^o$ is transformed so that $\mathscr{H}_i$ is passive from u_i to y_i

where $\mathscr{H}_i$, should be designed to be strictly passive.

$\mathscr{H}_i$ can be either a dynamic system, or a static non-linearity. If it is dynamic, it is assumed to have the form

$$\mathscr{H}_i : \begin{cases} \dot{\xi}_i &= f_i(\xi_i, u_i) \\ y_i &= h_i(\xi_i, u_i) \end{cases} \tag{4.11}$$

where y_i is the velocity error and $\xi_i \in \mathbb{R}^{n_i}$ is the state variable of subsystem $\mathscr{H}_i$. Both $f()$ and $h()$ are assumed to be C^2 functions such that

$$f_i(0, u_i) = 0 \Rightarrow u_i = 0 \tag{4.12}$$

and

$$h_i(0, 0) = 0 \tag{4.13}$$

As stated, $\mathscr{H}_i$ is designed to be strictly passive with a C^1, positive definite, radially unbounded storage function, making the origin of $\mathscr{H}_i$ globally asymptotically stable.

If $\mathscr{H}_i$ is a static block, it is restricted to be of the form

$$y_i = h_i(u_i) \tag{4.14}$$

where $h_i : \mathbb{R}^p \to \mathbb{R}^p$ is a locally Lipschitz function satisfying

$$u_i^{\mathrm{T}} y_i = u_i^{\mathrm{T}} h(u) > 0 \quad \forall u_i \neq 0 \tag{4.15}$$

thus making it strictly passive by [Khalil, 2002, Definition 6.1].

4.2.2 Internal feedback for the omniwheeled robot

As discussed in Chapter 2, the mathematical model for the omniwheeled robots can be expressed as

$$\dot{x}_i = \tau_i \tag{4.16}$$

where $x_i \in \mathbb{R}^2$ is the i'th robot position, and $\tau_i \in \mathbb{R}^2$ is the control input, namely the velocity. Note that this is a pure kinematic expression.

By designing the internal feedback as

$$\tau_i = u_i + v(t) \tag{4.17}$$

the resulting transformed system is simply

$$\mathscr{H}_i = h(u_i) = u_i \tag{4.18}$$

Indeed, $y_i = h(u_i) = u_i$ satisfies (4.15) since $u_i^{\mathrm{T}} u_i > 0 \ \forall u_i \neq 0$, rendering the transformed system strictly passive from u_i to y_i.

4.2.3 Step 2: External feedback

Before the suggested control-law is presented, let us first prepare the setup given in Step 1. Each of the agent' dynamics $i \in \{1, \ldots, N\}$ can be concatenated to form a single block diagram. By post-multiplying $\mathbf{x}$ with $D^{\mathrm{T}} \otimes \mathbf{I}_p$, and from (4.6), we get the structure in Figure 4.2, where

$$\mathbf{y} := [y_1^{\mathrm{T}}, \cdots, y_N^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{pN} \tag{4.19}$$

$$\mathbf{u} := [u_1^{\mathrm{T}}, \cdots, u_N^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{pN} \tag{4.20}$$

This will be the basis for the feedback structure.

[Bai et al., 2011] then suggests employing a feedback term of the form

$$u_i = -\sum_{k=1}^{\ell} d_{ik} \Psi_k(z_k) \tag{4.21}$$

where the z_k's are the difference-variables introduced in (4.1), and $\Psi_k : \mathbb{R}^p \to \mathbb{R}^p$ are nonlinearities to be designed. Note that u_i is implementable with local information, due to the fact that d_{ik} is zero for nodes not communicating on that

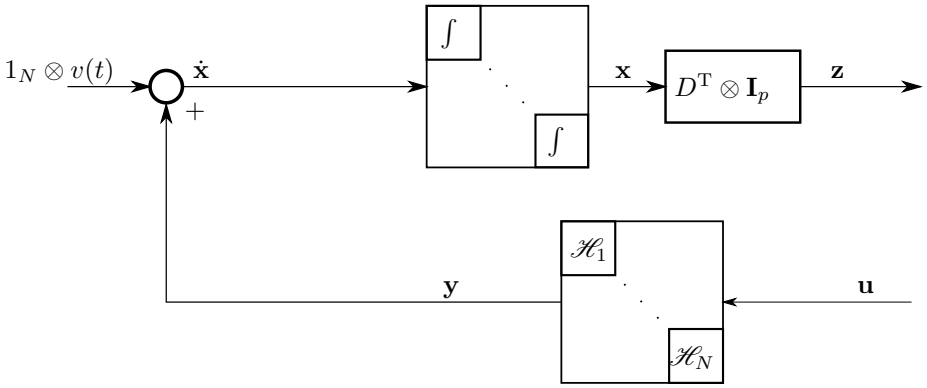


Figure 4.2: A block diagram showing the structure of the multi-robot system before the feedback is introduced.

particular link k. So, intuitively speaking, the feedback-term applies an input u_i as a sum of some function Ψ on the difference-variables z_k for each link.

To illustrate this feedback in the diagram, note that the u_i's in (4.21) can be expressed as

$$u_i = -[d_{i1}\mathbf{I}_p | \cdots | d_{i\ell}\mathbf{I}_p]\boldsymbol{\Psi} \tag{4.22}$$

where

$$\boldsymbol{\Psi} := [\Psi_1^{\mathrm{T}}, \cdots, \Psi_\ell^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{p\ell} \tag{4.23}$$

Then, the concatination of u_i for each i in (4.22) can be written as

$$\mathbf{u} = -(D \otimes \mathbf{I}_p)\boldsymbol{\Psi}(\mathbf{z}) \tag{4.24}$$

By inserting this into Figure 4.2, and rearranging of the blocks containing $D^{\mathrm{T}} \otimes \mathbf{I}_p$ and the integral[1], we get the structure given in Figure 4.3. Please note the appearance of $\dot{\mathbf{z}}$, as (4.6) implies that $\dot{\mathbf{z}} = (D^{\mathrm{T}} \otimes \mathbf{I}_p)\dot{\mathbf{x}}$.

4.2.4 Design criteria for the feedback

The goal for this feedback structure is to render the entire system in Figure 4.3 passive. The feedback-path from $\mathbf{u}$ to $\mathbf{y}$ is passive by design, and the feedforward path from $\dot{\mathbf{x}}$ toj $\mathbf{u}$ can be recognized as a symetric interconnection as in B.1. Thus, Ψ_k has to be designed such that the system is passive from $\dot{\mathbf{z}}$ to $\boldsymbol{\Psi}$.

[1] Remember that both of these blocks are linear, and thus can be re-arranged freely.

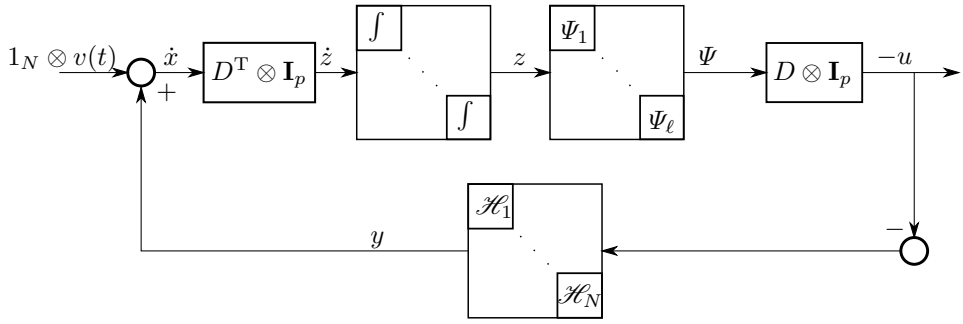


Figure 4.3: A block-diagram of the interconnected multi-robot system with feedback. $\mathscr{H}_i$ is the transformed dynamics from Step 1.

One can accomplish this by designing the nonlinearities to be on the form

$$\Psi_k(z_k) = \nabla P_k(z_k) \tag{4.25}$$

where $P_k(z_k)$ is a non-negative C^2 function. For application to the general problem, [Bai et al., 2011] lists several additional restrictions for P_k and Ψ_k. But, for the formation problem discussed in here, we can design P_k to have the property

$$z_k^{\mathrm{T}} \nabla P_k(z_k) = z_k^{\mathrm{T}} \Psi_k(z_k) > 0, \quad \forall z_k \neq 0 \tag{4.26}$$

which ensures that the other requirements are met.[2] This makes Ψ_k a *monotone* function, belonging to the sector $[0, \infty]$, and thus making it passive from z_k to $\Psi_k(z_k)$.

4.3 Stability results

To investigate the stability of the system, and to assess whether that stability implies that the objectives A1 and A2 in (4.1)-(4.2) are met, the equilibrium points of the interconnected system in Figure 4.3 are first investigated.

As stated in [Khalil, 2002, Chapter 6.5], the closed loop state model will consist of the dynamic states from both subsystems, that is $(\mathbf{z}, \xi)$. First, (4.11) to (4.13) combined with the strictly-passive property, ensures that the equilibrium point $\xi = 0$ only occurs when $\mathbf{u} = 0$, thus making $(D \otimes \mathbf{I}_p)\boldsymbol{\Psi}(\mathbf{z}) = 0$. Further, $\xi = 0 \Rightarrow \mathbf{y} = 0$, which implies that $\dot{x}_i = v(t), \forall i \in \{1, \ldots, N\}$, which again implies by the construction of z_k that $\dot{z}_k = 0$. Last, from (4.5), $z \in \mathscr{R}(D^{\mathrm{T}} \otimes \mathbf{I}_p)$.

[2] As a remark, this criteria also implies that $\Psi_k(0) = 0$.

To sum up, the set of equilibria is given by

$$\mathscr{E} = \left\{(\mathbf{z}, \xi) |\ \xi = 0,\ (D \otimes \mathbf{I}_p)\boldsymbol{\Psi}(\mathbf{z}) = 0 \text{ and } z \in \mathscr{R}(D^{\mathrm{T}} \otimes \mathbf{I}_p)\right\} \tag{4.27}$$

We must now check that these equilibrium points are inside the target sets $\mathscr{A}_k$. $\xi = 0 \Rightarrow \mathbf{y} = 0$ is the goal defined in (4.1), so that is ok. Further, we want the set of equilibrium points of $(D \otimes \mathbf{I}_p)\boldsymbol{\Psi}(\mathbf{z}) = -\mathbf{u} = 0$ imply that $z_k \in \mathscr{A}_k$. It turns out, that for the agreement problem (where the goal is for the agents to *agree* on a value, i.e. $z_k = 0$) this holds when Ψ_k is designed as in (4.26). This is due to the fact that since $\mathbf{z} \in \mathscr{R}(D^{\mathrm{T}} \otimes \mathbf{I}_p)$ and $\boldsymbol{\Psi}(\mathbf{z}) \in \mathscr{N}(D \otimes \mathbf{I}_p)$ imply that $\mathbf{z}$ and $\boldsymbol{\Psi}(\mathbf{z})$ are orthogonal of each other, which by (4.26), is only possible if $\mathbf{z} = 0$.

The passivity and thus asymptotic stability of *agreement problem* is formally proved in [Bai et al., 2011]. In the next section, we will use this result for the formation problem.

4.4 Application to the formation problem

Corollary 2.2 in [Bai et al., 2011] states how the formation problem can be viewed as a shifted agreement problem, where, instead of $\Psi_k(z_k)$, we utilize $\Psi(z_k - z_k^d)$. Remember, z_k^d is the desired value for the link k.

It is further shown that

$$P_k = \frac{\delta_k}{2}|z_k - z_k^d|^2, \quad \delta_k \in \mathbb{R}_{>0} \tag{4.28}$$

which leads to

$$\Psi_k(z_k) = \delta_k(z_k - z_k^d) \tag{4.29}$$

satisfies (4.26) and makes the target set $\mathscr{A}_k$ a.s.

This leads to the final feedback term u_i, and is used for the rest of this thesis:

$$u_i = -\sum_{k=1}^{\ell} d_{ik}\delta_k(z_k - z_k^d) \tag{4.30}$$

4.5 Feasibility of the target sets $\mathscr{A}_k$

For the feasibility criteria (4.7) to hold, certain restrictions must be set to the desired formation structures z_k^d. It can be seen that z_k^d must lie in the range space of $(D^{\mathrm{T}} \otimes \mathbf{I}_p)$ for (4.7) to hold.

This means that given a desired formation $\mathbf{x}^f := [(x_1^f)^{\mathrm{T}}, \cdots, (x_N^f)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{pN}$, where $x_i^f \in \mathbb{R}^p, i = 1, \ldots, N$ is the i'th's agent position in the formation relative some common center, the target sets $\mathscr{A}_k$ can easily be calculated as

$$\mathbf{z}^d = (D^{\mathrm{T}} \otimes \mathbf{I}_p)\mathbf{x}^f \tag{4.31}$$

where $\mathbf{z}^d := [(z_1^d)^{\mathrm{T}}, \cdots, (z_N^d)^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{p\ell}$

For instance, for a simple triangle-formation as in Figure 4.4, the agents should reach the formation specified by (4.32) and Figure 4.4.

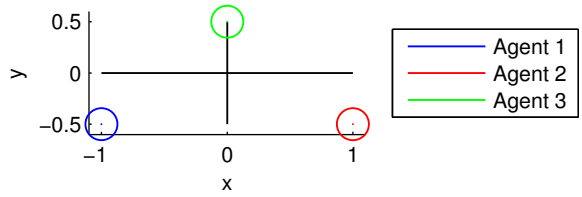


Figure 4.4: The triangle formation used in this example.

$$\begin{aligned} x_1^f &= [-1.0,\ -0.5]^{\mathrm{T}} \\ x_2^f &= [1.0,\ -0.5]^{\mathrm{T}} \\ x_3^f &= [0.0,\ 0.5]^{\mathrm{T}} \end{aligned} \tag{4.32}$$

If the agents are connected as in Figure 4.5, the resulting target sets are as follows:

$$\begin{aligned} \mathscr{A}_1 &= \left\{ z_1 | z_1 = z_1^d = [2,\ 0]^{\mathrm{T}} \right\} \\ \mathscr{A}_2 &= \left\{ z_2 | z_2 = z_2^d = [-1,\ 1]^{\mathrm{T}} \right\} \\ \mathscr{A}_3 &= \left\{ z_3 | z_3 = z_3^d = [-1,\ -1]^{\mathrm{T}} \right\} \end{aligned} \tag{4.33}$$

For ease of notation, $\mathbf{x}^f$ and $\mathbf{z}^d$ may in some places be displayed with their matrix counterpart, X^f and Z^d, where $X^f := [x_1^f | \cdots | x_N^f] \in \mathbb{R}^{p \times N}$, and $Z^d :=$

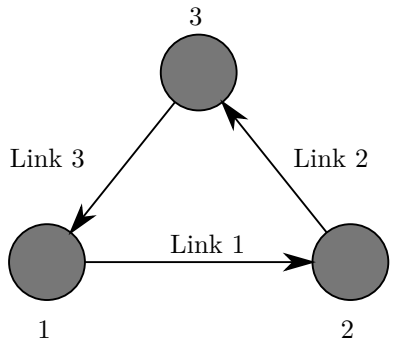


Figure 4.5: Communication graph for the formation example. Note that the graph is connected, and every node has the other two nodes as neighbours.

$[z_1^d|\cdots|z_\ell^d] \in \mathbb{R}^{p\times\ell}$. If not otherwise specified, the target sets $\mathscr{A}_k$ are always constructed from z_k^d as in (4.33).

4.6 Simulations

In this section we will simulate the system with $N = 4$ agents, operating in the plane ($p = 2$). The agents are modelled as the omni-wheels described in Chapter 2, with the differential equation

$$\dot{x}_i = \tau_i \tag{4.34}$$

Step 1 is trivially completed by setting as in 4.18 with $\tau_i\{u_i\} = u_i$. The agents have will have initial positions on the y-axis:

$$\begin{aligned} x_1^0 &= [-0.5,\ 0]^{\mathrm{T}} \\ x_2^0 &= [0.5,\ 0]^{\mathrm{T}} \\ x_3^0 &= [1.5,\ 0]^{\mathrm{T}} \\ x_4^0 &= [-1.5,\ 0]^{\mathrm{T}} \end{aligned} \tag{4.35}$$

To investigate the asymptotic behaviour of the system with different communication schemes, two separate simulations will be run. For both simulations, the target formation is a square with sides equal to one meter:

$$\begin{aligned} x_1^f &= [-0.5,\ -0.5]^{\mathrm{T}} \\ x_2^f &= [0.5,\ -0.5]^{\mathrm{T}} \\ x_3^f &= [0.5,\ 0.5]^{\mathrm{T}} \\ x_4^f &= [-0.5,\ 0.5]^{\mathrm{T}} \end{aligned} \tag{4.36}$$

Also, in both simulations, $\delta_k = 1$ for $k = \{1, \ldots, \ell\}$. Further, the common velocity $v(t)$ of the group, as described in (4.1) is set to zero.

The simulations can be summarized as follows

- Formation simulation 1: Simple connection.
  Here, the agents are connected like a square, where each agent has two neighbours. This gives a total of four links.
- Formation simulation 2: Full connection
  Every agent is now connected to every other agents, giving a total of six links.

The goal of both tests are the stability of the interconnected system, and that the agents reach the target sets.

4.6.1 Formation simulation 1: Simple connection

As can be seen in Figure 4.6, this test uses a simple connection scheme, where every agent has two neighbours.

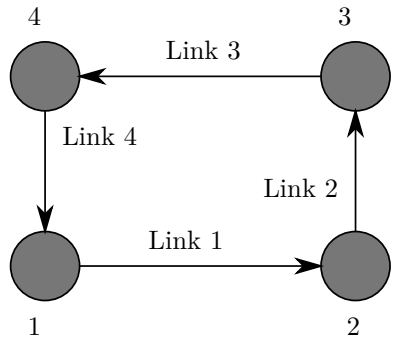


Figure 4.6: Communication graph for Formation Simulation 1. Notice that the graph is connected, and each agent has two neighbours.

Setup

The Incidence matrix D for this test is as in (4.37).

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{4.37}$$

According to (4.31), this leads to the following desired values for the z_k's:

$$\begin{aligned} z_1^d &= [1.0,\ 0.0]^{\mathrm{T}} \\ z_2^d &= [0.0,\ 1.0]^{\mathrm{T}} \\ z_3^d &= [-1.0,\ 0.0]^{\mathrm{T}} \\ z_4^d &= [0.0,\ -1.0]^{\mathrm{T}} \end{aligned} \tag{4.38}$$

Results

As can be seen from Figure 4.7, the trajectories of each agent converge to the desired formation. The agents 1-4 are represented by the colors blue, red, green and purple, respectively. The dashed lines illustrates the trajectory of a particular agent, while the small circle is the final position. The grey box illustrates the links, and is denoted with the final value of z_k. By comparing the final formation with the graph presented in Figure 4.6, we see that the agents have the same positions. This is done to illustrate the links. By comparing the final link values with the desired values given in (4.38), we see that the controller successfully achieves the desired formation.

4.6.2 Formation simulation 2: Full connection

In this simulation, every node is a neighbour of every other node, resulting in the communication topology illustrated in Figure 4.8.

Setup

The communication pattern results in the following incidence matrix:

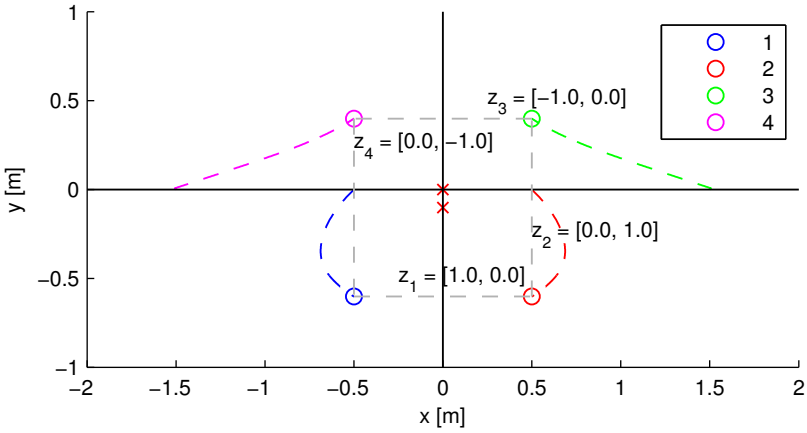


Figure 4.7: Results of Formation Simulation 1. The dashed lines shows the trajectory of each agent, with the circle as the final position. The grey box illustrate the links and the annotations denote the final link values z_k. And finally, the initial and final position of the centroid is marked with a cross.

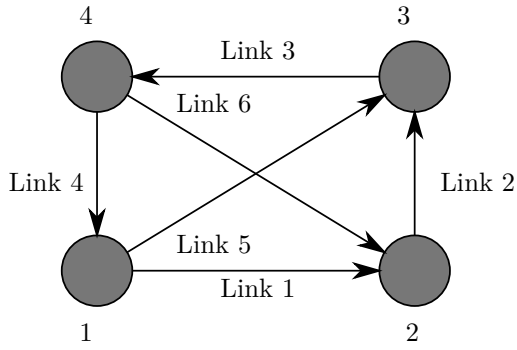


Figure 4.8: Communication graph for Formation Simulation 2. The graph is connected, and every agent is a neighbour to every other agent, thereby "full connection".

$$D = \begin{bmatrix} -1 & 0 & 0 & 1 & -1 & 0 \\ 1 & -1 & 0 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & -1 \end{bmatrix} \tag{4.39}$$

According to (4.31), this leads to the following desired values for the z_k's:

$$\begin{aligned} z_1^d &= [1.0,\ 0.0]^{\mathrm{T}} \\ z_2^d &= [0.0,\ 1.0]^{\mathrm{T}} \\ z_3^d &= [-1.0,\ 0.0]^{\mathrm{T}} \\ z_4^d &= [0.0,\ -1.0]^{\mathrm{T}} \\ z_5^d &= [1.0,\ 1.0]^{\mathrm{T}} \\ z_6^d &= [1.0,\ -1.0]^{\mathrm{T}} \end{aligned} \tag{4.40}$$

Results

Figure 4.9 shows the trajectory of each agent. The agents 1-4 are illustrated by the colors blue, red, green and purple, respectively. The circle represents the final position, while the dashed line is the trajectory of that particular agent. It is seen that the formation is reached. The links are not displayed in this figure, as there is too many of them to be viewed without interfering with the other elements, but the formation is indeed reached. The centroid of the formation is illustrated by a red cross, where the final position is slightly bellow the origo.

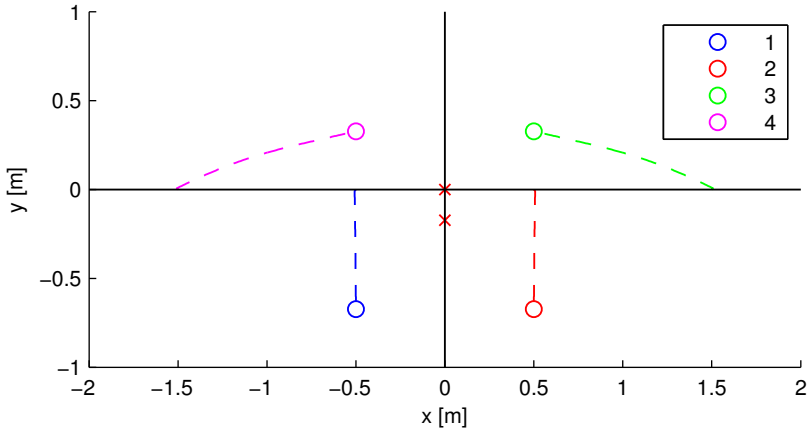


Figure 4.9: Results of Formation simulation 2. The dashed lines shows the trajectory of each agent, with the circle as the final position. And, the initial and final position of the centroid is marked with a cross.

4.7 Formation experiment for four robots

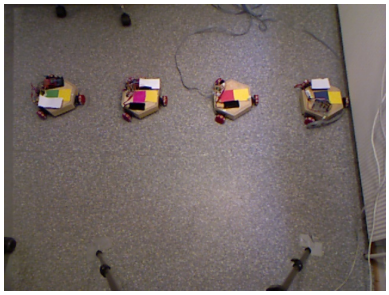
In this section, a test similar to the simulation done in the previous section will be conducted. In this setup, four robots will attempt to stabilize in a square-like formation.

4.7.1 Setup

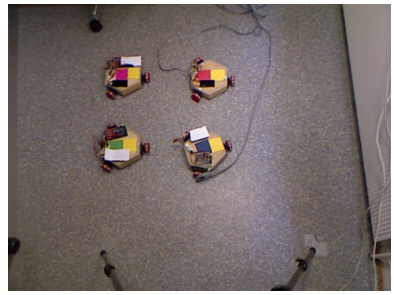
The four robots are connected by a total of 6 links, like illustrated in Figure 4.8. The goal is for the robots to reach a formation described by a square with sides equal to 0.4 m, which according to (4.31), leads to the following desired values for the z_k's:

$$\begin{aligned} z_1^d &= [0.4,\ 0.0]^{\mathrm{T}} \\ z_2^d &= [0.0,\ 0.4]^{\mathrm{T}} \\ z_3^d &= [0.4,\ 0.0]^{\mathrm{T}} \\ z_4^d &= [0.0,\ -0.4]^{\mathrm{T}} \\ z_5^d &= [0.4,\ 0.4]^{\mathrm{T}} \\ z_6^d &= [0.4,\ -0.4]^{\mathrm{T}} \end{aligned} \tag{4.41}$$

The robots are initially placed in a straight line, as pictured in Figure 4.10(a).



(a) Initial positions of the four robots, before test starts.



(b) Final positions of the four robots.

Figure 4.10: Images from the camera, taken before and after the tests. Robots 1-4 are coloured green, magenta, red and blue, respectively.

4.7.2 Results

The results of the test can be seen in Figure 4.11. The dashed lines are each robot' trajectory, while the small circle represents final position. Robots 1-4 are coloured green, magenta, red and blue, respectively. It is seen that the robots reach the desired formation. The reader should note that the centroid of the formation is shifted a little bit to the right, due to drifting, as there have been no effort to control the position if the centroid in this test.

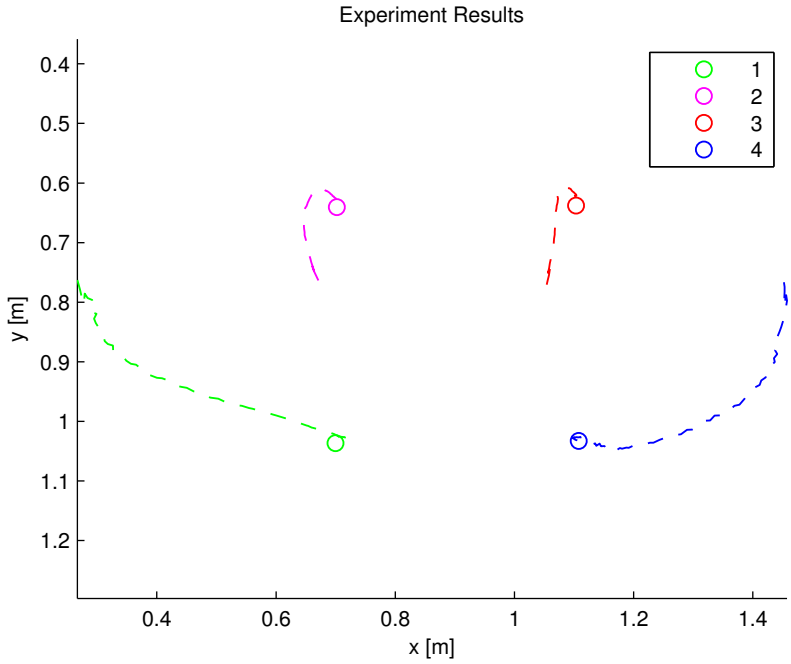


Figure 4.11: Results of the four-agent experiment. The dashed lines shows the trajectory of each agent, with the circle as the final position.

4.7.3 Discussions

When comparing the results of this test to the simulation results in Figure 4.9, the similarities are striking. They both reach the formation in a somewhat similar manner, with minor deviations to the trajectory. Most of this deviations come from tuning and the oversimplistic model used in the simulation.

4.8 Summary

As can be seen, in both of the simulations and the experiment, the agents converge to the prescribed target sets $\mathscr{A}_k$, and the formation is achieved. Both the shape and orientation of the group is stabilized. What is interesting to note, is the difference in trajectories for the two simulations (Figure 4.7 and Figure 4.9). In the second simulation, the agent seems to move more directly to the desired position, while in the first it can be seen that two of the agents moves in an arc towards the goal. This shows the impact the communication pattern has on the system. Consider agent 1, in the lower left corner (blue). For test 1, it has two neighbours, agents 2 and 4. Initially, the target for link 1 is reached, but this is not the case for link 4. Thus, the agent will move to the left. When agent 4 starts to reach its target position relative to agent 3 (green, upper right corner), agent 1 will move back right.

For simulation 2 (and the experiment), agent 1 has three neighbours. z_1 is satisfied, but both z_4 and z_5 has a deviation from the desired value. But since the error in the x-axis is somewhat symmetric for these links, the agent will move directly downwards to satisfy the deviation on the y-axis.

It can be concluded, that by increasing the number of links, the trajectories will generally tend to be more optimal in the sense of distance travelled.

We have also seen that the transition from simulation to implemented controllers on the robot has been successful. The code generated from Simulink Coder presents a unique way to speed up the development cycle, and we verified that it functions properly.

The reader should note that no attempt has been made in this section to control the centroid of the formation. As can be seen in both tests, it moves somewhat downwards. In the next chapter, the formation control techniques derived in this chapter will be combined with the single-agent behavioural control presented in Chapter 3, and the common mission velocity vector $v(t)$ will be utilized to control the centroid position.

Chapter 5

Combining NSB with Cooperative Formation Control

In the previous chapters, cooperative control and behavioural control have been studied. Simulations and experiments have proven their stability properties separately, but it is now time to put them together. Collision avoidance will be added to ensure the integrity of the robots.

In this chapter, two simulations and two tests will be run. In the first setup, two agents will be considered. The agents should reach a prescribed formation, without colliding with each other. This simulation is intended to show the behaviour of the proposed control-scheme.

Finally, a more complex environment is introduced in the second simulation and test. This will consists of four (two for the test) agents, moving in formation in the presence of static obstacles.

5.1 Collision avoidance

The problem of collision avoidance can be solved by imagining each agent as an obstacle. Although the agents are moving and are thus not static, this simplification will work since both agents involved in a collision course will deviate from it's path to avoid collision. This allows the framework described in Chapter 3 to be used with only small changes. Most notably, the agents will transmit their position to

the other agents, and updating their map of the environment. Consider $\mathbf{p}$ as the agent position, and $\mathbf{p}_c$ the position of the closest agent. Then, the task Jacobian is simply

$$\begin{aligned}\mathbf{J}_c &= \frac{\partial}{\partial \mathbf{p}}||\mathbf{p}-\mathbf{p}_c|| \\ &= \frac{(\mathbf{p}-\mathbf{p}_c)^{\mathrm{T}}}{||\mathbf{p}-\mathbf{p}_c||} \in \mathbb{R}^{1\times p}\end{aligned} \tag{5.1}$$

According to (3.5), the task-velocity output $\mathbf{v}_c$ can be calculated as

$$\mathbf{v}_c = \mathbf{J}_c^{\dagger}\lambda_c(d_c - ||\mathbf{p}-\mathbf{p}_c||) \tag{5.2}$$

where λ_c is a suitable positive gain and d_c is the minimum distance between each agent. Further, the task is activated by the same rules as those for obstacle avoidance.

5.2 Cooperative control as a task function

The behavioural control schemes based on NSB introduced earlier gives a quite useful framework for incorporating multiple tasks. That is, the output u_i from the formation control can be considered as a separate task. This task will have the lowest priority, as it is more important to keep the integrity of the agents (collision) than to stay in formation. When considering an environment without obstacles (that is, only collision avoidance and formation control), the controller can be connected as in Figure 5.1. Notice that the internal feedback controller $\mathscr{H}_i$ along with the mission velocity $v(t)$ is added at the end of the chain. For this scheme to work, we must assume that the addition of the mission velocity will not result in a collision. Since the mission velocity affects every agent in the system, this is a reasonable assumption. But, one could image situations where it would happen, for instance if formation was broken by some other factors. When applying the mission velocity, we must then make sure that the formation is kept.

The reader should also note, that for the omniwheels described in this thesis, $\tau_i = u_i + v(t)$, making it possible to use $u_i + v(t)$ as input to the NSB framework directly. This enables the mission velocity to also be checked for possible collisions, and this approach is discussed in Section 5.6.

As stated, the collision avoidance task is only activated when a collision is imminent. In this setup, it translates to when u_i is pointing towards another agent, and the vector is directed towards a point inside a circle with radius d_c around that agent.

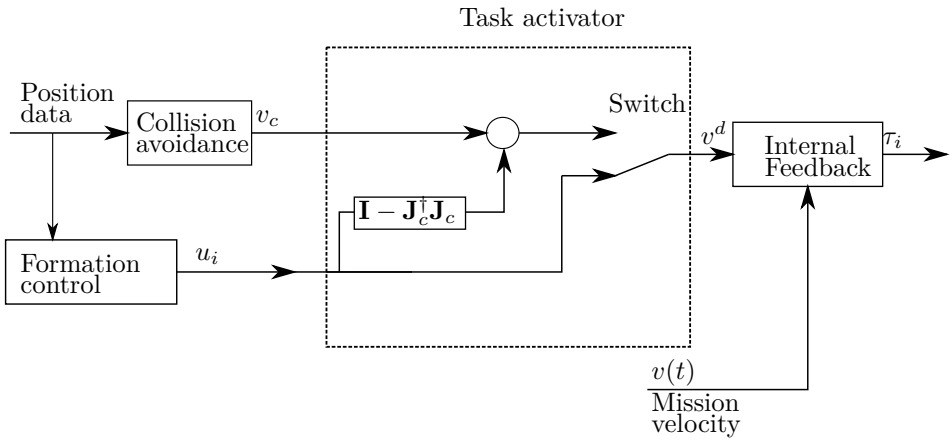


Figure 5.1: Interconnected controller with formation control and obstacle avoidance. The Task activator analyses the input flows, and enables the switch only if a collision is imminent.

5.2.1 Stability analysis

This will be a brief qualitative proof that such an interconnection is stable. Assuming first that the desired formation $\mathbf{x}^f$ does not violate the minimum distance d_c in the collision avoidance control, then it is easily seen that when the collision control is deactivated, we are left with the same structure as earlier. When the collision task is activated, the NSB framework guarantees the stability of each task as long as they don't violate each other. As per our first assumption, this makes the interconnected controller stable.

5.3 Simulation: Two agents

In this simulation, two agents are considered. Their task functions consists of formation control and collision avoidance. The initial positions are given by

$$\begin{aligned} x_{1_0} &= [0,\ 0]^{\mathrm{T}} \\ x_{2_0} &= [1,\ 0.1]^{\mathrm{T}} \end{aligned} \tag{5.3}$$

The agents are connected by a single link, which gives the obvious $D = [-1,\ 1]^{\mathrm{T}}$. Further, the formation is specified by

$$\begin{aligned} x_1^f &= [1,\ 0]^{\mathrm{T}} \\ x_2^f &= [0,\ 0]^{\mathrm{T}} \end{aligned} \tag{5.4}$$

That is, agent 1 should be at a distance of 1 along the x-axis from agent 2. The behavioural control parameters for collision avoidance are $d_c = 0.2$ and $\lambda_c = 1$. Finally, the agents are modelled by the pure kinematic dynamics described in Chapter 2.

In this simulation, the only objective is to reach the desired information without collisions, and thus the mission velocity $v(t) \equiv 0$. As always, the agents are assumed to have knowledge of it's neighbour position.

5.3.1 Results

The final trajectory of the two agents can be seen in Figure 5.2. Agent 1 is coloured blue, while agent 2 is red. The dashed lines is the trajectory, while the circles represent final position. Figure 5.3 shows a complete time-series of the simulation, with several situations included. Note in Figure 5.3(c), how the formation control output u would violate the collision constraint, and thus making the collision avoidance task active. This creates an output v_d which continuously moves the agent along the circle arc.

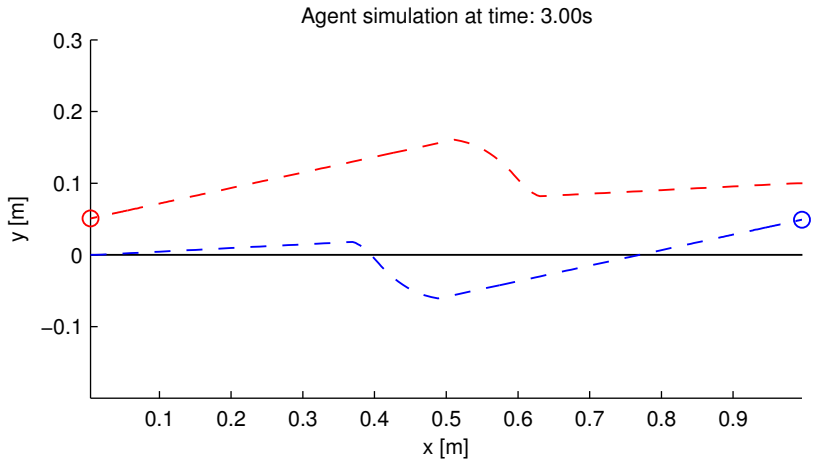
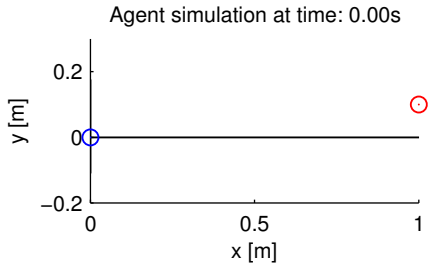
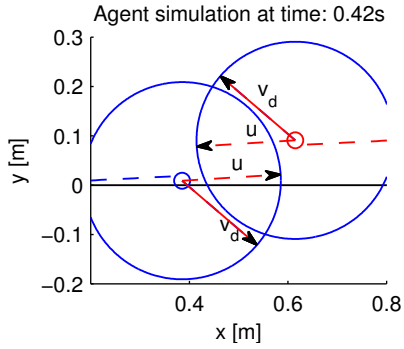


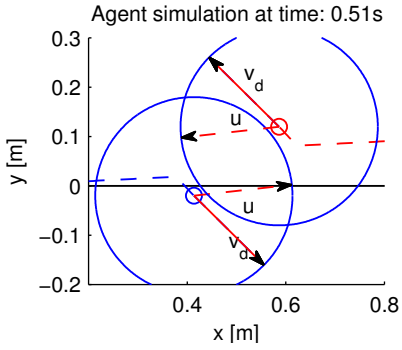
Figure 5.2: The trajectories and final position of the two agents in the simulation. Agent 1 is coloured blue, while agent 2 is red.



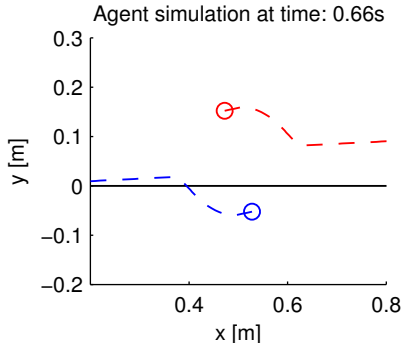
(a) Initial position of agent 1 (blue) and agent 2 (red).



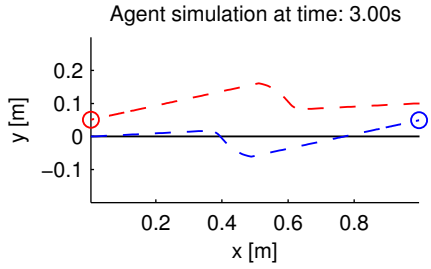
(b) Point of first intersection, where the collision avoidance task is activated.



(c) Here, the agents are moving away from each other, to avoid collision.



(d) The agents are no longer on collision course, and the collision avoidance task is deactivated.



(e) Final position.

Figure 5.3: A time-series of the simulation. The small circles are agents' 1 (blue) and 2 (red) position, while the dashed lines shows its trajectory. The larger blue circle around each agent is the minimum distance that must be kept. The nomenclature used on the velocity vectors are those presented in Figure 5.1.

5.4 Experiment: Two agents with collision avoidance

The simulation done in the previous section is now repeated on the robots. We consider two agents, whose objective is to reach a predefined formation without collisions. The controller structure is identical to the simulation, with the two tasks being collision avoidance and formation control. Collision avoidance is the highest priority task. In this test, the radii for collision avoidance is $d_c = 0.3$, as the robots are approximately 30 cm wide. As before, the two agents are connected by a single link, resulting in $D = [-1,\ 1]^{\mathrm{T}}$.

In this test, the robots should form a line with 80 cm between the two robots on the x-axis:

$$\begin{aligned} x_1^f &= [-0.8,\ 0]^{\mathrm{T}} m \\ x_2^f &= [0,\ 0]^{\mathrm{T}} m \end{aligned} \tag{5.5}$$

5.4.1 Results

The trajectory and final position of the two robots can be seen in Figure 5.4. The formation is reached, and by studying Figure 5.5, it is seen that the robots avoided the imminent collision. Specifically, at Figure 5.5(c), we see that the collision is detected and the controller steers the robot away. The annotation $\mathrm{v_d}$ represents the resulting desired direction while $\mathrm{v_t}$ is the output from the formation behaviour.

5.5 Discussions

As can be seen, by using the control scheme introduced in this chapter, the agents reach the desired formation without collisions in both simulations and real-life tests on the robots. The results also show a high degree of similarity in results from the simulations, compared with the tests. Although it was a simple example, it shows the power and flexibility of the NSB framework, and how the cooperative control schemes can be extended to provide useful functionality. The reader should of course note that the agent in this thesis have very simple dynamics. Although both frameworks work well with more complex models [1], some more care has to be taken on tuning of the parameters. The kinematic output could also be used

[1] For instance, [Bai et al., 2011, Chapter 6] uses the passivity based cooperative control schemes on Lagrangian systems, and the NSB framework can be used on the output velocity as before.

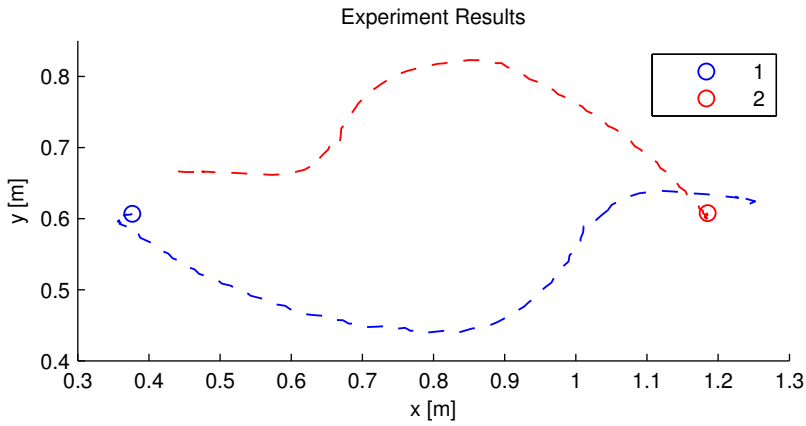


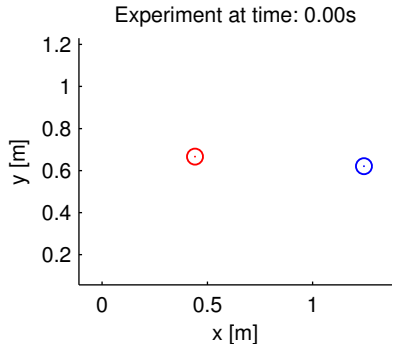
Figure 5.4: Test results. Agent 1 is coloured blue, while agent 2 is red. The formation is reached.

to create reference trajectories for low-level controller to a more complex system. By limiting the controller to the case of pure kinematics, a direct approach to the addition of the obstacle-avoidance problem can be done, and this is the focus of the next sections.

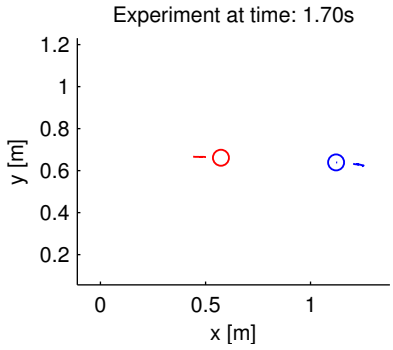
5.6 Adding obstacle avoidance: Direct approach

In the previous section, the behavioural control for collision avoidance was added to the coordinated agents, to provide a method , to keep the integrity of the robots. Further, we now wish to add obstacle avoidance to the controller. The direct approach here, would be to continue to build up the NSB chain, as seen in Figure 3.3. The problem with this approach is that if the internal controller is created at the end of the chain, as in Section 5.2, the mission velocity would not be included in the NSB framework. This is a reasonable approach when considering only collision avoidance, as the mission velocity affects all agents, but can't ordinarily be done with obstacle avoidance. The mission velocity must also be checked for possible obstacle collisions!

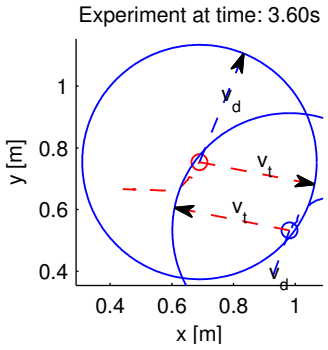
When considering the omniwheels discussed in this thesis, it was seen in Chapter 2, that the agents have a pure kinematic dynamics. And, it was seen in Section 4.2.2, the internal low-level controller was a simple addition of the formation output and mission velocity. That is, $\tau_i = u_i + v(t)$. In this special case, τ_i is simply the velocity reference. This enables us to use feedback τ_i directly as input to the behavioural control part of the controller, rather than at the end as illustrated in Figure 5.1.



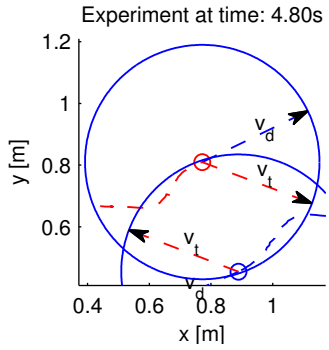
(a) Initial position of robot 1 (blue) and robot 2 (red).



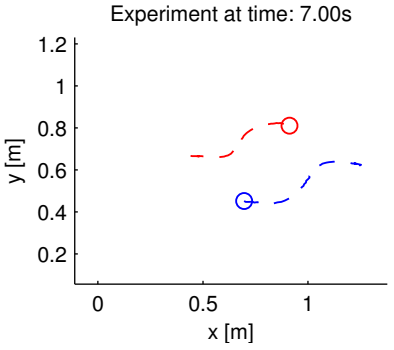
(b) The robots are moving towards each other to reach the prescribed formation.



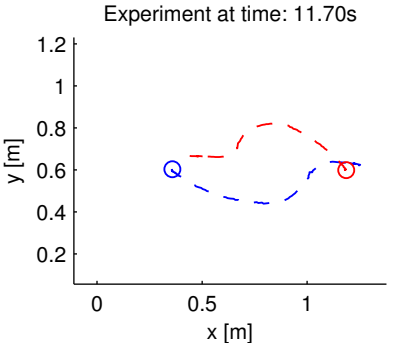
(c) A possible collision was detected, and both robots starts to circumvent the other robot.



(d) The robots nearly have a clear path.



(e) The collision is avoided, and the robots move towards their goal.



(f) Final positions, the formation is reached.

Figure 5.5: Timeseries of the test. The robots 1 and two are coloured blue and red, respectively. The dashed lines shows the trajectory, while the small circles represents current position. The larger blue circle in **(c)** and **(d)** represents the minimum safe-distance of each agent.

This creates the structure seen in Figure 5.6, valid for omnirobots discussed and tested here.

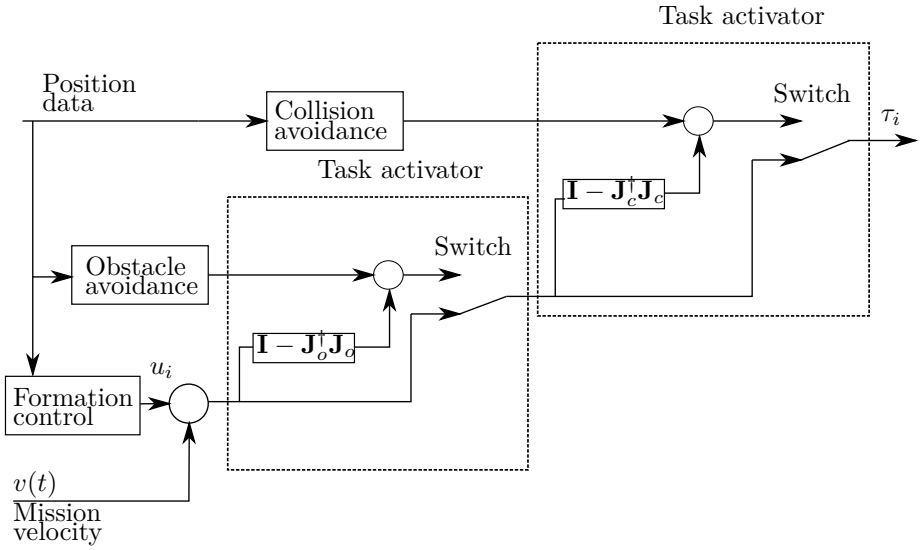


Figure 5.6: Specialized kinematic controller for the kinematic omniwheels. In this setup, each agent has it's own obstacle avoidance task. The rules for activation are the same as discussed in Chapter 3.

5.7 Simulation with obstacle and collision avoidance

In this simulation, four planar agents will be considered. The agents should avoid colliding with each other, as in the simulation in the previous section. But in addition, the agents should avoid static obstacles while at the same time following a mission velocity $v(t)$.

But, care must be taken when including a mission velocity $v(t)$ since such a velocity might violate obstacle constraints. Thus, the added mission velocity must also be checked for possible obstacle collisions. This is done by the controller introduced in the previous section, by Figure 5.6.

The mission velocity is created to make agent 1 tend to the position $(x, y) = [10,\ 0]$. This is done by adding a separate controller for agent 1, whose output is distributed to the other agents. The controller is similar to the one introduced in Section 3.5.2, but is repeated here for completeness:

$$v(t) = \lambda_t(x_1 - p_t) \tag{5.6}$$

where $x_1 \in \mathbb{R}^p$ is agent 1' position, $p_t \in \mathbb{R}^p$ is the target location, and $\lambda_t \in \mathbb{R}_{>0}$ is a positive gain.

As discussed in Chapter 2, the robots cannot move faster than $1m/s$. This restriction is present in the simulated model, but should be included in the controller as well. Specifically, the output of the goto-target task is saturated at $1m/s$. See also the discussions in Section 3.5.3.

Further, the simulation consists of two separate states. The first state, Initialization, is where the agents reach the desired formation. In this state, the mission velocity is $v(t) = 0$. When the formation is reached, the system is in Move state. Here, the mission velocity in (5.6) is activated. The state diagram is shown in Figure 5.7.

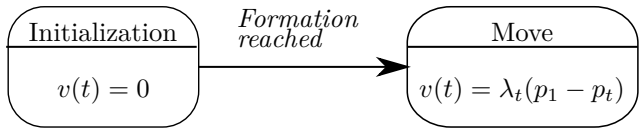


Figure 5.7: State diagram of the four-agent simulation. In the Initialization state, the agents move to the desired formation. When it is reached, the mission velocity is activated.

The agents initial positions are given by:

$$\begin{aligned} x_{1_0} &= [2,\ 2]^{\mathrm{T}} \\ x_{2_0} &= [0,\ 0.5]^{\mathrm{T}} \\ x_{3_0} &= [1,\ 1]^{\mathrm{T}} \\ x_{4_0} &= [-1,\ -1]^{\mathrm{T}} \end{aligned} \tag{5.7}$$

and the desired formation is a square with sides equal to 2:

$$X^f = \begin{bmatrix} 0 & 2 & 0 & 2 \\ 0 & 2 & 2 & 0 \end{bmatrix} \tag{5.8}$$

The gain λ_t in (5.6) is chosen as $\lambda_t = 1$. Further, $\lambda_c = 4$, $\lambda_o = 2$. The minimum agent distance is $d_c = 0.4$ as before, and the obstacle radius are $d_o = 0.5$.

Two static obstacles are present, their locations are $[7,\ 2.3]^{\mathrm{T}}$ and $[6,\ -0.1]^{\mathrm{T}}$.

5.7.1 Results

The final position and trajectory of the four agents can be seen in Figure 5.8. Further, Figure 5.9 shows the time-series of the simulation, captured at certain interesting situations. In both figures, the agents 1-4 are coloured as blue, red, green and purple, respectively. In Figure 5.9(b), we see that agent 1 (blue) and agent 2 (red) must deviate from their current path to avoid a collision. Further, the figures clearly shows that the agents reach the desired formation (Figure 5.9(c)), and stays close to this formation even when some of the agents have to move around an obstacle (Figure 5.9(e)). In the last frame, Figure 5.9(f), we see that the agents have reached the final position, and that the formation is kept.

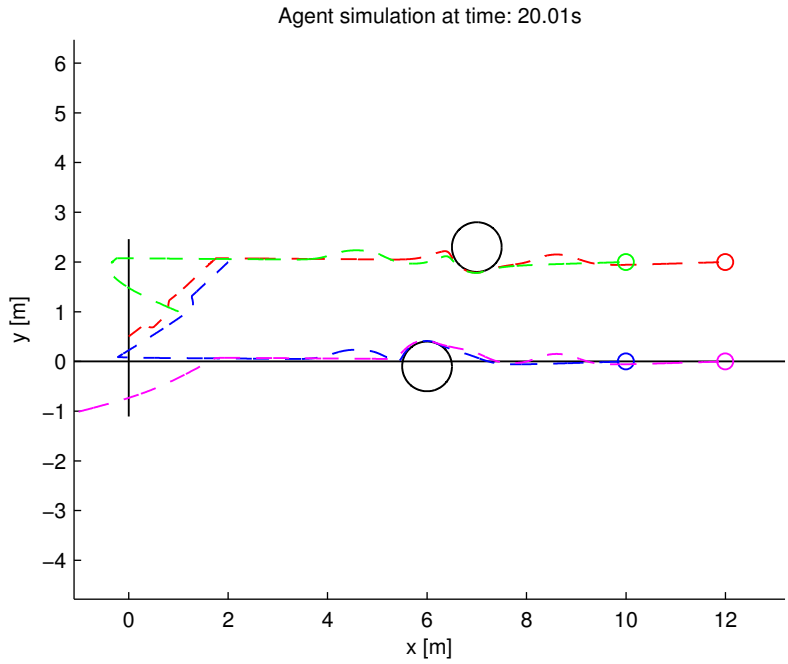
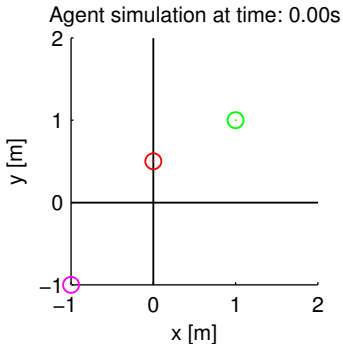


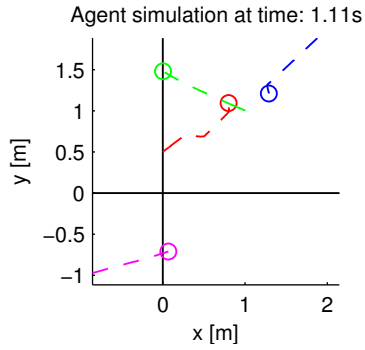
Figure 5.8: The final position and trajectory of the simulation.

5.7.2 Discussion

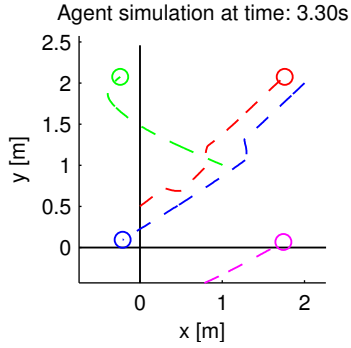
As the results show, the proposed kinematic controller makes the agents have the desired behaviour. They maintain the formation, while at the same time voiding collisions and obstacles. But the weaknesses of this direct approach must be



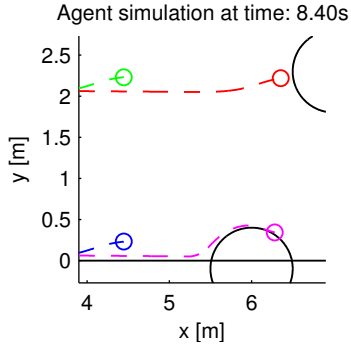
(a) Initial positions of the four agents.



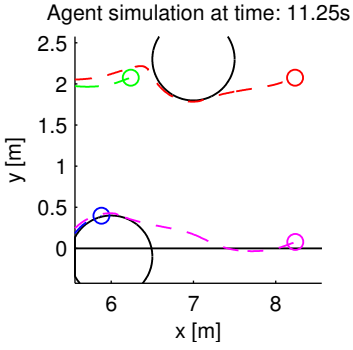
(b) As the agents move to the desired formation, agent 1(blue) and 2(red) must deviate on their course, to avoid a collision.



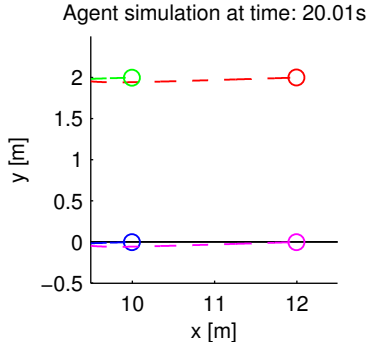
(c) The agents have reached the desired formation.



(d) The agents encounter an obstacle, and avoids it.



(e) Agent 1 and 3 also reaches the obstacles.



(f) Agent 1 reaches its desired position p_t.

Figure 5.9: Timeseries of the simulation. Agents 1-4 are coloured blue, red, green and purple, respectively. The dashed lines shows the trajectory, while the small circles represents current position.

addressed. Mainly, to be able to use obstacle avoidance as described in 5.6, the velocities must be directly manipulable by $\tau_i = u_i + v(t)$. This will generally not be the case. Further, there is a noticeable "bump" in agent 1's trajectory in Figure 5.8. This happens because the other agents are moving to avoid the obstacle, and thus agent 1 moves to adjust its position in the formation. The same effects can be observed for all four agents.

When dealing with agents with more complex dynamics, it could be desired to let the group of agents move as a single unit rather than avoiding the obstacles individually. This is further discussed in Chapter 6.

5.8 Experiment: Two robots with obstacle- and collision avoidance

The test from the above section is now repeated on the actual robots. In this test, two agents are considered, as opposed to four in the previous section (the main reason for this is the lack of space in the lab). The robots are equipped with the same interconnected controller as introduced in Figure 5.6. Since there now are only two agents, the desired formation is a vertical line:

$$X^f = \begin{bmatrix} 0 & 0 \\ 0 & 0.4 \end{bmatrix} \text{ m} \tag{5.9}$$

that is, robot 2 shall position it selves 40 cm above robot 1. There is one static obstacle present in the environment, who's position is tracked by the camera and sent to the robots. As before, the robots will first gather to the correct formation in the first state, and then move to the target location. The target location is also tracked by the camera. Figure 5.10 displays the robots (robot 1 has the green marker, robot 2 the magenta), the obstacle (red) and target location (blue).

As in the previous section, the mission velocity is created by robot 1, and distributed to the other agent(s).

5.8.1 Results

A timeseries of the results of the test can be seen in Figure 5.11. Figure 5.11(a) shows the initial position of the robots. The blue and red circle represents robot 1 and 2, while the cyan marker is the target robot 1 must reach. The black circle is the static obstacle that must be avoided. In Figure 5.11(b), the robots have reached the formation and robot 1 starts to move towards the target. It's mission

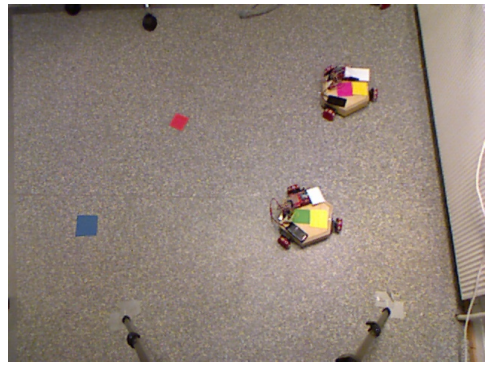
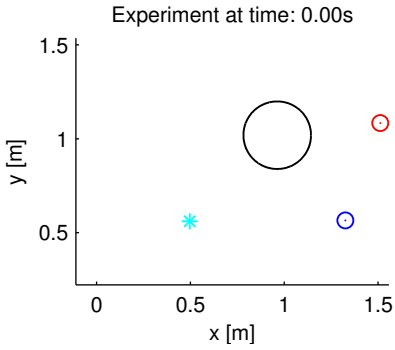


Figure 5.10: Image of the robot' position before the test starts. The two robots are pictured together with the target (blue) and the obstacle (red).

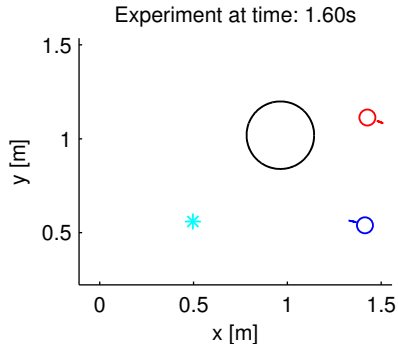
velocity is sent to robot 2. Further, it is seen that robot 2 avoids the obstacle, and the group stabilizes at its final position in Figure 5.11(f).

5.8.2 Discussions

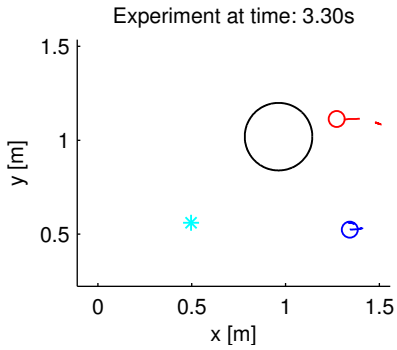
The results show that the proposed controller meets its design objective. The group of robots reach the desired formation, and the group moves towards the target position. It is thus seen that the direct approach of adding obstacle avoidance is feasible for the robots discussed in this thesis. The test also proved the capabilities of the hardware platform, as it was possible for agent 1 to distribute it's mission velocity to the other agents in real time.



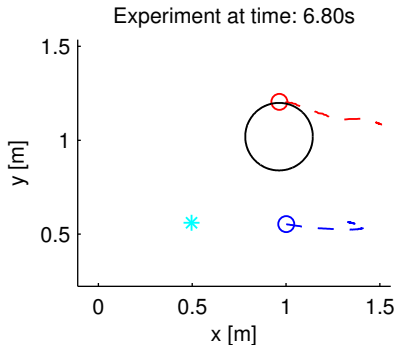
(a) Initial position of the robots.



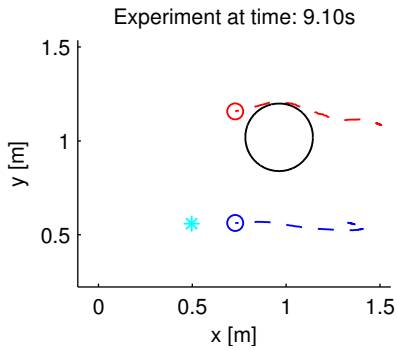
(b) The desired formation is reached.



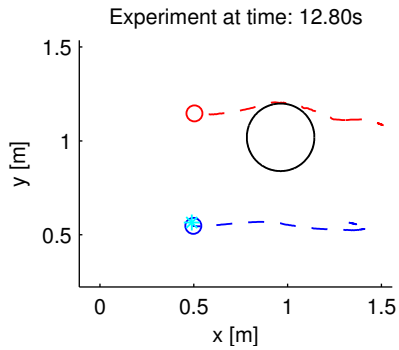
(c) Robot 2 is approaching the obstacle, and activates its obstacle avoidance task.



(d) Robot 2 has successfully circumvented the obstacle.



(e) Both robots starts to approach the final position.



(f) Final position for both robots. Target location is reached for robot 1, the formation is still kept.

Figure 5.11: Timeseries of the test results. Agents 1-2 are coloured blue and red, respectively. The dashed lines shows the trajectory, while the small circles represents current position. The cyan marker is the target location, while the black circle is the obstacle to be avoided.

5.9 Summary

In this chapter, it was seen that the output from the formation controller could be used as a task-function in the NSB framework. This made it easy to combine several group functions, and made it possible for the agents to avoid collisions and obstacles while still keeping formation. For this purpose, two different controllers was presented. The first combined the formation control problem with collision avoidance. This controller was tested in both a simulation and in a real-world test, and it was seen that this controller could be used for agents with more complex dynamics than the kinematic robots considered in this thesis. Further, a pure kinematic controller that included obstacle avoidance was presented. It allowed the agents to simultaneously avoid both obstacles and collisions, at the cost of a less generic controller. But, the output of this kinematic controller could be used to generate reference signals to be used by e.g. a guidance controller.

The controllers were verified by both simulations and tests on robots.

In the next chapter, a different approach to the problem is presented. Here, the goal is to let the group of agents move as a single unit around obstacles, rather and avoiding them individually.

Chapter 6

Search and Rescue

In this chapter, a search-and-rescue mission will be simulated. The mission consists of five agents, which should be able to hold several different formations, dependent on the current mission state. The position of the object of interest is for the purpose of this simulation known in advance, but the simulation will contain the same elements as a more realistic search situation. Further, the concept of an agent leader is introduced.

6.1 Mission summary

The mission consists of five stages, each has to be completed before the next one. A simple state-space diagram is illustrated in Figure 6.1. Bellow follows a sequential description of each state.

1) Initialization This is the first state. Here, the agents are scattered after deployment, and needs to reach a formation designed to minimize drag in the next state. The desired formation is a v-shape, with the leader agent in the middle.

2) Move After the formation is reached, the agents should move as a single group towards the search area. Along the way, there will be static obstacles present. The agents should avoid these obstacles as a single group, that is, they should not break formation. Rather, the entire group should move together, to ensure that no collisions with obstacles occur.

3) Init search When the search location is reached, the agents should scatter to a search formation. This is a single vertical line, with a distance of 1 m

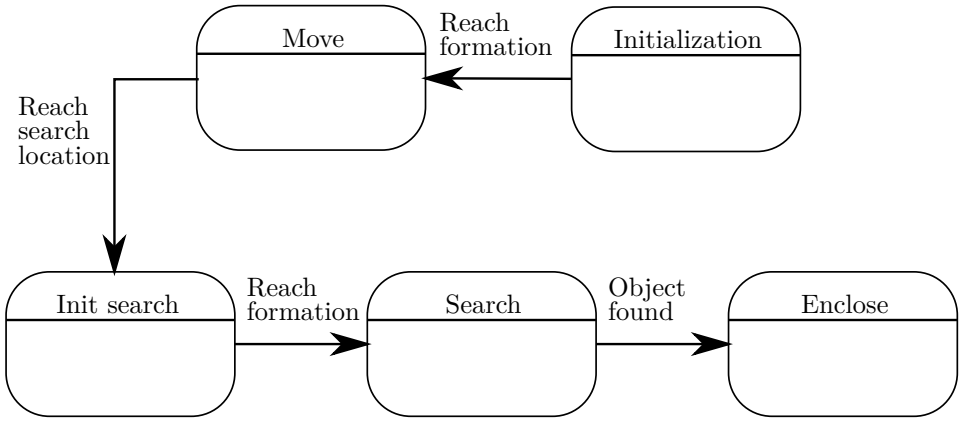


Figure 6.1: A state-space diagram of the Search and Rescue mission.

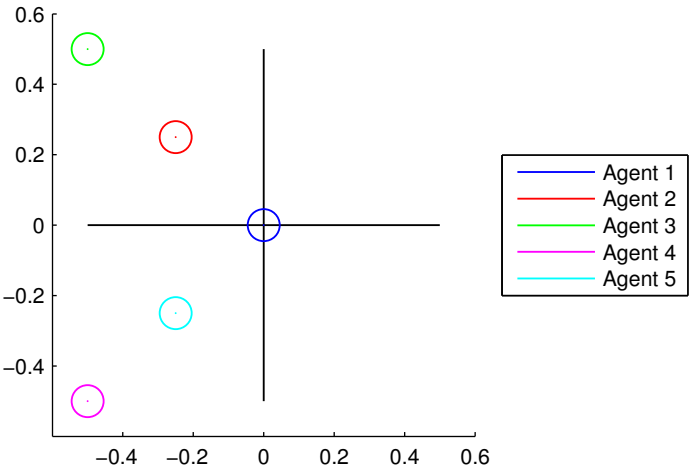
between each agent.

4) Search When the search formation is achieved, the agents should move along a horizontal line, illustrating a search for a target. In this mission, the target is static, and is considered found when agent 1 reaches its position.

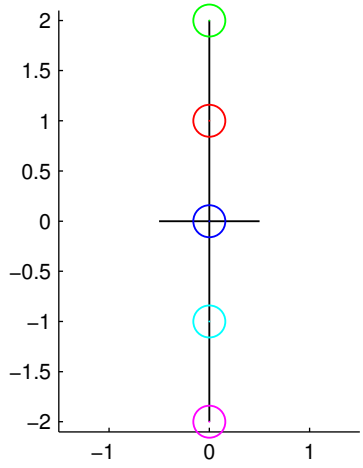
5) Enclose After the target has been found, the agents should enclose the target with a circle-like formation.

The reader should note that the handling of technical problems or re-tries of search procedures are considered out of scope of this thesis.

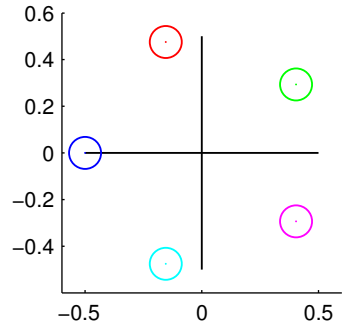
The formations for the states Move, Search and Enclose are given in Figure 6.2(a) - Figure 6.2(c), respectively.



(a) Formation used when moving to the search area (state 1 and 2). It is designed to minimize drag.



(b) Formation used when searching for the target (state 3 and 4). It is designed to cover as much ground as possible.



(c) When the target of interest is found (state 5), the agents should enclose it in a circle-formation.

Figure 6.2: Formation specifications for the Search and Rescue mission. The colour-labels displayed in Figure 6.2(a) is valid for all three formations.

6.2 Agent leader

For this mission, agent 1 will presume the role of a group leader. Mainly, this means that $u_1 \equiv 0$ forcing the formations to be reached around the leaders current position. Further, all mission velocities is created by agent 1, and is distributed to the other agents. It is assumed that every other agent is connected to the leader.

The mission states that the agents should avoid collisions with each other, as well as being able to move around static obstacles as a single group. Also, we want to avoid the specialized controller structure presented in Section 5.6, as we want this procedure to be valid for agents with more complex dynamics. The solution will be to let the mission velocity of agent 1 to be created by a separate behavioural control, specifically designed to let the group avoid the obstacles.

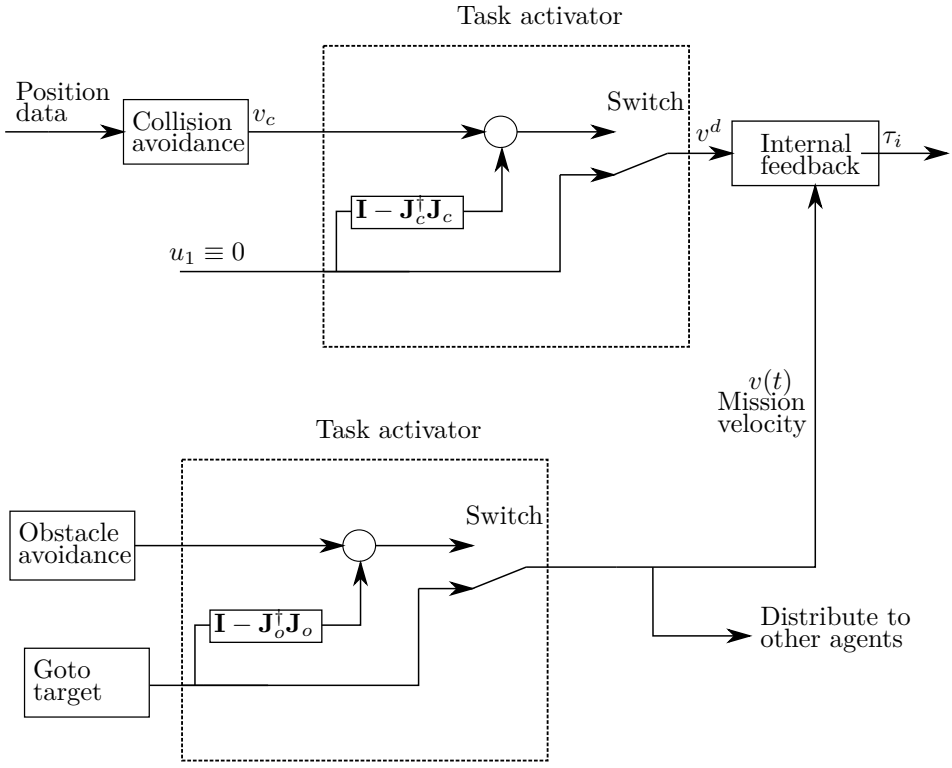


Figure 6.3: Illustration of the leader' control system. It consists of two separate NSB blocks, whose output is merged by the passivity based framework. The upper part of the diagram is shared with the other agents, except that instead of $u_1 \equiv 0$, the other agents receive their u_i from the formation control, as seen in Figure 5.1.

The complete controller is presented in Figure 6.3. It is seen that two separate NSB-blocks are present. The top contains the collision avoidance and formation control tasks. But as this controller is intended for the leader, $u_1 \equiv 0$ since we want the other agents to form the formation around this leader agent. The bottom part of the controller generates the mission velocity $v(t)$. It contains two tasks, collision avoidance and a task function to go to a target location. These tasks are the same as those presented in Section 3.5, and the obstacle avoidance task is activated by the rules discussed in Section 3.5.3. The resulting mission velocity $v(t)$ is used by the low-level controller, and is distributed to the other agents.

The obstacle avoidance task works by calculating the distance from the robot to the obstacle. Since the leader now should keep the entire group away from the obstacle, this distance should be calculated from the barycentre of the robots. This can simply be calculated by

$$x_{bary} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{6.1}$$

where $x_i \in \mathbb{R}^p$ is agent i's position. Also, the size of the current formation has to be taken into account. By defining $d_{farthest}$ as the distance from the barycentre to the agent farthest away (in the current formation), the obstacle-avoidance task can be altered from (3.16) to this:

$$\mathbf{v}_o = \mathbf{J}_o^{\dagger}\lambda_o(d_o + d_{farthest} - ||x_{bary} - x_o||) \tag{6.2}$$

where $x_o \in \mathbb{R}^p$ is the closest obstacle.

6.3 Simulation setup

The initial positions of the agents are given by

$$X_0 = \begin{bmatrix} 1 & 0 & 1 & -1 & -1 \\ \frac{1}{2} & \frac{1}{2} & 2 & -1 & 0 \end{bmatrix} \tag{6.3}$$

where $X_0 = [x_1^{\mathrm{T}}, \ldots, x_5^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{p\times N}$. These initial positions are somewhat random, and should illustrate that the agents may be deployed by different means. The group are set to be fully connected, that is, every agent is a neighbour to every other agent. This is not a necessary assumption for the controller to work as expected, but as seen in Section 4.6, it may improve agent' trajectories.

The search is initiated when the leader reaches the search area, which is given by the position $(x, y) = [15,\ 0]^{\mathrm{T}}$. The target is located at $(x, y) = [18,\ 0]^{\mathrm{T}}$. The gains for formation control and collision avoidance are $\delta_k = 1\ \forall k$ and $d_c = 4$, respectively. The minimum inter-agent distance are $d_c = 0.2$.

Further, the control parameter for the leader's mission velocity creation, the gain Λ_t to control the convergence of the goto-target task from Section 3.5.2 was chosen as $\Lambda_t = \texttt{diag}\{2, 2\}$. The saturation of the velocity output of this task was lowered to $0.7m/s$, to allow agents to "catch up" if they fell behind. As for the obstacle avoidance task, each obstacle has a diameter of 0.5, giving $d_o = 0.5$.

There are two static obstacles in the area. They are at located at $[6,\ 1]^{\mathrm{T}}$ and $[10,\ -0.5]^{\mathrm{T}}$.

6.4 Results

Figure 6.4 shows the simulation results, and displays the trajectories and final position of each agent. Agents 1-5 are displayed with the colors blue, red, green, purple and light-blue, respectively. The dashed line shows the trajectory up until the current position, which is displayed by a circle. As can be seen, the group avoids the obstacles, and surrounds the target. Figure 6.6 shows the time-series of the simulation, highlighting certain interesting situations. Note for instance in Figure 6.6(c) how the whole group avoids the obstacle, while keeping formation. Further, the change of formation in Figure 6.6(e) is successful.

Figure 6.5 shows the velocity of the leader over the simulation time. There are five distinct peak-values in the x-direction velocity. Firstly, it is seen that the maximum velocity is saturated by the controller output to $0.7m/s$ as discussed earlier. The first peak occurs after the initialization state is complete; here the formation is achieved. The agents then tends in the positive x-direction. The two following peaks represents the times when the group encountered an obstacle. The last positive peak occur when the agents are performing the search. The negative spike occurs when the leader has found the target, and moves slightly backwards. For velocity information for all five agents, see Figure C.2.

6.5 Discussion

The goal of the simulation was to see if the agents could maneuver in a formation to the search area, find a target, and enclose it with a new formation. All while avoiding collisions with other agents and obstacles. As can be seen form Figure 6.4, the mission was successfully carried out.

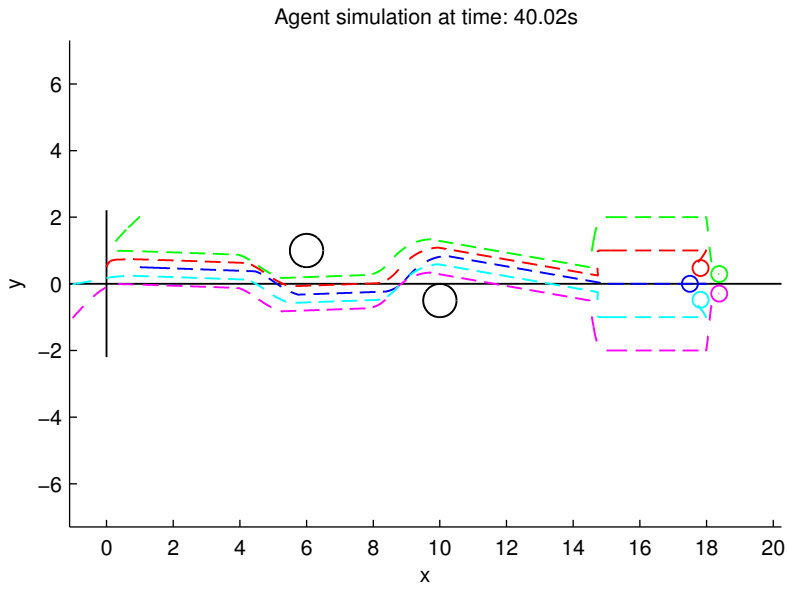


Figure 6.4: The trajectories of the agents in the mission simulation. The dashed lines shows the trajectories of Agents 1-5, colored as blue, red, green, purple and light-blue, respectively. The obstacles are marked as solid black circles.

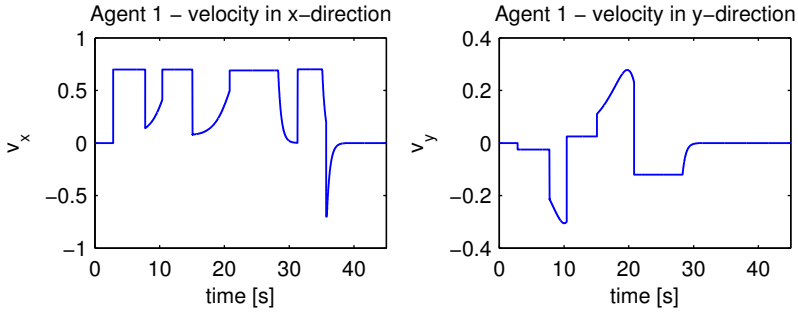
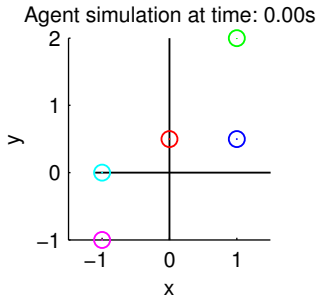
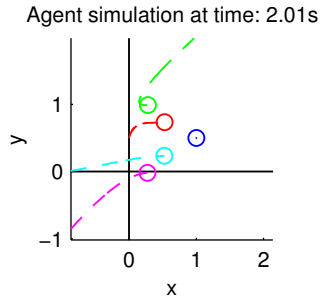


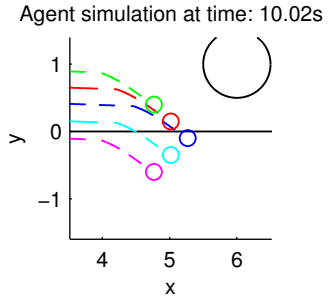
Figure 6.5: The velocity of the leader agent in x- and y-direction, respectively.



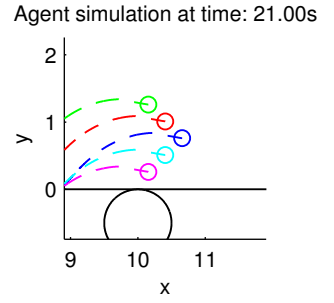
(a) Initial positions of all five agents.



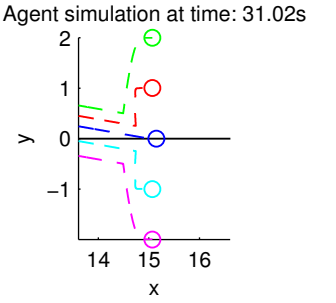
(b) The minimum-drag formation is reached, and the system enters the Move-state.



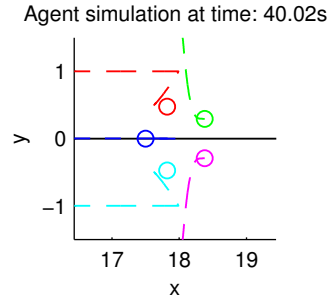
(c) The leader distributes mission velocities to make the group avoid the obstacle.



(d) The second obstacle is also avoided.



(e) The group has reached the search area, and have now formed a search-formation.



(f) The target was found, and the group surrounds it with a circle-like formation.

Figure 6.6: Timeseries of the mission simulation. The agents 1-5 are coloured blue, red, green, purple and light-blue, respectively. The circle represent position at a given time, while the dashed lines indicate the trajectory.

The behavioural control part of the leader controller activates the obstacle avoidance task when the group comes close to the obstacles, and they are on collision course.

It should be noted that the formation remains constant, even though they are moving around the obstacles. As the goal was to minimize drag, it would desired to rotate the formation so it points in the forward direction. This is not considered further in this thesis, but would be an important aspect when this is to be implemented on the robots.

The simulation results shows the power of the passivity framework. It allowed the adoption of complex missions to be carried out, and it had the flexibility to introduce the concept of a group leader. And although the controller mainly works on the kinematic level, the framework easily allows more complex dynamics to be introduced. This has to do with the requirements for the strictly passive internal feedback that is needed. As long as the controller can be made, the framework guaranties asymptotic stability of the formation problem.

In this simulation, it was assumed that every agent was a neighbour to every other agent. As stated earlier, the only requirement is that the connection graph is connected, and that every agent had access to the mission velocity $v(t)$.

The number of agents in this simulation, five, is set somewhat at random. The framework could easily be used on a larger number of agents. Also, by creating sub-nodes in the system, missions could be carried out with several squadrons, each containing a subset of agents, which worked together. It should be noted that the number of calculations needed for the formation problem is proportional to each agents number of neighbours.

Chapter 7

Conclusion and Closing Discussions

The goal of this thesis was to investigate the use of a cooperative control system to control the behaviour and formation of a multi-robot system. By carefully investigating and discussing each part of the controller separately, it was put together to form a complete controller capable of making the agents avoid collisions with each other and obstacles, while moving in formations. The results was verified by simulations and experiments, thus combining advanced control theory with practice.

First, the mathematical model for the robots was developed. The robots are planar vehicles, designed to have holonomic dynamics. It was showed that the robot dynamics could be described by pure kinematic equations, and that further allowed the controllers to be tested at a kinematic level.

The methods for behavioural control was introduced, along with the Null-Space based Behavioural control (NSB) framework. This framework allowed predefined tasks to be assigned a priority, and guaranteed stability of the overall goal as long as lower priority tasks didn't conflict with each other. For the purpose of this thesis, two important task functions were chosen. These described the goals for collision and obstacle avoidance, and how to move a robot towards a target location. Simulations and experiments were conducted to illustrate the properties of the framework.

The formation problem was solved by using a passivity-based solution to the agreement problem. This allowed the creation of a decentralized cooperative controller, capable of bringing the group of robots (agents) to a specified relative formation.

By creating a strictly passive internal feedback for the system, the passivity-based approach allows the control of complex dynamics. This include both Newtonian and Lagrangian systems. The impact of the communication topology was investigated by performing two simulations, where it was shown that the agents had improved convergence rate when more communication links was present.

To combine the behaviour of obstacle and collision avoidance with the formation controller, the formation output was chosen as a separate task in the NSB framework. The obstacle-avoidance task was altered to provide collision avoidance between each agent, and a simulation and an experiment was done to prove it's functionality. To introduce obstacle avoidance, two methods was proposed. The first took advantage of the simple kinematic dynamics of the omniwheels, and applied a direct approach to the problem by stacking the tasks in the NSB framework. Since no low-level controller is required, the common mission velocity could easily be included in the chain of tasks and checked for possible collisions.

When a low-level controller is required, this thesis proposes using a leader-agent to create mission velocities that steers the entire group around obstacles. As mission velocities applies to every agent in the group, it does not need to be checked for inter-agent collisions as long as the formation is kept. In this structure, the leader uses an additional NSB controller to generate the mission velocity. To test this setup, a Search and Rescue mission was simulated. In this simulation, five agents including one leader successfully navigated in an environment with static obstacles while holding a minimum-drag like formation. Further, when the object of interest was reached, the formation was shifted to a circle to enclose the target.

The output of the kinematic controllers presented in this thesis could also be used to generate references for other control systems, e.g. a guidance system for marine vessels. Although this topic is not discussed in detail in this thesis, it could be an interesting topic for future research.

Appendix A

Graph Theory

This chapter gives a brief introduction to basic graph theory. Although the source of this material is listed in [Bai et al., 2011], this is standard theory found in many textbooks.

A.1 Formal definition of a graph

A graph G is an abstract representation of a set $\mathscr{V}$ of nodes, connected by links E. These links may either be directed or undirected, and describes which direction information can flow between the nodes. A directed link (i, j) is an *incomming link* to node j and an *outgoing link* from node i, and is viewed as an arrow from node i, pointing towards node j. In a directed link (i, j) we denote node i as being on the *negative* side of the link, and node j as the *positive* end of the link.

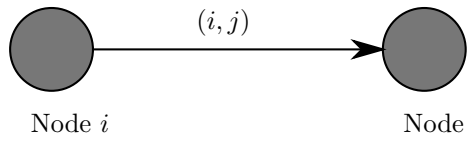


Figure A.1: A simple graph, showing the direction of the link (i, j).

If both (i, j) and (j, i) is in E, we combine the link to an undirected link, and is displayed as a bi-directional arrow.

We say node i is a *neighbour* of node j if the link (i, j) exists in the graph G.

The graph G is called *undirected* when it only contains undirected links, otherwise it is a *directed* graph.

An undirected graph is called *connected* if there exists a path from any one node to another.

A.2 Graph incidence matrix D

For an undirected graph, we may assign an orientation to G by considering one of the two nodes of a link to be the positive end. Further, we denote the set $\mathscr{L}_i^+$ ($\mathscr{L}_i^-$) of links for which node i is at the positive (negative) end.

Denote ℓ as the number of links in G, and let N be the number of nodes. We then define the $N \times \ell$ incidence matrix D of an undirected graph G as

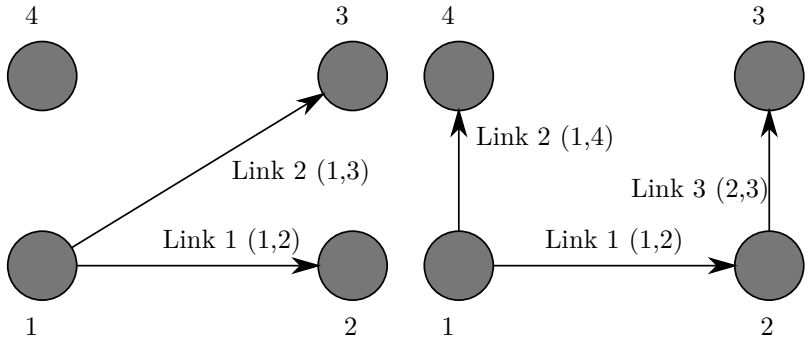
$$d_{ik} := \begin{cases} +1 & \text{if} k \in \mathscr{L}_i^+ \\ -1 & \text{if} k \in \mathscr{L}_i^- \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

For example, consider the graph in Figure A.2(a). It has two links, $(1, 2)$ and $(2, 3)$. The incidence matrix D for this graph is thus given by:

$$D = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{A.2}$$

It can be seen that the i'th row dictates the links going in (+1) and out (-1) of node i, and the k'th column shows for which two nodes the link k is connected to. The incidence matrix for Figure A.2(b) is given in (A.3).

$$D = \begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{A.3}$$



(a) Example graph with two links.

(b) Example of a connected graph with three links.

Figure A.2: Two different graphs to illustrate the computation of the graph incidence matrix.

Appendix B

Additional Passivity Theorems

B.1 Symetric input-output transformation

As stated in [Bai et al., 2011];
Lemma B.1. *Let the system $\mathscr{H}$ in Figure B.1 be passive, and let E be a matrix with a compatible dimension. Then the system in Figure B.1 is passive from $\bar{u}$ to $\bar{y}$.*

Proof. Note that $u^{\mathrm{T}}y = (E^{\mathrm{T}}\bar{u})^{\mathrm{T}}y = \bar{u}^{\mathrm{T}}\bar{y}$. Thus, the passivity from u to y translates to the passivity from $\bar{u}$ to $\bar{y}$. □

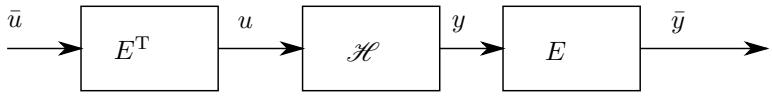


Figure B.1: Pre- and post- multiplication of a matrix and its transpose preserves the passivity of $\mathscr{H}$.

Appendix C

Supplementary Figures

C.1 The HSV image format

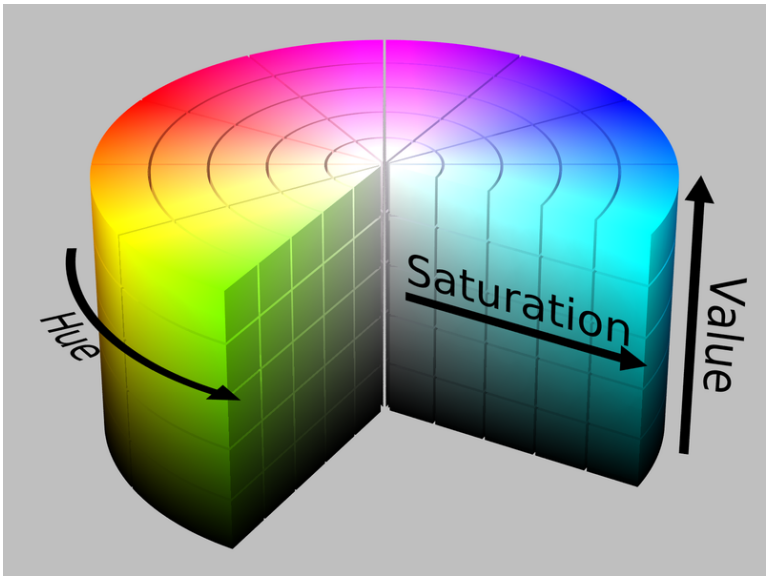


Figure C.1: Description of the HSV color model. *Courtesy wikipedia.org*

C.2 Agent velocities from Search and Rescue simulation

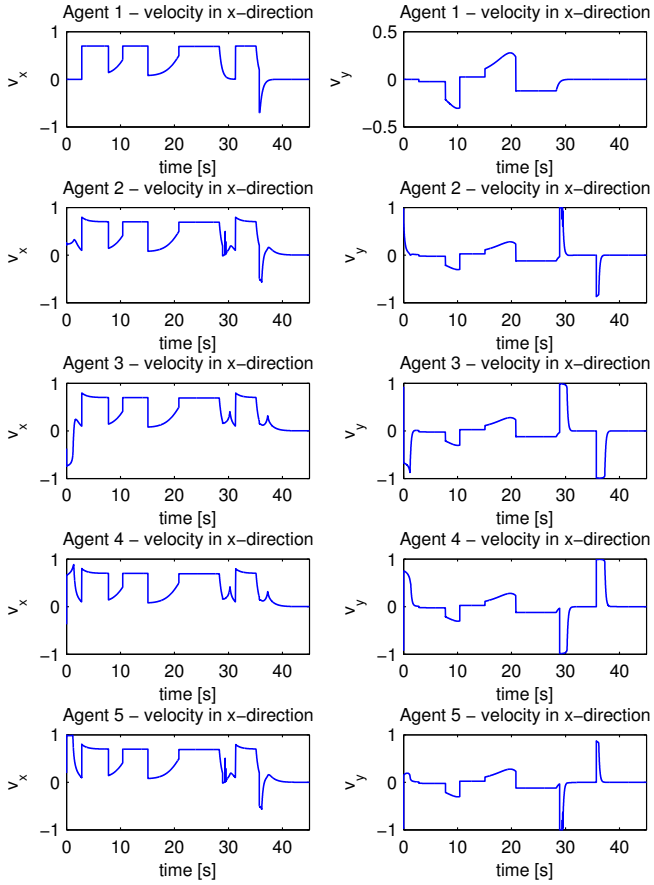


Figure C.2: Velocities of Agents 1-5 in both directions. The data is from the simulation of the Search and Rescue mission in Chapter 6.

Bibliography

Antonelli, G., Arrichiello, F., and Chiaverini, S. (2008a). *Flocking for multi-robot systems via the Null-Space-based Behavioral control* (cited on p. 20).

Antonelli, G., Arrichiello, F., and Chiaverini, S. (2008b). "The NSB control for 3-dimensional flocking of multi-robot systems." In: *2008 International Conference on Information and Automation*, pp. 101–106 (cited on p. 3).

Antonelli, G., Arrichiello, F., and Chiaverini, S. (2005). "The null-space-based behavioral control for mobile robots." In: *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 15–20 (cited on pp. iii, v, 19).

Antonelli, G., Arrichiello, F., Chiaverini, S., and Cassino, S. (2009). "Experiments of Formation Control With Multirobot Systems Using the Null-Space-Based Behavioral Control." In: 17.5, pp. 1173–1182 (cited on p. 20).

Arcak, M. (2007). "Passivity as a Design Tool for Group Coordination." In: *IEEE Transactions on Automatic Control* 52.8, pp. 1380–1390 (cited on pp. iii, v, 2).

Arrichiello, F. (2006). "Coordination control of multiple mobile robots." PhD thesis (cited on pp. 3, 19–22, 24–26).

Arrichiello, F., Heidarsson, H. K., Chiaverini, S., and Sukhatme, G. S. (Dec. 2011). "Cooperative caging and transport using autonomous aquatic surface vehicles." In: *Intelligent Service Robotics* 5.1, pp. 73–87 (cited on pp. 3, 20).

Bai, H., Arcak, M., and Wen, J. T. (July 2007). "Group Coordination when the Reference Velocity is Available Only to the Leader: An Adaptive Design." In: *2007 American Control Conference* 3, pp. 5400–5405 (cited on p. 2).

Bai, H., Arcak, M., and Wen, J. T. (2011). *Cooperative control design: A systematic, passivity-based approach* (cited on pp. 2, 4, 35–37, 39, 41, 42, 58, 81, 85).

Beagleboard (2013). *Homepage of the BeagleBone project: http://beagleboard.org* (cited on p. 9).

Børhaug, E., Pavlov, A., Panteley, E., and Pettersen, K. Y. (May 2011). "Straight Line Path Following for Formations of Underactuated Marine Surface Vessels." In: *IEEE Transactions on Control Systems Technology* 19.3, pp. 493–506 (cited on p. 2).

Børhaug, E., Pavlov, A., and Pettersen, K. Y. (2007). "Straight line path following for formations of underactuated underwater vehicles." In: *Decision and Control, 2007 ...* 1, pp. 2905–2912 (cited on p. 2).

Børhaug, E., Pavlov, A., and Pettersen, K. Y. (2006). "Cross-track formation control of underactuated surface vessels." In: *Proceedings of the 45th IEEE Conference on Decision & Control*, pp. 5955–5961 (cited on p. 2).

Brooks, R. A. (1986). "A Robust Layered Control System for a Mobile Robot." In: *IEEE Journal of Robotcs and Automation* 2, pp. 14–23 (cited on p. 20).

Canny, J. (1986). "A Computational Approach to Edge Detection." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 8.6, pp. 679–698 (cited on p. 11).

Cao, Y., Yu, W., Ren, W., and Chen, G. (Feb. 2013). "An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination." In: *IEEE Transactions on Industrial Informatics* 9.1, pp. 427–438 (cited on p. 2).

Dunbar, W. B. and Murray, R. M. (Apr. 2006). "Distributed receding horizon control for multi-vehicle formation stabilization." In: *Automatica* 42.4, pp. 549–558 (cited on p. 2).

Emery, R. and Balch, T. (2001). "Behavior-based control of a non-holonomic robot in pushing tasks." In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)* 3, pp. 2381–2388 (cited on p. 20).

Fax, J. and Murray, R. (Sept. 2004). "Information Flow and Cooperative Control of Vehicle Formations." In: *IEEE Transactions on Automatic Control* 49.9, pp. 1465–1476 (cited on p. 2).

Fossen, T. I. (2011). *Marine Craft Hydrodynamics and Motion Control.* Wiley (cited on pp. 12, 17).

Ihle, I.-A. F., Arcak, M., and Fossen, T. I. (Sept. 2007). "Passivity-based designs for synchronized path-following." In: *Automatica* 43.9, pp. 1508–1518 (cited on pp. 2, 35).

Khalil, H. K. (2002). *Nonlinear Systems.* 3rd editio. Prenctice Hall (cited on pp. 17, 37, 38, 41).

Kleppe, A. L. (2013). "Hardware Platform for a Multi-Robot System." Master Thesis. NTNU (cited on p. 1).

Murphy, R. (2000). *Introduction to AI robotics.* 1st ed. Cambridge, Mass.: MIT Press (cited on p. 3).

Murray, R. M. (2007). "Recent Research in Cooperative Control of Multivehicle Systems." In: *Journal of Dynamic Systems, Measurement, and Control* 129.5, p. 571 (cited on p. 2).

Olfati-Saber, R. and Murray, R. (2002). "Distributed Cooperative Control of Multiple Vehicle Formations Using Structural Potential Functions." In: *Proc. of IFAC World Congress*. Elsevier (cited on p. 2).

OpenCV (2013). *Homepage of the OpenCV project: http://opencv.org* (cited on p. 10).

Qu, Z., Wang, J., and Hull, R. (2008). "Cooperative control of dynamical systems with application to autonomous vehicles." In: *IEEE Transactions on Automatic Control* 53.4, pp. 894–911 (cited on p. 2).

RaspberryPI (2013). *Homepage of the Raspberry PI project: http://raspberrypi.org* (cited on p. 9).

Sadowska, A. and Kostic, D. (2012). "Distributed formation control of unicycle robots." In: *2012 IEEE International Conference on Robotics and Automation*. iii, pp. 1564–1569 (cited on p. 2).

Shiyou, D., Xiaoping, Z., and Guoqing, L. (Oct. 2011). "The Null-Space-Based Behavioral Control for Swarm Unmanned Aerial Vehicles." In: *2011 First International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 1003–1006 (cited on pp. 3, 20).

Skjetne, R., Fossen, T. I., and Kokotović, P. V. (Mar. 2004). "Robust output maneuvering for a class of nonlinear systems." In: *Automatica* 40.3, pp. 373–383 (cited on p. 2).

Skøien, K. R. and Vermeer, H. (2010). *General Platform for Unmanned Autonomous Systems*. Tech. rep. (cited on p. 9).

Suzuki, S. and Abe, K. (1985). "Topological Structural Analysis of Digitized Binary Images by Border Following." In: *CVGIP* 30.1, pp. 32–46 (cited on p. 11).

Verret, S. (2005). "Current state of the art in multirobot systems." In: *Defence Research and Development Canada-Suffield, ...* December (cited on p. 3).

Zhang, H., Lewis, F., and Das, A. (2011). "Optimal design for synchronization of cooperative systems: state feedback, observer and output feedback." In: *IEEE Transactions on Automatic Control* 56.8, pp. 1948–1952 (cited on p. 2).