Comparison of Distributed Model Predictive Control Approaches for Transporting a Load by a Formation of Mobile Robots

Henrik Ebel, Ehsan Sharafian Ardakani, Peter Eberhard

Institute of Engineering and Computational Mechanics
University of Stuttgart
Pfaffenwaldring 9, 70569 Stuttgart, Germany
[henrik.ebel, ehsan.sharafian, peter.eberhard]@itm.uni-stuttgart.de

Abstract

This paper investigates and discusses two different distributed formation control approaches based on model predictive control (MPC). Specifically, the presented control schemes are used to govern the motion of omnidirectional mobile robots that shall maintain a given formation shape while following a path through a previously unknown environment. The setup of the control schemes accounts for the requirements of transporting an elastic plate purely by normal and friction forces. The intricacy of this task motivates the choice of model predictive control since it allows to explicitly constrain the movements of the robots. The two schemes analyzed in this contribution are fundamentally different in their optimization and communication strategies. The performance of the schemes is carefully examined in various simulations. These include situations in which the robots have to shrink the formation in order to squeeze through narrow passages. An exemplary experimental result involving real robot hardware is also presented.

Keywords: formation control, model predictive control, distributed control, transportation, motion in unknown environments

1. Introduction

Over the last couple of years, technological advances have made it seem increasingly feasible to solve complicated tasks by employing a group of autonomous devices that need to cooperate to achieve a common goal. Driven by possible advantages like increased flexibility and robustness, this has led to a growing scientific interest in the development of distributed control schemes that make this vision realizable from a control-theoretic point of view [1, 2].

One common application for this kind of distributed control is the control of a group of mobile robots, with formation control being a typical task. In the latter, each individual robot employs a control law, using only the information available to itself, to stay in a certain formation shape while moving the formation as a whole to a certain point or along a provided trajectory. The specific task studied in this paper consists of controlling a formation of omnidirectional mobile robots that transport a plate through an obstacle-ridden environment. This setup is illustrated in Figure 1. The plate is not fixed to the robots in any way but rather transported only by friction and normal forces. Therefore, when compared with typical formation control tasks, it is even more paramount that the robots behave in a well-controlled way, without exerting too large propulsion forces that lead to excessive slippage between the robots and the plate. Hence, it seems worthwile to employ a control scheme that allows to explicitly constrain the control inputs, which makes model predictive control (MPC) a natural choice. MPC is an optimization-based control scheme in which, in every time step, a constrained optimization problem is solved over a time-horizon of finite length to determine a sequence of control inputs that is optimal with respect to some given cost function. In the optimization problem, to the end of being able to evaluate the cost function and the constraints, the system dynamics is used to predict the state of the system over the optimization horizon under the influence of the considered control input. After solving the optimization

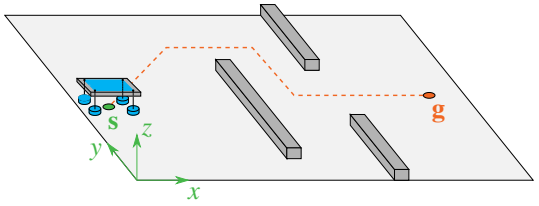

Figure 1: Illustration of four robots transporting a plate through an environment with three obstacles. A possible path leading from the starting point **s** to the goal point **g** is dashed in orange.

problem, the first part of the calculated optimal input sequence is applied to the system, the horizon recedes by one time step and the optimization problem is solved anew with the measured or observed current system state as the new initial condition, thereby establishing a feedback in the control loop [3].

However, typically, MPC is applied to systems where one central entity solves an overall optimization problem to determine the optimal control behavior. In contrast to that, the distributed control of mobile robots necessitates that each robot solves its own optimization problem and may only exchange information with the other robots before and after the optimization. Fortunately, in recent years, a multitude of algorithms and methods have been developed that make it possible to apply model predictive control to a group of distributed systems [1]. The specific schemes proposed and compared in this paper are rooted in the general theory presented in [4] and [5]. These two approaches are fundamentally different in their optimization and communication strategies. In [4], the systems optimize and transmit information sequentially one after another, while in [5], they can optimize concurrently but possibly need several optimization and communication iterations to achieve a satisfying performance. For this reason, hereafter, the former scheme will be referred to as sequential distributed MPC (SDMPC), while the latter will be denoted as iterative distributed MPC (IDMPC). As will be seen, these and further theoretic differences lead to inherently different proposed formulations of the optimization problems for this paper's formation control task. The novel contribution of this paper is to consider and thoroughly examine the usefulness of these kinds of schemes for a demanding formation control and transportation task.

The paper is organized as follows. Firstly, the mathematical models for the simulation of the robots, the plate and the interactions between the robots and the plate are briefly presented. Then, the following section deals with an introduction of the two discussed control schemes and the designs of the controllers employed for the plate transportation. In Section 4, some simulation results for the two schemes and an experimental result for the IDMPC scheme applied to hardware robots are presented and thoroughly discussed. The findings are then summarized in the final section.

2. Problem Formulation

For simulation purposes, the robots themselves, which are assumed to be omnidirectional and hence are not subjected to non-holonomic constraints, are modeled as point masses with friction, yielding

$$\mathbf{M}\ddot{\mathbf{q}}_i + \mathbf{D}\dot{\mathbf{q}}_i = \mathbf{u}_i \tag{1}$$

for the i-th robot with the diagonal mass and damping matrices $\mathbf{M} \in \mathbb{R}^{2\times 2}$ and $\mathbf{D} \in \mathbb{R}^{2\times 2}$. The vector $\mathbf{q}_i \in \mathbb{R}^2$ signifies the robot's position in the inertial frame of reference and $\mathbf{u}_i \in \mathbb{R}^2$ denotes the propulsion forces. Geometrically, for issues like collision detection, the robots are assumed to have a circular footprint of finite diameter $d > 0$.

The plate is assumed to rest on rigid, vertical beams that are attached to each robot. It is assumed that, at each time instant, at least three non-aligned robots are in contact with the plate. The contact areas are assumed to be point shaped. While, without loss of generality, it is possible to employ arbitrary friction laws, the results in this paper make use of a Coulomb-type friction model taken from [6]. For an individual contact point at which a normal force of absolute value F_N is acting and the slipping velocity $\mathbf{v}_\mathrm{slip} \in \mathbb{R}^3$ is observed, the friction force reads

$$\mathbf{F}_\mathrm{F}\left(\mathbf{v}_\mathrm{slip}\right) = \gamma_1 \tanh(\gamma_2 \left|\mathbf{v}_\mathrm{slip}\right|) F_\mathrm{N} \mathbf{v}_\mathrm{slip} / \left|\mathbf{v}_\mathrm{slip}\right|, \qquad \mathbf{F}_\mathrm{F}(\mathbf{0}) := \mathbf{0} \tag{2}$$

with the material-dependent parameter γ_1 and the design parameter γ_2. To simulate the kinematics of the plate, it is assumed here to be a rigid cuboid with a uniformly distributed mass. The generalized rigid body coordinates are chosen to $\mathbf{y} = \begin{bmatrix} x & y & z & \alpha & \beta & \gamma \end{bmatrix}^\mathrm{T}$. Therein, α, β and γ are the Cardan angles describing the orientation of the plate. With this, the plate's equations of motion can be given in the form

$$\mathbf{M}_\mathrm{p}\ddot{\mathbf{y}} + \mathbf{k}_\mathrm{p}\left(\mathbf{y}, \dot{\mathbf{y}}\right) = \mathbf{q}_\mathrm{p}\left(\mathbf{y}\right) \tag{3}$$

with the mass matrix $\mathbf{M}_\mathrm{p}$, the vector of generalized forces $\mathbf{k}_\mathrm{p}$ and the vector of applied forces $\mathbf{q}_\mathrm{p}$. The latter consists of the force of gravity in the negative z-direction, the normal and friction forces at the contact points and the moment generated by these contact forces. Thus, for being able to simulate the system consisting of $N \geq 3$ robots and the plate, the normal forces at the contact points need to be calculated appropriately. However, for more than three robots, the motion of the plate when modeled as a rigid body would be overdetermined. Hence, the plate is discretized using the finite element method by employing linear Reissner-Mindlin plate elements. The finite element equation system is then recast into a linear complementarity problem (LCP) [7] so that the complementarity condition ensures that the unilateral contact is modeled appropriately.

With the above considerations, it is possible to appropriately simulate the behavior of the system consisting of a plate and a number of robots that are each governed by given control inputs $\mathbf{u}_i(\cdot)$. Henceforth, the following section deals with the introduction of the studied control laws.

3. Control Schemes

In the following, the SDMPC and IDMPC schemes are introduced and the specific setup of the optimization problem for the plate transportation task is outlined.

Principally, as will be seen, in the studied SDMPC scheme, only dynamically decoupled systems can be considered. Therefore, in each robot's MPC optimization problem, directly the individual robots' dynamics is used. Consequently, the considered overall system dynamics does not contain any couplings between the robots. All couplings necessary to achieve the cooperative formation control goal are present in the cost function. In contrast to that, in the proposed IDMPC scheme, an artificially designed system is used that contains all of the robots' dynamics, the dynamics of the formation's center of mass, and the dynamics of the robots relative to this center of mass. In principle, the formation control task then comes down to tracking a trajectory with this system in a distributed fashion by using IDMPC. Thus, while the system dynamics of the artificial system is more elaborate and contains more states, the cost function is much simpler.

3.1. Sequential Distributed MPC

To obtain a model usable within the MPC scheme, the individual dynamics of the robots from Equation (1) are transferred to the state-space and discretized using zero-order hold with the sampling time T_s^c, yielding

$$\mathbf{x}_i(k+1) = \mathbf{A}\mathbf{x}_i(k) + \mathbf{B}\mathbf{u}_i(k), \qquad k \in \mathbb{N}_0, \quad \mathbf{x}_i(0) := \mathbf{x}_{i,0}, \quad i \in \{1,\dots,N\} \tag{4}$$

with $\mathbf{x}_i := \begin{bmatrix}\mathbf{q}_i^\mathrm{T} & \dot{\mathbf{q}}_i^\mathrm{T}\end{bmatrix}^\mathrm{T}$. Notation-wise, hereafter, at time step $t \in \mathbb{N}_0$, the predicted state sequence for a planned input sequence $\mathbf{u}_i(\cdot\,|\,t)$ will be denoted as $\mathbf{x}_i(\cdot\,|\,t)$. The robots other than robot i are referred to as neighbors of robot i with the corresponding set of indices $\mathscr{N}_i := \{1,\dots,N\} \setminus \{i\}$. Their predicted state sequences shall be collected in the vector $\mathbf{x}_{-i}^\mathrm{T}(\cdot\,|\,t) := \begin{bmatrix}\mathbf{x}_{\mathrm{n}_1}^\mathrm{T}(\cdot\,|\,t) & \mathbf{x}_{\mathrm{n}_2}^\mathrm{T}(\cdot\,|\,t) & \dots & \mathbf{x}_{\mathrm{n}_{N-1}}^\mathrm{T}(\cdot\,|\,t)\end{bmatrix}^\mathrm{T}$ with $n_j \in \mathscr{N}_i$ and, without loss of generality, $n_1 < n_2 < \dots < n_{N-1}$. The first step in devising an appropriate optimization problem is to come up with a cost function that accurately reflects the goals of the formation control task. To this end, we define the abbreviations

$$\mathbf{r}_i(\cdot\,|\,t) := \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{x}_i(\cdot\,|\,t), \qquad \hat{\mathbf{r}}_{n_j}(\cdot\,|\,t) := \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\hat{\mathbf{x}}_{n_j}(\cdot\,|\,t) \quad \forall n_j \in \mathscr{N}_i, \tag{5}$$

$$\hat{\mathbf{c}}_i(\cdot\,|\,t) := \frac{1}{N}\left(\mathbf{r}_i(\cdot\,|\,t) + \sum_{n_j \in \mathscr{N}_i} \hat{\mathbf{r}}_{n_j}(\cdot\,|\,t)\right), \tag{6}$$

which define the predicted position trajectory of robot i, the assumed position trajectories of the neighbors and the assumed trajectory of the formation's center of mass. The matrix $\mathbf{I} \in \mathbb{R}^{2\times 2}$ denotes the identity. Since, in general, the exact trajectories predicted in the optimization problem of the neighboring robots are not known to robot i, a suitable approximation denoted by $\hat{\mathbf{x}}_{n_j}(\cdot\,|\,t)$ is needed. For the moment, assume that such an approximation is available. The calculation of these assumed trajectories is part of the sequential optimization strategy outlined below. Finally, $\mathbf{r}_\mathrm{rc}(\cdot\,|\,t) \in \mathbb{R}^2$ shall refer to the provided reference trajectory as planned at time t, and $\mathbf{r}_i^\Delta \in \mathbb{R}^2$ shall denote the desired position of robot i relative to the formation's center of mass. In general, this quantity might also be time-dependent, for instance if the formation is reorganized, but this dependency will be omitted in the subsequent notation in the interest of an improved readability.

With the above, robot i's cost function for an optimization horizon of length H can be defined in the form

$$\begin{aligned} J_i(\mathbf{x}(t), \hat{\mathbf{r}}_{-i}(\cdot\,|\,t), \mathbf{u}_i(\cdot\,|\,t)) = \sum_{k=t}^{t+H-1} \Bigg[& \left\|\hat{\mathbf{c}}_i(k\,|\,t) - \mathbf{r}_\mathrm{rc}(k\,|\,t)\right\|_{\mathbf{Q}_\mathrm{r}}^2 + \left\|\mathbf{r}_i(k\,|\,t) - \hat{\mathbf{c}}_i(k\,|\,t) - \mathbf{r}_i^\Delta\right\|_{\mathbf{Q}_\mathrm{f}}^2 \\ & + \sum_{n_j \in \mathscr{N}_i} \left\|\hat{\mathbf{r}}_{n_j}(k\,|\,t) - \hat{\mathbf{c}}_i(k\,|\,t) - \mathbf{r}_{n_j}^\Delta\right\|_{\mathbf{Q}_\mathrm{r}}^2 + \left\|\mathbf{u}_i(k\,|\,t)\right\|_{\mathbf{R}}^2 \Bigg] \end{aligned} \tag{7}$$

with $\|\mathbf{v}\|_\mathbf{L}^2 := \mathbf{v}^\mathrm{T}\mathbf{L}\mathbf{v}$ for some positive semi-definite matrix $\mathbf{L}$ and a matching vector $\mathbf{v}$. The three different weighting matrices in Equation (7) are chosen to be positive definite. The terms in the square brackets of the equation correspond to the tracking error of the formation's center of mass, the relative errors within the formation of robot i and its neighbors, and the robot's control effort.

Thereby, using an appropriate set of constraints, the MPC optimization problem for an individual robot can now be

formulated as

$$
\begin{aligned}
\underset{\mathbf{u}_i(\cdot\,|\,t)}{\text{minimize}} \quad & J_i(\mathbf{x}_i(t), \hat{\mathbf{r}}_{-i}(\cdot\,|\,t), \mathbf{u}_i(\cdot\,|\,t)) \\
\text{subject to} \quad & \mathbf{x}_i(k+1\,|\,t) = \mathbf{A}\mathbf{x}_i(k\,|\,t) + \mathbf{B}\mathbf{u}_i(k\,|\,t),\ k \in \{t, \dots, t+H-1\}, \\
& \|\mathbf{u}_i(t+k\,|\,t)\|_\infty \leq u_{\max},\ k \in \{0, \dots, H-1\}, \\
& \|\Delta\mathbf{u}_i(t+k\,|\,t)\|_\infty \leq \Delta u_{\max},\ k \in \{0, \dots, H-1\}, \\
& \left\| \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x}_i(t+k\,|\,t) \right\|_\infty \leq v_{\max},\ k \in \{h, \dots, H-1\}, \\
& \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x}_i(t+H\,|\,t) = \mathbf{0}, \\
& \mathbf{x}_i(t\,|\,t) = \mathbf{x}_i(t)
\end{aligned}
\tag{8}
$$

with $\Delta\mathbf{u}_i(k\,|\,t) := \mathbf{u}_i(k\,|\,t) - \mathbf{u}_i(k-1\,|\,t)$. The optimal input sequence that solves the optimization problem for time step t shall be denoted by $\mathbf{u}_i^\star(\cdot\,|\,t)$, with the resulting optimal state sequence referred to as $\mathbf{x}_i^\star(\cdot\,|\,t)$. At time t, $\mathbf{x}_{-i}^\star(\cdot\,|\,t-1)$ shall represent the neighbors' optimal state sequences from the preceding time step. Inspired by [4], the robots now solve their respective optimization problems sequentially one after the other, with intermediate communication steps. The precise scheme is depicted in Algorithm 1.

It is worth noting that each robot only needs to send information once every time step, although, for $N > 2$, information needs to be waited for and received more than once. Since the robots optimize sequentially, the cumulative optimization time increases linearly with N. Henceforth it makes sense to examine a scheme that allows the robots to optimize concurrently. However, note that the sequential scheme in principle would allow for couplings in the constraints of the optimization problem while still maintaining recursive feasibility, although no such constraints are used in the present application.

Algorithm 1 Sequential distributed MPC scheme

input: initial feasible sequences $\mathbf{x}_i(\cdot\,|\,0)$, $\hat{\mathbf{x}}_{-i}(\cdot\,|\,0)$, $\hat{\mathbf{u}}_i(\cdot\,|\,0)$ $\forall i \in \{1, \dots, N\}$

2: **at all time steps** $t \geq 1$**:**

all systems $i \in \{1, \dots, N\}$ set

$$
\hat{\mathbf{x}}_{n_j}(t+k\,|\,t) = \begin{cases} \mathbf{x}_{n_j}^\star(t+k\,|\,t-1) & \text{for } k \in \{0, \dots, H-1\}, \\ \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{x}_{n_j}^\star(t+H-1\,|\,t-1) & \text{for } k = H \end{cases} \qquad \forall n_j \in \mathcal{N}_i
$$

$$
\hat{\mathbf{u}}_i(t+k\,|\,t) = \begin{cases} \mathbf{u}_i^\star(t+k\,|\,t-1) & \text{for } k \in \{0, \dots, H-2\}, \\ \mathbf{0} & \text{for } k = H-1, \end{cases}
$$

$$
\hat{\mathbf{x}}_{-i}(\cdot\,|\,t) = \begin{bmatrix} \hat{\mathbf{x}}_{n_1}^{\mathrm{T}}(\cdot\,|\,t) & \hat{\mathbf{x}}_{n_2}^{\mathrm{T}}(\cdot\,|\,t) & \dots & \hat{\mathbf{x}}_{n_{\text{end}}}^{\mathrm{T}}(\cdot\,|\,t) \end{bmatrix}^{\mathrm{T}},\ n_1, \dots, n_{\text{end}} \in \mathcal{N}_i
$$

at all time steps $t \geq 0$**:**

for $i = 1, \dots, N$ **do**

solve (8) using $\hat{\mathbf{x}}_{-i}(\cdot\,|\,t)$

$\Rightarrow$ solutions $\mathbf{u}_i^\star(\cdot\,|\,t)$, $\mathbf{x}_i^\star(\cdot\,|\,t)$

send $\mathbf{x}_i^\star(\cdot\,|\,t)$ to all neighbors, replacing $\hat{\mathbf{x}}_i(\cdot\,|\,t)$ in the neighbors' memory

end for

4: all systems $i \in \{1, \dots, N\}$ apply $\mathbf{u}_i^\star(t\,|\,t)$

$t = t + 1$

6: go to step 2

3.2. Iterative Distributed MPC

As mentioned at the outset, the proposed IDMPC controller is set up in a fundamentally different way than the presented SDMPC controller. As will be seen, it is possible to consider couplings in the system dynamics which is used in the scheme's optimization problem. Therefore, in the subsequent disquisitions, an artificial system is constructed that models the formation's dynamics as a whole. To make this system dimension-wise smaller, without loss of generality, we do not consider the full robot model from Equation (1) in this construction process but rather the simple single-integrator dynamics $\dot{\mathbf{x}}_i = \mathbf{u}_i$ for each robot $i \in \{1, \dots, N\}$. Then, $\mathbf{x}_i \in \mathbb{R}^2$ is the position of robot i. Consequentially, in the end, the IDMPC scheme will not deliver propulsion forces for the robots but rather desired velocities. Nevertheless, for comparability, the simulations shall still be performed using the original robot model. Thus, an auxiliary PI-controller

with an anti-wind up mechanism and the same bounds on the propulsion forces as for the SDMPC simulations will be used to govern the propulsion forces based on the IDMPC scheme's desired velocity values.

The first step consists of the construction of a state-space equation that describes the dynamics of the formation. To this end, note that the dynamics of the formation's center of mass are now given by $\dot{\mathbf{x}}_\mathrm{c} = 1/N \sum_{i=1}^{N} \dot{\mathbf{x}}_i = 1/N \sum_{i=1}^{N} \mathbf{u}_i$.

Furthermore,

$$\bar{\mathbf{x}} := \begin{bmatrix} \mathbf{x}_1^\mathrm{T} & \cdots & \mathbf{x}_N^\mathrm{T} & \mathbf{x}_\mathrm{c}^\mathrm{T} & (\mathbf{x}_\mathrm{c}-\mathbf{x}_1)^\mathrm{T} & \cdots & (\mathbf{x}_\mathrm{c}-\mathbf{x}_N)^\mathrm{T} \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{2(2N+1)}, \tag{9}$$

$$\bar{\mathbf{u}} := \begin{bmatrix} \mathbf{u}_1^\mathrm{T} & \cdots & \mathbf{u}_N^\mathrm{T} \end{bmatrix}^\mathrm{T} \in \mathbb{R}^{2N}. \tag{10}$$

define the state and control input of the overall formation dynamics. To the end of describing these dynamics, consider

$$\bar{\mathbf{B}} := \frac{1}{N} \begin{bmatrix} \mathbf{B}_\mathrm{t} \\ \mathbf{B}_\mathrm{b} \end{bmatrix} \in \mathbb{R}^{(2N+1)\times(N)} \text{ with } \mathbf{B}_\mathrm{t} := \begin{bmatrix} N & 0 & \cdots & 0 \\ 0 & N & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots \\ 0 & \cdots & 0 & N \end{bmatrix} \text{ and } \mathbf{B}_\mathrm{b} := \begin{bmatrix} 1 & \cdots & \cdots & 1 \\ 1-N & 1 & \cdots & 1 \\ 1 & 1-N & 1 & \cdots \\ \vdots & \ddots & \ddots & \ddots \\ 1 & \cdots & 1 & 1-N \end{bmatrix}. \tag{11}$$

Finally, with $\otimes$ denoting the Kronecker product, we identify

$$\dot{\bar{\mathbf{x}}} = \left(\bar{\mathbf{B}} \otimes \mathbf{I}\right) \bar{\mathbf{u}} =: \bar{\mathbf{B}}_2 \bar{\mathbf{u}}. \tag{12}$$

Naturally, for a given center of mass and $N-1$ given relative robot positions, the position of the N^th robot is also determined. Thus, for tracking, define a possible system output as

$$\mathbf{y}_\mathrm{p} = \begin{bmatrix} \mathbf{0}_{2N\times 2N} & \mathbf{I}_{2N\times 2N} & \mathbf{0}_{2N\times 2} \end{bmatrix} \bar{\mathbf{x}} =: \hat{\mathbf{C}} \bar{\mathbf{x}} \tag{13}$$

with the matrix dimensions given in the subscripts. Without loss of generality, in Equation (13), the relative position of the last robot is not part of the output. Furthermore, since $\mathrm{rank}(\hat{\mathbf{C}}\bar{\mathbf{B}}_2) = 2N$, the output $\mathbf{y}_\mathrm{p}$ is controllable.

Next, the formation dynamics is discretized, once again with a sampling time of T_s^c, yielding the discrete formation dynamics

$$\bar{\mathbf{x}}(k+1) = \mathbf{A}_\mathrm{d} \bar{\mathbf{x}}(k) + \mathbf{B}_\mathrm{d} \bar{\mathbf{u}}(k), \tag{14}$$

$$\mathbf{y}_\mathrm{p}(k) = \mathbf{C}_\mathrm{d} \bar{\mathbf{x}}(k). \tag{15}$$

To the end of being able to better control the smoothness of the control signals when designing the cost function, the system will be reformulated once again, so that $\mathbf{u}(k) := \Delta\bar{\mathbf{u}}(k) := \bar{\mathbf{u}}(k+1) - \bar{\mathbf{u}}(k)$ serves as the new control input. To accomplish this, we define

$$\mathbf{A}_\mathrm{f} = \begin{bmatrix} \mathbf{A}_\mathrm{d} & \mathbf{B}_\mathrm{d} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \quad \mathbf{B}_\mathrm{f} = \begin{bmatrix} \mathbf{B}_\mathrm{d} \\ \mathbf{I} \end{bmatrix}, \quad \mathbf{C}_\mathrm{f} = \begin{bmatrix} \mathbf{C}_\mathrm{d} & \mathbf{0} \end{bmatrix}, \tag{16}$$

and the new overall system dynamics

$$\mathbf{x}(k+1) = \mathbf{A}_\mathrm{f} \mathbf{x}(k) + \mathbf{B}_\mathrm{f} \mathbf{u}(k), \tag{17}$$

$$\mathbf{y}_\mathrm{p}(k) = \mathbf{C}_\mathrm{f} \mathbf{x}(k) \tag{18}$$

with $\mathbf{x}(t) := \begin{bmatrix} \bar{\mathbf{x}}^\mathrm{T}(t) & \bar{\mathbf{u}}^\mathrm{T}(t-1) \end{bmatrix}^\mathrm{T}$. This system will now be used to state the MPC optimization problem. For notation purposes, decompose

$$\mathbf{y}_\mathrm{p}(t) =: \begin{bmatrix} \mathbf{y}_1(t)^\mathrm{T} & \mathbf{y}_2(t)^\mathrm{T} & \cdots & \mathbf{y}_N(t)^\mathrm{T} \end{bmatrix}^\mathrm{T}, \tag{19}$$

$$\hat{\mathbf{y}}_\mathrm{d}(t) =: \begin{bmatrix} (\mathbf{y}_\mathrm{d}(t))_1^\mathrm{T} & (\mathbf{y}_\mathrm{d}(t))_2^\mathrm{T} & \cdots & (\mathbf{y}_\mathrm{d}(t))_N^\mathrm{T} \end{bmatrix}^\mathrm{T} \tag{20}$$

with $\mathbf{y}_i(t), (\mathbf{y}_\mathrm{d}(t))_i \in \mathbb{R}^2 \quad \forall i \in \{1,\ldots,N\}$, and $\hat{\mathbf{y}}_\mathrm{d}(\cdot)$ being a reference to be tracked. Then, define N individual stage-cost functions

$$L_i(\mathbf{x},\mathbf{u}) := \|\mathbf{y}_i - (\mathbf{y}_\mathrm{d})_i\|_\mathbf{Q}^2 + \frac{1}{N}\|\mathbf{u}\|_\mathbf{R}^2 \tag{21}$$

with positive definite weighting matrices $\mathbf{Q}$ and $\mathbf{R}$. These stage costs are used to determine the N individual cost functions of the form

$$J_i(\mathbf{x}(t), \mathbf{u}(\cdot\,|\,t)) := \sum_{k=0}^{H-1} L_i(\mathbf{x}(t+k\,|\,t), \mathbf{u}(t+k\,|\,t)). \tag{22}$$

The overall cooperative cost function is now defined as

$$J(\mathbf{x}(t),\mathbf{u}(\cdot\,|\,t)) = \sum_{i=1}^{N} J_i(\mathbf{x}(t),\mathbf{u}(\cdot\,|\,t))\,. \tag{23}$$

Furthermore, for convenience, we define by $\mathbf{C}_v^i$ the matrix that, by multiplication from the left, extracts the velocities of robot i from the overall system state $\mathbf{x}$.

With the above, the MPC optimization problem for the iterative scheme can be given in the form

$$\begin{aligned}
\underset{\mathbf{u}_i(\cdot\,|\,t)}{\text{minimize}} \quad & J(\mathbf{x}(t),\mathbf{u}(\cdot\,|\,t)) \\
\text{subject to} \quad & \mathbf{x}(k+1) = \mathbf{A}_\mathrm{f}\mathbf{x}(k) + \mathbf{B}_\mathrm{f}\mathbf{u}(k),\ k \in \{t,\dots,t+H-1\}, \\
& \mathbf{y}_\mathrm{p}(k) = \mathbf{C}_\mathrm{f}\mathbf{x}(k),\ k \in \{t,\dots,t+H-1\}, \\
& \left\|\mathbf{C}_v^i\mathbf{x}(t+k)\right\|_\infty \leq v_\mathrm{max},\ k \in \{1,\dots,H-1\}, \\
& \mathbf{C}_v^i\mathbf{x}(t+H\,|\,t) = \mathbf{0}, \\
& \mathbf{u}_j(k\,|\,t) = \mathbf{u}_j^n(k\,|\,t) \quad \forall j \neq i, \\
& \mathbf{x}(t\,|\,t) = \mathbf{x}(t).
\end{aligned} \tag{24}$$

In this optimization problem, $n \in \mathbb{N}$ is the number of the current iteration in the iterative distributed MPC algorithm. Since, in general, the optimization problem for one robot i depends on the unknown current control inputs from all robots $j \neq i$, a distributed scheme must be used that delivers a usable approximation of the centralized solution. The iterative distributed MPC algorithm used in this study is summarized in Algorithm 2. In its Gauss-Jacobi-type structure, this distributed optimization strategy is taken from [5, 8, 9]. Note that recursive feasibility is guaranteed from iteration to iteration through the convex combination of the candidate and the optimal solution in step 2 of the algorithm. Also, although not in the scope of this paper, in a modified case, with a constant setpoint and a zero terminal constraint, convergence to the setpoint can be formally established rather conveniently – as long as the problem is initially feasible. Once again, the convex combination in step 2 is key to the proof. In particular, these observations hold independently from the number of iterations $n_\mathrm{max} \geq 1$.

When using, as above, the output $\mathbf{y}_\mathrm{p}$ in the setup of the distributed controllers, the N^th robot would need to be treated differently than the first $N-1$ robots, which can complicate the implementational setup of the controllers. Therefore, one can also use the output

$$\mathbf{y} = \begin{bmatrix}\mathbf{0}_{2(N+1)\times 2N} & \mathbf{I}_{2(N+1)\times 2(N+1)}\end{bmatrix}\bar{\mathbf{x}}. \tag{25}$$

Consequently, the formation control task then consists of tracking a reference $\mathbf{y}_\mathrm{d}(\cdot)$ of higher dimension than before. Naturally, each prescribed setpoint $\mathbf{y}_\mathrm{d}(t) \in \mathbb{R}^{2(N+1)}$ must then be consistent in the way that the desired position of the N^th robot ought to fit to the prescribed positions of the first $N-1$ robots and the formation's center of mass. The overall cost function then contains one additional term for the deviation of the N^th robot from its setpoint. Because of the ease of implementation, the results shown in this paper indeed make use of $\mathbf{y}$ instead of $\mathbf{y}_\mathrm{p}$ as the output.

4. Results and Discussion

In the following, the two schemes are put to the test in simulations to see whether one or both of them are practically suitable to the studied formation control and plate transportation task. Before considering the full plate transportation case, a more simple test case that deals solely with the acquisition of a goal formation shall help to find out whether there are any fundamental differences in the schemes' performances. All simulations are performed in Matlab.

4.1. Formation Acquisition

In this test, four robots, starting at disparate locations with zero velocity, shall form a rectangular aim formation around the point $\begin{bmatrix}5 & 5\end{bmatrix}^\mathrm{T}$. The initial positions and the aim formation are illustrated on the left of Figure 2. The corresponding setpoints for the center of mass of the formation and the relative positions within the formation are directly and constantly set to the desired aim value. The IDMPC scheme is employed with only $n_\mathrm{max} = 1$ iteration to get a baseline for the achievable performance. The other plots in Figure 2 show the x-positions of the robots and the tracking error of the formation's center of mass plotted over time for both control schemes. As can be seen from the results, at least in this test, the differences in behavior between the two schemes is practically negligible. At the same time, it is worth remembering that in the IDMPC scheme, only single-integrator dynamics have been considered in

Algorithm 2 Iterative distributed MPC scheme

1: **input:** initial feasible sequences $\mathbf{u}_i^0(\cdot\,|\,0)\ \forall i \in \{1,\dots,N\}$
2: **at all time steps** $t \geq 1$**:**
for $n = 0,\dots,n_{\max}-1$ **do**
in parallel, all robots $i \in \{1,\dots,N\}$
solve (24) with $\mathbf{u}_j(k\,|\,t) = \mathbf{u}_j^n(k\,|\,t) \quad \forall j \neq i$,
$\Rightarrow$ solution $\mathbf{u}_i^{\star,n}(\cdot\,|\,t)$
set $\mathbf{u}_i^{n+1}(\cdot\,|\,t) = \frac{1}{N}\,\mathbf{u}_i^{\star,n}(\cdot\,|\,t) + \left(1-\frac{1}{N}\right)\mathbf{u}_i^n(\cdot\,|\,t)$
send $\mathbf{u}_i^{n+1}(\cdot\,|\,t)$ to all other robots
end for
3: in parallel, all systems $i \in \{1,\dots,N\}$
apply $\mathbf{u}_i^{n_{\max}}(t\,|\,t)$
set $\mathbf{u}_j^0(t+k\,|\,t+1) = \begin{cases} \mathbf{u}_j^{n_{\max}}(t+k\,|\,t) & \text{for } k \in \{1,\dots,H-1\}, \\ \mathbf{0} & \text{for } k = H \end{cases} \quad \forall j \in \{1,\dots,N\}$
4: $t = t+1$
5: go to step 2

the optimization problem, with an appropriately designed auxiliary PI-controller establishing the compatibility to the second-order simulation model. Consequently, the results indicate that this was indeed an acceptable simplification for the considered robot dynamics.

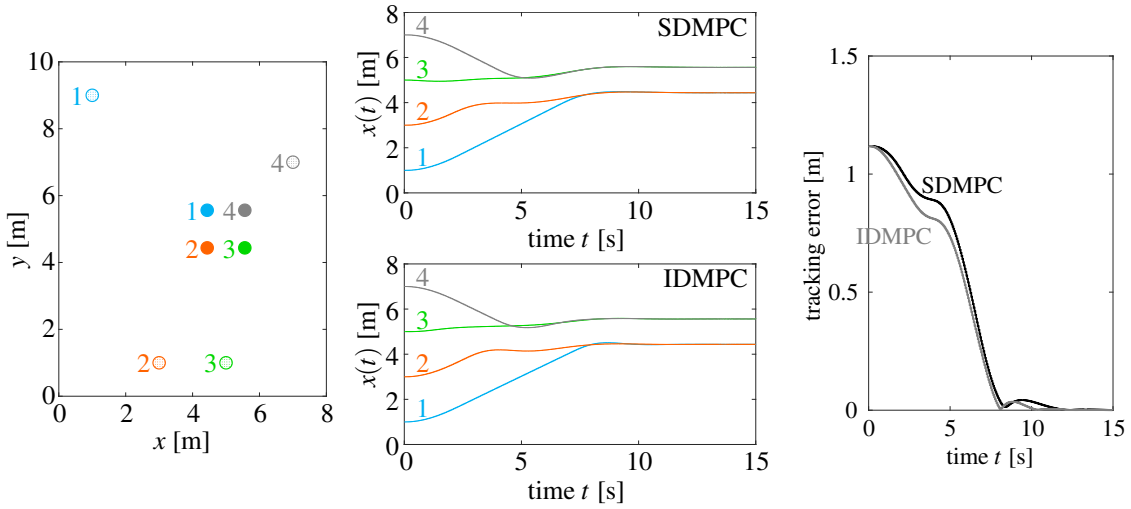

Figure 2: Illustration of the setup and plot of the results of the formation acquisition scenario. In the left picture, the solidly colored circles indicate the aim positions while the stippled circles indicate the initial positions of the robots. In the middle, the upper picture is the result for the robots' positions in the x-directions for the SDMPC scheme, the lower one for the IDMPC scheme, respectively. The right picture shows the tracking error plotted over time, i.e. the distance of the center of mass from the setpoint.

4.2. Plate Transportation

The two control schemes are now put to the test in a demanding plate transportation scenario. For this, five robots are put into an unknown environment that includes narrow passages and various obstacles. The robots shall move the center of mass of the formation, and thereby of the plate, to a certain goal point. The environment is assumed to be unknown to the robots. In previous research, we have devised a mapping, path planning and trajectory generation scheme tailored to the task of plate transportation through unknown environments [10]. In this paper's application, always the robot in the front left of the formation uses simulated, local distance measurements – as would be available from a light detection and ranging sensor with finite accuracy – to construct a map and, in a second step, a path through the environment. In the map, recognized obstacles are approximately memorized in the form of convex polytopes. Non-convex obstacles can be represented using multiple convex polytopes. Expanded versions of the obstacles are used to generate a graph that is used to determine a feasible path through the environment by means of a shortest path algorithm. Whenever

a new path is generated, it is made available to the other robots – for instance together with the data that needs to be communicated anyway within the respective distributed MPC scheme. Using this path, the robots then plan a trajectory, i.e. in particular the velocity along the path, based on the predicted formation and tracking errors of the MPC schemes. Therefore, they will move slower along sections of the path that are expected to be more difficult to traverse. Since obstacles are memorized, this allows to navigate even through maze-like environments. Whenever one of the robots' local distance measurements indicates that an obstacle is closer than a certain threshold, the robots begin to shrink the formation to make narrow passageways traversable in a safe manner. To this end, the traveling height of the plate is assumed to be larger than the height of the obstacles. Results obtained with the two schemes can be found in Figures 3–5. In this test, $n_{\max} = 2$ iterations have been used in the IDMPC scheme. The plate is plotted as a transparent rectangle with thick black borders, obstacles are represented by dark grey areas and the robots are depicted as colored circles. Furthermore, Figure 6 shows the robot velocities in the y-direction for the two schemes. The results show that both schemes are capable of safely delivering the plate to the goal position. Since the environment is unknown, the formation first attempts to drive to the goal in a straight line, notices that the direct path is blocked, then checks the lower part for a free path and finally takes the upper route through a narrow passage which it traverses in shrunken form. In our experience, the only real difference in the performance of the two schemes lies in the fact that the formation shape seems to stay a little bit more rigid for the SDMPC scheme than for the IDMPC scheme. Also, as can be seen in Figure 6, the robot velocities develop a little bit more smoothly in the SDMPC case, which is probably more an effect of the different control structure than of the optimization scheme itself. However, in summary, both schemes control the plate successfully and rather smoothly even through this demanding environment.

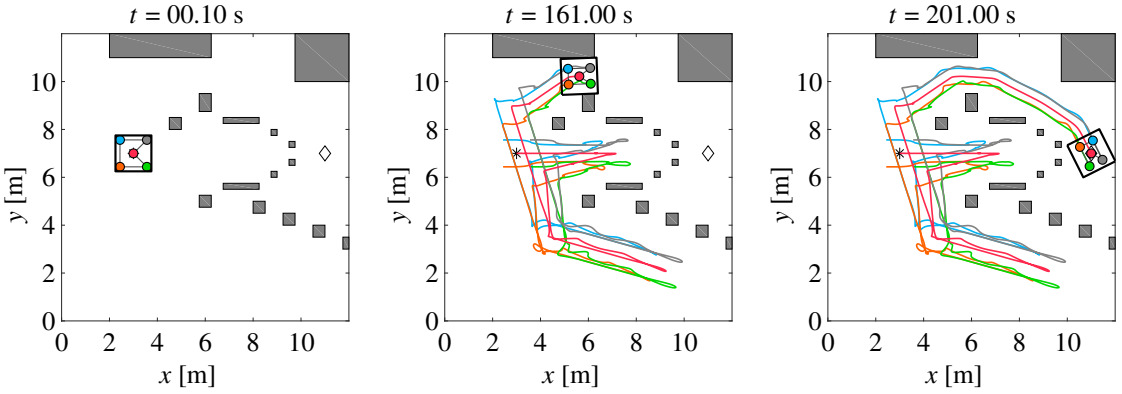

Figure 3: SDMPC results from the simulation of the plate transportation

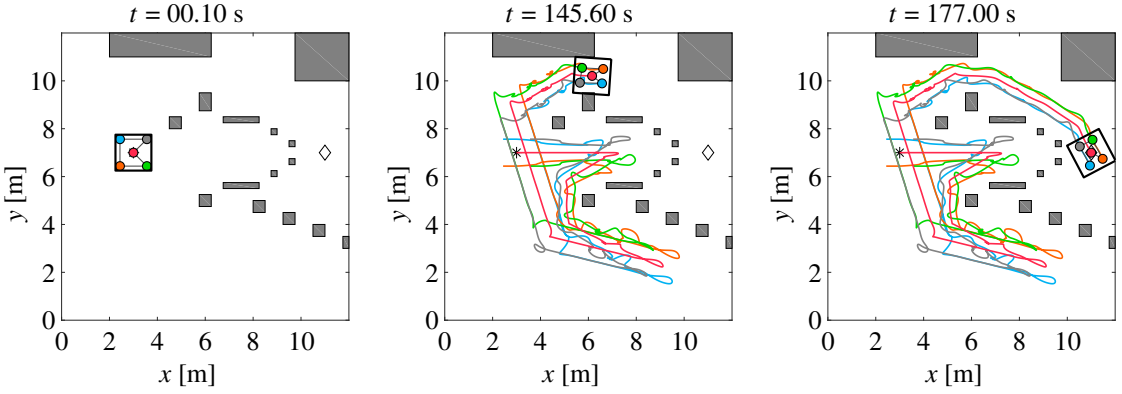

Figure 4: IDMPC results from the simulation of the plate transportation

4.3. Hardware Experiments

Since both schemes have proven to be quite promising in various simulations, of which, in the brevity of this paper, we unfortunately could only show a very small sample of, we currently are in the process of preparing an environment to test the control schemes in real-world hardware experiments. In this environment, to provide enough computational resources, the control schemes run on a rather powerful workstation computer, with the calculations for each robot running in its own respective Matlab process. The distance sensors are simulated. The Matlab processes, and thereby

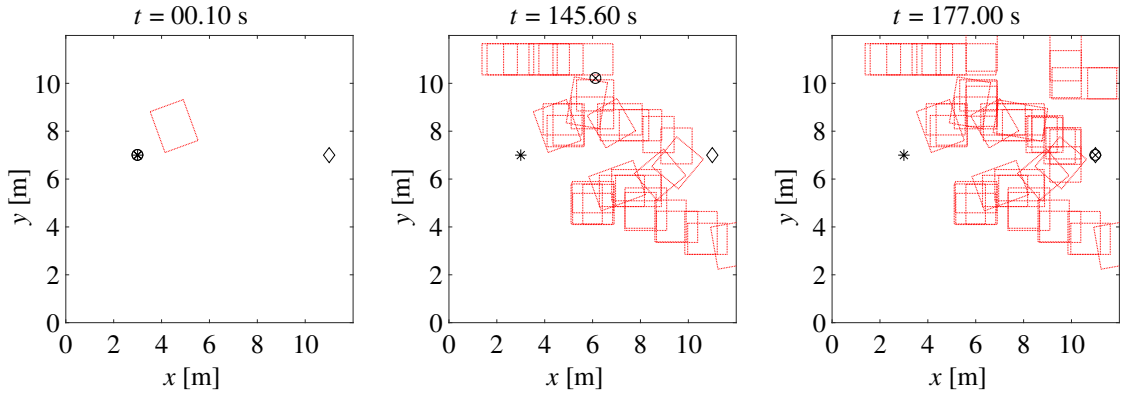

Figure 5: Plot of the constructed map at three time steps from the simulation with the IDMPC scheme – areas that shall not be entered by the center of mass are marked by red-bordered rectangles. The formation's current center of mass is marked by an encircled cross.

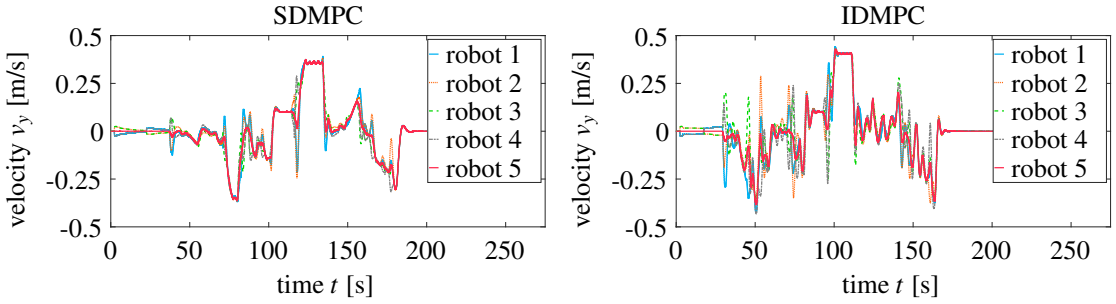

Figure 6: Plot of the robot velocities $v_y(t)$ in the y-direction for the two schemes. When compared with Figures 3 and 4, in their initial positions, the robots are numbered in counter-clockwise direction, beginning at the top left, with the robot in the middle being robot 5.

metaphorically the robots, communicate via the TCP/IP protocol by means of the Robotic Operating System (ROS) [11] as implemented in the Matlab Robotics System Toolbox. In each sampling instant, again via ROS, each process sends its control inputs to another computer that then passes the control inputs over a wireless network to a group of FESTO Robotinos [12, 13]. The same computer tracks the positions of all the robots by camera and sends the currently measured positions to the controlling processes via ROS. An exemplary result recorded using the IDMPC scheme with $n_{\max} = 1$ is provided in Figure 7. The results show that the robots successfully assume the desired formation shape, then shrink the formation to pass through a narrow passage and finally safely reach the goal in the original size of the formation. While the described computational environment is set up more to the benefit of a convenient development environment and not at all tailored to maximum efficiency, we have been able to maintain usable sampling times. In the results shown in Figure 7, an MPC sampling time of 0.2 s is used. This has proven to be low enough for the present scenario. At the same time, in our experience, for modest values of $n_{\max}$, the IDMPC scheme allows for shorter sampling times than SDMPC due to its concurrent optimization within one iteration. More elaborate hardware experiments also involving the actual transportation of a plate are currently in preparation.

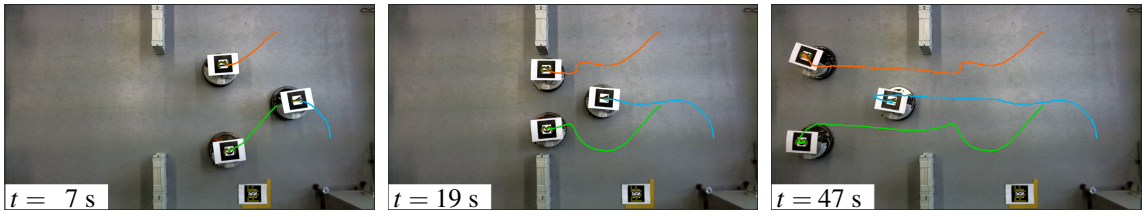

Figure 7: Experimental results obtained with the IDMPC scheme. The reference path for the center of mass was defined a priori. The robots have a diameter of approximately 0.37 m. The open-source ARToolKit is used for localization.

5. Conclusion

In this paper, two quite different formation control approaches for mobile robots based on distributed model predictive control have been presented and discussed. The choice of model predictive control, despite its comparatively significant computational complexity and cost, has been motivated by the task of transporting a plate purely by normal and friction forces, which makes a very well-controlled behavior a necessity. As shown by conclusive simulation results, despite their quite different theoretical setups, through their ability to explicitly constrain the control inputs, both schemes are capable of solving the formation control and plate transportation tasks in a satisfactory manner. This holds true even when navigating through very intricate environments and when altering the size of the formation to squeeze through narrow passages. Finally, the results from early hardware experiments suggest that, principally, the schemes are also implementable in real-time with real robot hardware and network-based information exchange.

Acknowledgments

This research was partially funded by the German Research Foundation via the Cluster of Excellence SimTech at the University of Stuttgart. This support is highly appreciated.

References

[1] R. Scattolini, "Architectures for distributed and hierarchical model predictive control - a review," *Journal of Process Control*, vol. 19, no. 5, pp. 723-731, 2009.

[2] F. Bullo, J. Cortés and S. Martínez, Distributed control of robotic networks, Princeton: Princeton University Press, 2009.

[3] J. Maciejowski, Predictive control with constraints, Harlow: Pearson Education, 2001.

[4] M. A. Müller, M. Reble and F. Allgöwer, "Cooperative control of dynamically decoupled systems via distributed model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 22, no. 12, pp. 1376-1397, 2012.

[5] A. N. Venkat, J. B. Rawlings and S. J. Wright, "Stability and optimality of distributed model predictive control," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 6680-6685, Seville, 2005.

[6] C. Makkar, W. E. Dixon, W. G. Sawyer and G. Hu, "A new continuously differentiable friction model for control systems design," in *Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 600-605, Monterey, 2005.

[7] R. W. Cottle and G. B. Dantzig, "Complementary pivot theory of mathematical programming," *Linear Algebra and Its Applications*, vol. 1, no. 1, pp. 103-125, 1968.

[8] A. N. Venkat, I. A. Hiskens, J. B. Rawlings and S. J. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1192–1206, 2008.

[9] B. T. Stewart, A. N. Venkat, J. B. Rawlings, S. J. Wright and G. Pannocchia, "Cooperative distributed model predictive control," *Systems & Control Letters*, vol. 59, no. 8, pp. 460-469, 2010.

[10] H. Ebel, E. Sharafian Ardakani and P. Eberhard, "Distributed model predictive formation control with discretization-free path planning for transporting a load," Submitted to *Robotics and Autonomous Systems*.

[11] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Y. Ng, "ROS: An open-source robot operating system," In *Proceedings of the Open-Source Software Workshop of the International Conference on Robotics and Automation (ICRA)*, Kobe, 2009.

[12] R.-C. Weber, Festo Robotino manual, Denkendorf: Festo Didactic GmbH & Co. KG, 2007.

[13] Q. Tang and P. Eberhard, "Cooperative search by combining simulated and real robots in a swarm under the view of multibody system dynamics," *Advances in Mechanical Engineering*, vol. 5, Article ID 284782, 11 pages, 2013.