Cooperative guidance of Lego Mindstorms NXT mobile robots

Julien Marzat, Hélène Piet-Lahanier, Arthur Kahn
ONERA – The French Aerospace Lab, F-91123 Palaiseau, France,
{julien.marzat,helene.piet-lahanier,arthur.kahn}@onera.fr

Keywords: autonomous robots, cooperative control, model predictive control, source localization

Abstract: This paper presents experimental results of cooperative guidance laws embedded on Lego Mindstorms NXT mobile robots for two types of missions. The first one is navigation to a waypoint as a fleet with collision and obstacle avoidance, following a model predictive control (MPC) framework. The second one is source localization, i.e., finding the maximum of a potential field, for which a distributed estimation and control strategy is proposed. Experiments show the ability to perform the two missions on these basic mobile robots, in spite of their limited computational resources. In particular, the search for the optimal control sequence through a dedicated discretization of the command space makes it possible to implement real-time MPC.

1 INTRODUCTION

Cooperation between autonomous mobile robots is a challenging task which is currently a very active research field. Several approaches have been suggested for designing decentralized control laws that allow each vehicle to follow a trajectory without affecting the performances of the other vehicles of the fleet, while achieving a required cooperative task [Murray, 2007]. An efficient method is to evaluate distributively a common criterion based on each vehicle action and measurements and the interaction between vehicles. The determination of the control laws can be be derived relying on approaches such as model predictive control (MPC) [Dunbar and Murray, 2006] with individual optimization.

This approach has been validated by simulation but requires experimental testing to ensure that it can be embedded on autonomous platforms. The chosen test system is the low-cost Lego Mindstorms NXT. This popular platform has already been used for basic control of a single robot [Costa et al., 2011, Valera et al., 2011] in particular for control education [Canale and Brunet, 2013] and for testing estimation or localization algorithms [Pinto et al., 2012]. A few attempts at cooperativeness have been made with these robots mainly for coordination and flocking [Benedettelli et al., 2009, Maze et al., 2012]. These works did not consider MPC as a potential solution for fleet coordination and collision avoidance, since it is usually regarded as computationally too expensive. However, our experiments show that it is possible to embed an MPC algorithm on robots with limited computing capacities if a suboptimal search procedure is adopted.

Two classical missions illustrate experimentation of embedded cooperative control and estimation on Lego Mindstorms NXT (see description of the robots in Section 2). The first one (Section 3) involves the navigation to waypoints by a fleet of robots, while avoiding collisions between agents and with obstacles. The second one (Section 4) consists in finding the localization of a source as the maximum of a potential field based on local measurements performed by the agents while moving in fleet.

2 LEGO MINDSTORMS NXT MOBILE ROBOTS

A fleet of N Mindstorms robots has been considered, all built according to a two-wheel differential structure (Figure 1). The dynamical model of the i-th vehicle is

$$\begin{cases} x_i(k+1) = x_i(k) + \Delta t\, v_i(k)\cos(\chi_i(k)) \\ y_i(k+1) = y_i(k) + \Delta t\, v_i(k)\sin(\chi_i(k)) \\ \chi_i(k+1) = \chi_i(k) + \Delta t\, u_i^{\omega}(k) \end{cases} \quad (1)$$

where $\mathbf{p}_i = [x_i, y_i]^T$ is the vehicle position and χ_i its direction angle, which form the state vector $\mathbf{x}_i = [x_i, y_i, \chi_i]^T$; Δt is the sampling timestep. The velocity v_i was set to a constant value for simplicity, the only control input is thus the rotational speed $\mathbf{u}_i = u_i^{\omega}$, constrained between $\pm\omega_{max}$.

For the i-th robot, practical control of linear and angular velocity via the controllable rotation speeds of the wheels ω_i^l and ω_i^r is achieved by

$$\begin{cases} v_i = \frac{(\omega_i^l + \omega_i^r)}{2} r \\ u_i^\omega = \frac{(\omega_i^r - \omega_i^l)}{2L} r \end{cases} \quad (2)$$

where r is the wheel radius and L the half-axis length. Localization of each robot is performed using odometry based upon the previous model (embedded wheel sensors have an accuracy of 1 degree). Since the test missions were short, this localization method was deemed sufficient for estimating the position of the robots in spite of the error accumulated by odometry.

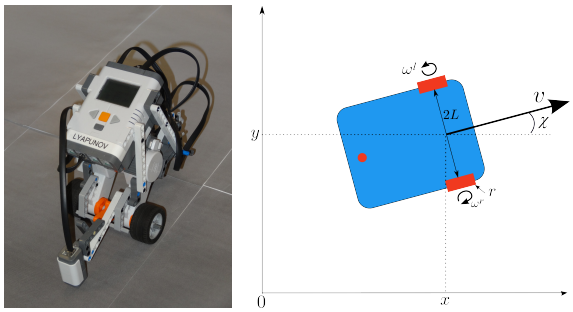

Figure 1: Lego Mindstorms NXT robot

The Lego Mindstorms NXT embedded computational capabilities are provided by a main ARM CPU clocked at 48Mhz assisted by a co-processor ATmega clocked at 20Mhz, both communicating with 64kB of DRAM and a storage flash memory of 256kB. Estimation and control algorithms should thus be designed accordingly in terms of number of operations and data stored to be able to run on this architecture. It can be programmed in many languages, here NXC ("not exactly C") was chosen for its simplicity and ability to control all the parameters and elements of the robot. Bluetooth communication was used to allow robots to share their estimated positions and possibly other information with the rest of the fleet. A major practical constraint of the Mindstorms system is that the communication graph requires a single master robot with a maximum of three slaves, which limits the fleet to four robots. An exchange rate larger than 20Hz was recorded in practice for sharing position information between 2 robots.

3 FLEET NAVIGATION WITH COLLISION AND OBSTACLE AVOIDANCE BY MPC

The first mission considered is the navigation of a fleet of Mindstorms toward a waypoint, with collision and obstacle avoidance. This is tackled using a simplified version of the MPC framework initially defined in [Rochefort et al., 2012], where more theoretical details can be found.

3.1 MPC method

In distributed MPC [Scattolini, 2009], each vehicle computes its control inputs at each timestep as a solution of an optimization problem over the future predicted trajectory. For tractability reasons, finite prediction and control horizon lengths, respectively denoted H_p and H_c are used. Future control inputs and resulting state trajectories are

$$\mathbf{U_i} = ((\mathbf{u}_i(k))^T, (\mathbf{u}_i(k+1))^T, ..., (\mathbf{u}_i(k+H_c-1))^T)^T$$
$$\mathbf{X_i} = ((\mathbf{x}_i(k+1))^T, (\mathbf{x}_i(k+2))^T, ..., (\mathbf{x}_i(k+H_p))^T)^T$$

When $H_c < H_p$, we assume that the control inputs are null after H_c steps. Once the optimal input sequence $\mathbf{U_i}^\star$ has been computed, each vehicle communicates its predicted trajectory to the rest of the fleet and applies the first entry $\mathbf{u}_i^\star(k)$. The optimization problem at time k is stated as

$$\begin{aligned} &\text{minimize } J_i(\mathbf{U_i}, \mathbf{X_i}) \\ &\text{over } \mathbf{U_i} \in \mathcal{U}_i^{H_c} \\ &\text{with } \mathbf{x}_i(t) \text{ satisfying } (1),\ \forall t \in [k+1; k+H_p] \end{aligned} \quad (3)$$

J_i is the cost function associated with vehicle i. The constraints coupling the dynamics of the vehicles, such as collision avoidance, are taken into account by penalization. At the next timestep, each vehicle solves its optimization problem considering that the other vehicles follow their predicted trajectories. J_i comprises a navigation cost J_i^{nav}, a safety cost J_i^{safety} and a control cost J_i^u such that

$$J_i(k) = J_i^{nav}(k) + J_i^{safety}(k) + J_i^u(k) \quad (4)$$

Formulation of each cost function is presented in the following subsections. Weighting coefficients $W^\bullet$ are tuned to set relative priorities between each aspect of the mission.

3.1.1 Navigation cost

The navigation cost J_i^{nav} aims at controlling how vehicles navigate to waypoints. It is decomposed into

$$J_i^{nav}(k) = J_i^{nav,direct}(k) + J_i^{nav,fleet}(k) \quad (5)$$

The reference trajectory to the next waypoint $\mathbf{p}_p$ is composed of points $\mathbf{p}_{i,p}^{ref}(n|k)$ $(n \in [k+1, k+H_p])$. They correspond to positions that vehicle i would reach at timestep n if moving along a straight line to

$\mathbf{p}_p$ at nominal velocity v_i. These reference points are thus defined by (6) and the associated cost $J_i^{nav,direct}$ is given by (7). The notation $\widehat{\mathbf{p}}_i(n|k)$ represents the predicted position of robot i at time n, starting from instant k.

$$\mathbf{p}_{i,p}^{ref}(n|k) = \mathbf{p}_i(k) + (n-k)\Delta t v_i \frac{\mathbf{p}_i(k) - \mathbf{p}_p}{\left\|\mathbf{p}_i(k) - \mathbf{p}_p\right\|} \quad (6)$$

$$J_i^{nav,direct}(k) = W^{nd} \sum_{n=k+1}^{k+H_p} \left\| \widehat{\mathbf{p}}_i(n|k) - \mathbf{p}_{i,p}^{ref}(n|k) \right\|^2 \quad (7)$$

$J_i^{nav,fleet}$ aims at keeping the vehicles together as a fleet. Its definition penalizes the predicted distance $\widehat{d_{ij}}(n|k) = \left\|\widehat{\mathbf{p}}_j(n|k) - \widehat{\mathbf{p}}_i(n|k)\right\|$ between vehicles i and j ($i \neq j$), as

$$J_i^{nav,fleet}(k) = W^{nv} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{n=k+1}^{k+H_p} \frac{1 + \tanh\left(\left(\widehat{d_{ij}}(n|k) - \beta_{ij}^f\right)\alpha_{ij}^f\right)}{2} \quad (8)$$

where coefficients β_{ij}^f and α_{ij}^f are defined by

$$\beta_{ij}^f = 6\left(d_{\text{loss}}^v - d_{\text{des}}^v\right)^{-1}, \quad \alpha_{ij}^f = \frac{1}{2}\left(d_{\text{loss}}^v + d_{\text{des}}^v\right) \quad (9)$$

The coefficient d_{des}^v defines a desired distance between the vehicles inside the fleet whereas d_{loss}^v is the maximum distance allowed between vehicles of the fleet. Vehicles j ($i \neq j$) beyond this maximum distance are not considered any more by vehicle i. This represents for example limited communication and/or sensing ranges.

3.1.2 Safety cost

The safety cost J_i^{safety} aims at avoiding collisions with obstacles, and between vehicles within the fleet. It is made of two costs

$$J_i^{safety}(k) = J_i^{safe,veh}(k) + J_i^{safe,obs}(k) \quad (10)$$

The first cost deals with collision avoidance between vehicles by penalizing the predicted distance d_{ij} between them:

$$J_i^{safe,veh}(k) = W^{sv} \sum_{\substack{j=1 \\ j\neq i}}^{N} \sum_{n=k+1}^{k+H_p} \frac{1 - \tanh\left(\left(\widehat{d_{ij}}(n) - \beta_{ij}^v\right)\alpha_{ij}^v\right)}{2} \quad (11)$$

where

$$\beta_{ij}^v = 6\left(d_{\text{des}}^v - d_{\text{safe}}^v\right)^{-1}, \quad \alpha_{ij}^v = \frac{1}{2}\left(d_{\text{des}}^v + d_{\text{safe}}^v\right) \quad (12)$$

where d_{safe}^v represents a safety distance between the vehicles. The costs $J_i^{safe,veh}$ and $J_i^{nav,fleet}$ with respect to the distance d_{ij} are plotted in Figure 2.

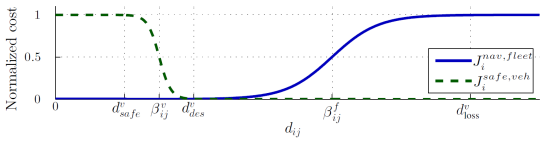

Figure 2: Flocking ($J_i^{nav,fleet}$) and avoidance ($J_i^{safe,veh}$) costs

The other cost $J_i^{safe,obs}$ penalizes the predicted distance $\widehat{d_{io}}$ of vehicle i to any obstacle o. It is defined as

$$J_i^{safe,obs}(k) = W^{so} \sum_{o=1}^{N^o} \sum_{n=k+1}^{k+H_p} \frac{1 - \tanh\left(\left(\widehat{d_{io}}(n|k) - \beta_{io}^o\right)\alpha_{io}^o\right)}{2} \quad (13)$$

where N^o stands for the number of obstacles and the parameters β_{io}^o and α_{io}^o are given by

$$\beta_{io}^o = 6\left(d_{\text{des}}^o - d_{\text{safe}}^o\right)^{-1}, \quad \alpha_{io}^o = \frac{1}{2}\left(d_{\text{des}}^o + d_{\text{safe}}^o\right) \quad (14)$$

where d_{des}^o and d_{safe}^o are desired and safety distances to obstacles. Locations of the obstacles are assumed to be known in the experiments, but could also be detected with infrared or ultrasonic Lego sensors.

3.1.3 Control cost

As traditionally introduced in MPC, the control cost $J_i^u(k)$ aims at limiting the control effort, and thus energy consumption of vehicle i. It is simply defined as

$$J_i^u(k) = W^{u,\omega} \sum_{n=k+1}^{k+H_c} \left(u_i^\omega(n)\right)^2 \quad (15)$$

3.1.4 Online computation of control inputs

As the computational cost should be reduced to cope with the robot resources, we limit the search to a finite set $\mathcal{S}$ of candidate control sequences. MPC guarantees stability even if the resulting sequences are not optimal but ensures the steadily decrease of the cost, as shown in [Scokaert et al., 1999]. At each timestep, the control problem (3) is solved as follows:

1. using a model of the vehicle dynamics, predict the effect of each control sequence of the set of candidates $\mathcal{S}$ on the state of the vehicle;
2. compute the cost J_i corresponding to each remaining candidate control sequence;
3. select the control sequence with smallest cost.

The distribution of the candidate control sequences is chosen so as to limit their number while providing a good coverage of the control space, according to the following three rules:

1. the set $\mathcal{S}$ of candidates includes the extreme control inputs to exploit the full vehicle potential;
2. the set $\mathcal{S}$ of candidates includes the null control input to allow to continue along the same path;
3. candidates are distributed over the entire control space with an increased density around the null control input.

The set of control inputs for the problem at hand reduces to the discretization, with a varying step γ,

$$\mathcal{S} = \left\{ \frac{2\pi\gamma}{\eta^{\omega}} \right\} \qquad \text{with } \gamma \in [1, \eta^{\omega}], \tag{16}$$

where η^{ω} should be chosen to keep the computation of predicted trajectories within the duration of a timestep.

3.2 Experimental results

The chosen discretization strategy to select the optimal control input makes it possible to embed the MPC guidance laws on such mobile robots with limited capabilities, which is otherwise not feasible. The parameters for the experiments presented here were $v_i = 0.1$ m/s, $\omega_{max} = 2.5$ rad/s (consistent with the actual motor limitations), $\Delta t = 0.3$ s, $H_c = 4$, $H_p = 8$ and $\eta^{\omega} = 11$. The computation time achieved at each iteration was constant and comfortably smaller than Δt.

Collision avoidance by MPC To validate the communication capabilities of the vehicles and the safety cost ensuring collision avoidance, the first scenario required each robot to go to the initial position of the other robot while avoiding it on the way. The global cost function was thus

$$J_i^1 = J_i^{nav,direct} + J_i^{safe,veh} + J_i^u \tag{17}$$

where the navigation cost $J_i^{nav,direct}$ for robot i steered it to reach the initial position $\mathbf{p}_j$ of robot j. The obtained trajectories are given in Figure 3, where it can be observed that the two robots reach their destination with good accuracy and collision avoidance is effective with the desired distance ($d^v_{\text{des}} = 0.5$ m here).

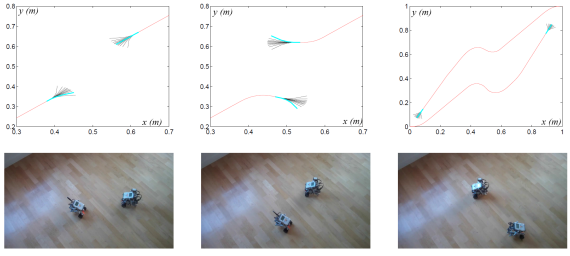

Figure 3: Collision avoidance trajectory

Safe fleet navigation by MPC In addition to the two costs tested in the first experiment, the second scenario involved a fleet behavior (cost $J_i^{nav,fleet}$) with a desired distance between the vehicles equal to 0.2 m and an obstacle (whose position is known) to avoid thanks to the cost $J_i^{safe,obs}$. For this experiment, the global cost function was

$$J_i^2 = J_i^{nav,direct} + J_i^{nav,fleet} + J_i^{safe,veh} + J_i^{safe,obs} + J_i^u \tag{18}$$

where now both vehicles were required to go to the same waypoint as a fleet. The obtained trajectories (Figure 4) illustrate the fleet behavior of the vehicles (distance $d^v_{\text{des}} = 0.2$ m is respected) before they encounter the obstacle and avoid it (with a safety distance $d^v_{\text{safe}} = 0.1$ m) and finally head toward the same waypoint.

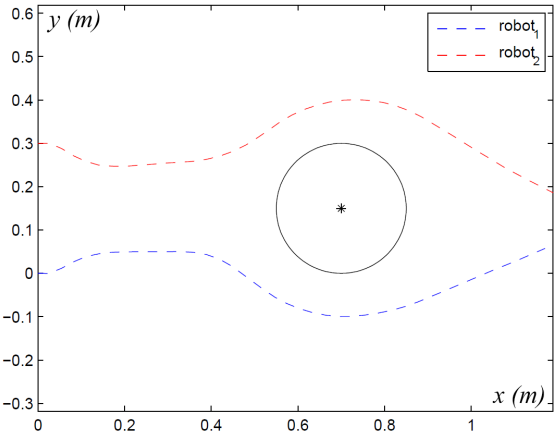

Figure 4: Fleet navigation with obstacle avoidance (motion from left to right)

4 SOURCE LOCALIZATION

The second scenario is the localization of the maximum of a potential field ϕ using the distributed measurements acquired by the robots of the fleet. This is a problem where cooperative distributed estimation is necessary, since a single robot would have difficulties to localize the source with a single sensor (information would be gathered only on its own path).

The MPC costs obtained for Scenario 1 are still valid for fleet management and collision or obstacle avoidance, but the navigation cost should now be replaced by the control described in Section 4.2.

4.1 Distributed estimation

The field ϕ, function of position $\mathbf{p}_i$, is assumed to be time-invariant and concave (maximum is unique,

second-order derivative $\nabla^2\phi$ is negative everywhere). The source-localization algorithm consists in estimating locally the spatial gradient $\nabla\phi$ of the field ϕ using the fleet of robots, each of the vehicle being able to measure (with identical sensors) the field value at its current position and broadcast it to the rest of the fleet. The fleet of vehicles will then move along the direction of the estimated gradient to head toward the maximum. For estimating the gradient, a vector $\mathbf{y}$ of m measurements is considered

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \tag{19}$$

where $y_j = \phi(\mathbf{p}_j) + e_j$ is the measurement of the field ϕ gathered at a vehicle position $\mathbf{p}_j$ and e_j is the measurement noise assumed to be distributed according to a zero-mean Gaussian distribution of variance σ_e^2. The different measurements y_j $(j = 1,...,m)$ are acquired by the robots of the fleet and can possibly be measurements at successive time steps. For instance, if m is chosen to be greater than the number of robots N, a design choice should be made to use past measurements of the robots associated to successive positions (with a timestep large enough to avoid ill-conditioning).

The actual function ϕ is unknown, so a linear estimation model is considered around a given position $\mathbf{p}_0 = [x_0, y_0]^T$ (corresponding to a virtual position within the fleet) as

$$\phi_l(\mathbf{p}) = \phi(\mathbf{p}_0) + [x - x_0 \; y - y_0][\nabla\phi_{x0} \; \nabla\phi_{y0}]^T, \tag{20}$$

which can be written as

$$\phi_l(\mathbf{p}) = [1 \; x - x_0 \; y - y_0]\beta, \tag{21}$$

where $\beta = [\phi(\mathbf{p}_0), \; \nabla\phi_{x0}, \; \nabla\phi_{y0}]^T$ is an unknown vector of parameters to be estimated using the measurement vector $\mathbf{y}$ and the corresponding positions $\mathbf{p}_j$ $(j = 1,...,m)$. The model with the m measurements can then be written as $\mathbf{y} = \mathbf{H}\beta$, where

$$\mathbf{H} = \begin{bmatrix} 1 & x_1 - x_0 & y_1 - y_0 \\ \vdots & \vdots & \vdots \\ 1 & x_m - x_0 & y_m - y_0 \end{bmatrix} \tag{22}$$

The least-square estimate of β is

$$\widehat{\beta} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y} \tag{23}$$

and the estimated gradient $\widehat{\nabla}\phi = [\widehat{\nabla}\phi_{x0}, \; \widehat{\nabla}\phi_{y0}]^T$ can be used as the steepest climbing direction to guide the fleet toward the unique maximum of the potential field. Note that with the assumed linear model, the estimated gradient $\widehat{\nabla}\phi$ is independent of the reference point $\mathbf{p}_0$ where it is computed.

4.2 Fleet control for maximum seeking

The fleet should then be guided toward the maximum by following the direction indicated by the estimated gradient. In a distributed scheme, each vehicle estimates the same gradient value using the measurements of all the fleet, then aligns its velocity on this direction. A speed control is considered as $\dot{\mathbf{p}}_i = \nabla\phi\|\nabla\phi\|^{-1}v_i$ with $v_i > 0$, assuming that the estimation error $\widehat{\nabla}\phi - \nabla\phi$ is small enough. It could then be proven that the robots will converge to the position of the unique maximum $\mathbf{p}^*$ by considering the Lyapunov function

$$V = (\mathbf{p}^* - \mathbf{p}_i)^T(\mathbf{p}^* - \mathbf{p}_i), \tag{24}$$

whose derivative along the system trajectory is

$$\dot{V} = -(\mathbf{p}^* - \mathbf{p}_i)^T\nabla\phi\|\nabla\phi\|^{-1}v_i. \tag{25}$$

The Taylor expansion of the actual field ϕ at position $\mathbf{p}_i$, evaluated at the maximum position $\mathbf{p}^*$ yields

$$\phi(\mathbf{p}^*) = \phi(\mathbf{p}_i) + (\mathbf{p}^* - \mathbf{p}_i)^T\left(\nabla\phi + \nabla^2\phi(\xi)(\mathbf{p}^* - \mathbf{p}_i)\right), \tag{26}$$

where ξ belongs to the 2D interval $[\mathbf{p}_i, \mathbf{p}^*]$. Then

$$\begin{aligned} \dot{V} &= (\mathbf{p}_i - \mathbf{p}^*)^T\nabla\phi\|\nabla\phi\|^{-1}v_i \\ &= (\phi(\mathbf{p}_i) - \phi(\mathbf{p}^*))\,\|\nabla\phi\|^{-1}v_i \\ &\quad +(\mathbf{p}_i - \mathbf{p}^*)^T\nabla^2\phi(\xi)(\mathbf{p}_i - \mathbf{p}^*)\|\nabla\phi\|^{-1}v_i. \end{aligned} \tag{27}$$

Since the field is concave, $\nabla^2\phi < 0$ everywhere and since $\mathbf{p}^*$ is the maximum location, $\phi(\mathbf{p}_i) - \phi(\mathbf{p}^*) \leq 0$, v_i is chosen positive, therefore $\dot{V} < 0$ and the gradient climbing control makes the vehicles converge toward the maximum of the field.

4.3 Experimental results

A grayscale map is considered as a potential field for the experiments, the maximum being located at the darkest spot. In addition to their positions, the robots now also share their measured gray level (using the Lego color sensor). A fleet of three robots is considered and the mission stops when the measurement of one of the robots reaches a predefined target value (another possible stopping condition could be a threshold on the gradient norm). For estimation purpose, 3 points are sufficient to estimate the gradient, 6 can also be considered for better redundancy by taking into account past positions and measurements of the robots. The NXT brick needs 10 ms to invert a 3x3 matrix and 80 ms to solve the least-square estimation problem (23) with 6 measurements, which is compatible with the chosen timestep of 300 ms. Figure 5 illustrates the gradient ascent in the joint space of positions and gray level (mapped between 0 and 1 from lighter to darker). The fleet successfully finds the location of the maximum.

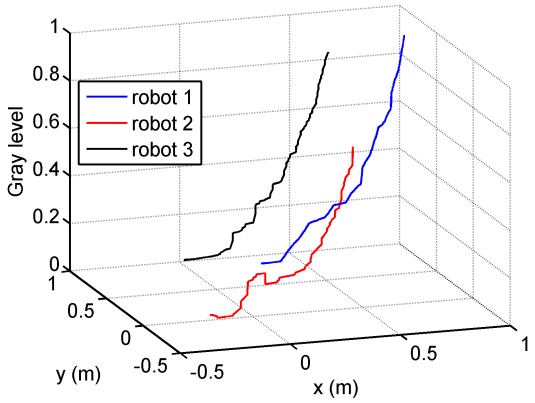

Figure 5: Fleet trajectory toward maximum, with measured potential level

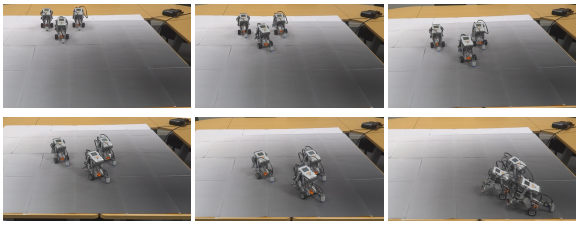

Figure 6: Sequence of fleet convergence to the maximum

5 CONCLUSIONS AND PERSPECTIVES

The experiments reported in this paper have highlighted the possibility to control a fleet of Lego Mindstorms NXT to fulfill two types of missions with these autonomous vehicles, notwithstanding their computational capabilities. A first scenario has shown that MPC can be a flexible solution to deal with fleet management and collision avoidance between vehicles and with obstacles. An efficient discretization strategy allows the MPC to find an efficient control sequence within constrained time. In a second scenario, a decentralized estimation and control scheme to find the maximum of a potential field has been presented. It involved linear parameter estimation to obtain the gradient of the field and a gradient-ascent control proven to converge to the actual maximum location. Implementation of the two strategies on the fleet of Lego Mindstorms NXT was successful, which shows the interest of these platforms as a practical testbed for cooperative estimation and control under strict implementation constraints.

ACKNOWLEDGEMENTS

The authors would like to thank Guillaume Broussin and Mathieu Touchard, who contributed to these experiments during their internship at ONERA.

REFERENCES

Benedettelli, D., Casini, M., Garulli, A., Giannitrapani, A., and Vicino, A. (2009). A Lego Mindstorms experimental setup for multi-agent systems. In *Proceedings of the IEEE Multi-conference on Systems and Control, Saint Petersburg, Russia*, pages 1230–1235.

Canale, M. and Brunet, S. C. (2013). A Lego Mindstorms NXT experiment for model predictive control education. In *Proceedings of the European Control Conference, Zurich, Switzerland*, pages 2549–2554.

Costa, P., Moreira, A., Gonçalves, J., and Lima, J. (2011). Proposal of a new real-time cooperative challenge in mobile robotics. In *Proceedings of the 18th IFAC World Congress, Milan, Italy*.

Dunbar, W. and Murray, R. (2006). Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42:549–558.

Maze, N., Wan, Y., Namuduri, K., and Varanasi, M. (2012). A Lego Mindstorms NXT-based test bench for cohesive distributed multi-agent exploratory systems: Mobility and coordination. In *Proceedings of the AIAA Infotech@Aerospace, Garden Crove, California*.

Murray, R. (2007). Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, 129(5):571–583.

Pinto, M., Moreira, A. P., and Matos, A. (2012). Localization of mobile robots using an extended Kalman filter in a Lego NXT. *IEEE Transactions on Education*, 55(1):135–144.

Rochefort, Y., Bertrand, S., Piet-Lahanier, H., Beauvois, D., and Dumur, D. (2012). Cooperative nonlinear model predictive control for flocks of vehicules. In *Proceedings of the IFAC Workshop on Embedded Guidance, Navigation and Control in Aerospace, Bangalore, India*.

Scattolini, R. (2009). Architectures for distributed and hierarchical model predictive control–a review. *Journal of Process Control*, 19(5):723–731.

Scokaert, P. O. M., Mayne, D. Q., and Rawlings, J. B. (1999). Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control,*, 44(3):648–654.

Valera, Á., Vallés, M., Marín, L., and Albertos, P. (2011). Design and implementation of Kalman filters applied to Lego NXT based robots. In *Proceedings of the 18th IFAC World Congress, Milan, Italy*.