2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Real-Time Distributed Receding Horizon Motion Planning and Control for Mobile Multi-Robot Dynamic Systems

José M. Mendes Filho[a, b, *], Eric Lucet[a] and David Filliat[b]

***Abstract*— This paper proposes an improvement of a motion planning approach and a modified model predictive control (MPC) for solving the navigation problem of a team of dynamical wheeled mobile robots in the presence of obstacles in a realistic environment. Planning is performed by a distributed receding horizon algorithm where constrained optimization problems are numerically solved for each prediction time-horizon. This approach allows distributed motion planning for a multi-robot system with asynchronous communication while avoiding collisions and minimizing the travel time of each robot. However, the robots dynamics prevents the planned motion to be applied directly to the robots. Using unicycle-like vehicles in a dynamic simulation, we show that deviations from the planned motion caused by the robots dynamics can be overcome by modifying the optimization problem underlying the planning algorithm and by adding an MPC for trajectory tracking. Results also indicate that this approach can be used in systems subjected to real-time constraint.**

I. Introduction

The capability of defining and executing a collision-free motion plan for passing from one configuration to another is a crucial aspect of robotics that can be specially difficult to solve for mobile multi-robot systems. A trending application that requires this capability is the use of robotic systems in industrial supply chains for processing orders and optimizing the storage and distribution of products. For example, Amazon employs the Kiva mobile-robot system, and IDEA Groupe employs the Scallog system for autonomously processing client orders [1], [2]. Such logistics tasks became increasingly complex as sources of uncertainty, such as human presence, are admitted in the work environment.

For ideally solving this navigation problem, different constraints must be taken into account in the motion planning; in particular geometric, kinematic and dynamic constraints. The first kind comes from the need of preventing some specific robot's configurations in order to avoid collisions, communication loss, etc. Kinematic constraints derive directly from the mobile robot architecture implying typically nonholonomic constraints. Dynamic constraints come mainly from inertial effects and interaction between different bodies in contact. For instance, those constraints may translate into accelerations not exceeding maximum values given that forces and torques are limited in a real system. Motion planning in presence of such constraints is usually referred as kinodynamic planning [3].

The importance of incorporating dynamics in motion planning dates back to the well-known Dynamic Window approach (DW) [4]. More recent work on navigation for teams of robots that uses DW can be found in [5]. In this work, real-time navigation for high speed robots respecting dynamic constraints is solved in three separated stages; an ERRT (Execution Extended Rapidly-Exploring Random Tree) path planner, a motion control layer and a velocity space safety search based on DW. Although decentralization of that approach is allegedly possible, only its centralized implementation was tested. Other works such as [6] treat kinodynamic planning in real-time in a dynamic environment but does not directly integrate multi-robot constraints. Authors of [7] propose a distributed motion planning approach that uses D* search algorithm in two separated layers of path and velocity planning with optimization at local and global levels but it is based on strong assumptions such as robots moving at constant speeds and being able to instantaneously halt.

Another work that relate to ours is the one presented in [8] on noncooperative distributed nonlinear MPC (disNMPC) and extended in [9] to work without stabilizing terminal constraints. Their disNMPC is similar to our Distributed Receding Horizon Motion Planning (DRHMP) scheme, but the underlying optimization problems and the use of the optimized solutions in the overall problem are very different.

The work presented here builds directly on the mathematical programming approach presented in [10], called DRHMP. The aim is to extend that approach so it can take into account dynamics constraints.

By means of a physics engine that can simulate rigid body dynamics (including collision detection), the approach is tested, evaluated and improved in two main aspects: i) feasibility of planned trajectories is improved by changing the optimization problems underlying the planning algorithm; ii) a model predictive control (MPC) that profits from this particular motion planning is conceived for minimizing deviations from the planned motion.

The paper's outline is as follows. The second section gives an overview of the DRHMP. The third section explains the two improvements listed above. The forth section is dedicated to the performance results obtained through simulation. The proposed MPC is compared to other controllers and measurements of elapsed time for planning and control routines are performed. Finally, in the last

[a] CEA, LIST, Interactive Robotics Laboratory, Gif-sur-Yvette, F-91191, France
[b] U2IS, ENSTA ParisTech, Inria FLOWERS team, Université Paris-Saclay, 828 bd des Maréchaux, 91762 Palaiseau cedex France
[*] corresponding author: jose.mendesfilho@cea.fr

section, we present our conclusions and perspectives.

II. DRHMP

As a team of robots evolves in their work environment, they progressively perceive new obstacles in their way to their goal configuration. Thus, trying to plan for the whole motion from initial to goal configurations is not a satisfying approach. Planning locally and replanning is more suitable for taking new information into account as it comes.

In the DRHMP in [10], each robot in the team computes its own local trajectory. Analogous to an MPC, a prediction time-horizon T_p and an implementation/computation timeslot T_c are defined (Fig. 1).

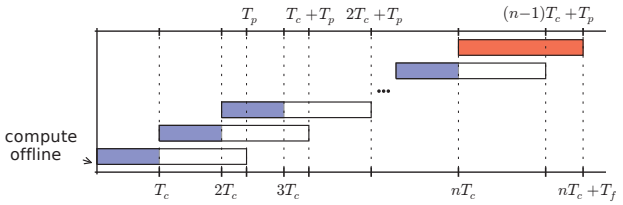


Fig. 1: Receding horizon scheme with termination plan. T_f is both prediction horizon and implementation timeslot for the termination plan in red.

T_p is the time-horizon for which a local solution to the motion problem will be created and T_c is the timeslot during which a portion of that solution is implemented while the next plan - created for the next time-horizon T_p - is being computed. The problem of producing a trajectory for a T_p horizon during T_c time is called here a receding horizon planning problem. It differs from the classical definition of MPC since not only the first value of the optimized solution is used for computing the system's input.

Moreover, for each receding horizon planning problem, the following steps are performed:

Step 1: Each robot in the team computes its own intended solution trajectory (denoted $(\hat{q}(t), \dot{\hat{q}}(t), \ddot{\hat{q}}(t))$[1] with q the configuration vector of the robot) by solving a constrained optimization problem. In that optimization problem all constraints are included except coupling constraints, that is, constraints that involve solving a conflict between multiple robots such as collision or loss of communication.

Step 2: Robots involved in a potential conflict (risk of collision, lost of communication) update their trajectories computed during Step 1 by solving a second constrained optimization problem that additionally takes into account geometric constraints for avoiding conflicts with other robots. This is done by using estimates of the intended trajectories of the other robots. If a robot is not involved in any conflict, Step 2 is not executed and its final solution trajectory is identical to the one found at Step 1.

[1]higher order derivatives of $\hat{q}(t)$ are guaranteed to exist by the choice of trajectory representation. B-splines are used for that purpose with its degree set to 3.

Differently from [10], [11], we do not consider that all robots involved in a conflict have finished Step 1 and exchanged information when any of them starts executing Step 2. Here, the robot estimates trajectories for the conflictual robots based on the available information at the end of Step 1. Those estimates allow asynchronous communication between robots: they can use different T_p and T_c and no defined frequency for their communication is imposed. Planning proceeds regardless of the communication frequency.

For each of these steps and for each robot in the team, one constrained optimization problem is solved. The cost function to be minimized in those optimization problems is the geodesic distance of a robot's current position to its goal position. This assures that the robots are driven towards theirs goals. However, when a robot arrives closer to its goal the receding horizon planning scheme does not produce the desired effect. For instance, near the goal, the robot can possibly take less time to reach it than the fixed T_p prediction horizon used for computing previous trajectories.

In [10] a termination procedure for reaching the goal is proposed. It takes the goal configuration as a hard constraint in the optimization problem and uses the time for reaching the goal as the cost function to be minimized. Fig. 1 illustrates that change in the planning process by introducing the time-horizon T_f.

Fig. 2 illustrates the overall result of this approach generated in simulation. Two robots avoid mutual collision while, simultaneously, avoiding collision to obstacles, aiming for their goals and respecting nonholonomic constraints.

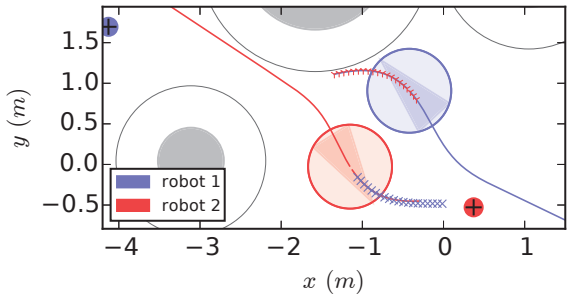


Fig. 2: Obstacles are represented by the grey regions. Blue X's over the red robot and red Y's over the blue robot represent their mutual trajectories estimates.

Trajectories generated with this approach are feasible regarding kinematics but not necessarily regarding the robots dynamics. In order to improve the quality of trajectories and assure its execution, we propose in the next section a change in the planning method and the addition of a controller.

III. Improvements for dealing with dynamics

Maximum acceleration constraints were added to all optimization problems during motion planning. This reduced tracking errors when directly applying planned trajectories to the simulated vehicles.

Besides improving the motion planner, a further step was to include a control law to minimize the deviation between

planned and executed trajectories. We modified an MPC introduced in [12] so it could take advantage of our particular motion planner.

A. Adding acceleration constraints

Maximum acceleration constraints were stated as follows:

$$\|[\ddot{x}\ \ddot{y}]\|^2 \leq a_{\max}^2 \tag{1}$$

$$\ddot{\psi}^2 \leq \alpha_{\max}^2 \tag{2}$$

$[x\ y]$ and ψ being the 2D position and orientation of the robot in the xy plane respectively, and $a_{\max}$ and $\alpha_{\max}$ the linear and angular maximum accelerations performed by the simulated robots. This way, the resulting receding horizon optimization problem solved at Step 2 by each vehicle can be written as follows:

$$\min_{q(t)} \|q(\tau_k + T_p) - q_{\text{goal}}\|, \quad \forall t \in [\tau_k, \tau_k + T_p] \tag{3}$$

$$\begin{cases} \dot{q}(t) = f(q(t), u(t)), & \forall t \in [\tau_k, \tau_k + T_p] \\ q(\tau_k) = q(\tau_{k-1} + T_c) & \\ \dot{q}(\tau_k) = \dot{q}(\tau_{k-1} + T_c) & \\ v^2(t) \leq v_{\max}^2, & \forall t \in (\tau_k, \tau_k + T_p] \\ \omega^2(t) \leq \omega_{\max}^2, & \forall t \in (\tau_k, \tau_k + T_p] \\ a^2(t) \leq a_{\max}^2, & \forall t \in [\tau_k, \tau_k + T_p] \\ \alpha^2(t) \leq \alpha_{\max}^2, & \forall t \in [\tau_k, \tau_k + T_p] \\ \mathrm{d}(O,t) \geq \varepsilon_o, & \forall t \in (\tau_k, \tau_k + T_p],\ \forall O \in \mathscr{O} \\ \mathrm{d}(R_c,t) \geq \varepsilon_r, & \forall t \in (\tau_k, \tau_k + T_p],\ \forall R_c \in \mathscr{C} \\ \mathrm{d}(R_d,t) \leq \min(d_{\text{com}}, d_{d,\text{com}}), & \forall t \in (\tau_k, \tau_k + T_p],\ \forall R_d \in \mathscr{D} \end{cases}$$

with $v = \|[\dot{x}\ \dot{y}]\|$, $\omega = \dot{\psi}$, $a = \|[\ddot{x}\ \ddot{y}]\|$, $\alpha = \ddot{\psi}$, $\mathrm{d}(\cdot,t)$ the distance from the robot to $\cdot$ at the moment t, $\mathscr{O}$ the set of the robot's known obstacles, $\mathscr{C}$ the set of the robot's collision candidate robots, $\mathscr{D}$ the set of the robot's communication loss candidate robots, d_{com} the robot's maximum communication range and f the kinematic model of the robot.

For performing Step 1 the two last (coupling) constraint equations are left out.

B. Model Predictive Control

Including a model predictive controller results in the following architecture in Fig. 3 for each robot.

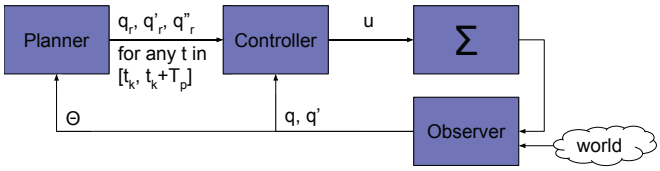


Fig. 3: The planner, knowing the initial and goal configurations of the robot, takes information about obstacles and other robots (denoted by Θ) from the observer and outputs a reference trajectory. The controller takes a state feedback from the observer and the reference trajectory from the planner and generates the system's input u.

The authors of [12] propose a Non-linear Continuous-time Generalized Predictive Control (NCGPC) meant for outdoor mobile robots. Following their approach, we derive a different control law that takes advantage of our receding horizon planner. That new control law is created by replacing the approximation for the reference output by the prediction of the motion planner.

1) Extended Unicycle Model:

To apply the same approach as in [12] we need an extended model for the unicycle mobile robot that integrates the kinematic and dynamic models. Furthermore, we need to be able to write the model in a non-linear control-affine form as shown bellow in eq. 4:

$$\dot{q} = f(q,u) = f_a(q) + \sum_{j=1}^{p} f_{b,j} u_j \tag{4}$$

From [13] we can write such an extended model as in eq. 5:

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{v} \\ \dot{\omega} \end{bmatrix}}_{\dot{q}} = \underbrace{\begin{bmatrix} v\cos\psi \\ v\sin\psi \\ \omega \\ \frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v \\ -\frac{\theta_5}{\theta_2}v\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix}}_{f_a(q)} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix}}_{f_b=[f_{b,1}\ f_{b,2}]} \underbrace{\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}}_{u} \tag{5}$$

where $q \in Q \subset \mathbb{R}^n \,|\, n = 5$ is the state vector and $u \in U \subset \mathbb{R}^p \,|\, p = 2$ the system input. The parameters vector $\theta \in \mathbb{R}^6$ characterizing the dynamics of the robot can be determined either by system identification or by the properties of the unicycle robot such as mass, moment of inertia, impedance of motors. Details on the latter method are provided in [13]. For our particular, simulated case an identification algorithm was used based on the minimization of error in velocities followed by minimization of error in position.

2) Criterion to be minimized:

The objective is to synthesize a control law that minimizes the quadratic error in position and orientation (i.e. pose) over a time-horizon ahead of the current instant t.

Since only error in pose is to be minimized, the system output can be written as follows:

$$z(t) = h(q(t)) = \begin{bmatrix} x & y & \psi \end{bmatrix}^t$$

with $z \in Z \subset \mathbb{R}^m \,|\, m = 3$. And the error as:

$$e(t) = z(t) - z_{\text{ref}}(t)$$

where $z_{\text{ref}}(t)$ is the reference output.

The criterion to be minimized can be written as:

$$J = \sum_{i=1}^{m} \tfrac{1}{2} \int_0^{T_i} (e_i(t+\tau))^2 d\tau$$

where T_i is the prediction horizon for the ith element of $z(t)$ and $e_i(t+\tau)$ the ith element of the prediction error at $t+\tau$ with $0 < \tau \leq T_i$. In this particular case, to find the control law that minimizes J is to find u satisfying the equation:

$$\frac{\partial J}{\partial u} = 0_{p\times 1}$$

For solving the above equation an expression for the prediction error must be defined and the criterion rewritten in a matrix form.

3) Predictive error definition:

In order to obtain an equation for the error $e(t+\tau)$ where the system input u is explicitly present we rewrite $z(t+\tau)$ using Taylor series:

$$z_i(t+\tau) = \sum_{k=0}^{\rho_i} z_i^{(k)}(t)\frac{\tau^k}{k!} + \varepsilon(\tau^{\rho_i})$$

As explained in [12] the vector $\rho = [\rho_1 \ \cdots \ \rho_m]$ is the relative degrees of a non-linear MIMO system (Multiple Input Multiple Output). It is a vector composed by possibly different values of relative degrees ρ_i for each output z_i. ρ_i is the least number of derivatives required to make explicit in the expression of z_i at least one component of the input vector u.

Furthermore, a non-linear control-affine MIMO system (eq. 4) has a relative degree $\rho = [\rho_1 \ \cdots \ \rho_m]$ around q^0 if:

1) $L_{f_{b,j}} L_{f_a}^{(k)} z_i = 0$ for all $1 \leq j \leq p$, for all $k < \rho_i - 1$, for all $1 \leq i \leq m$ and for all q in the neighborhood of q^0
2) the product $D^t D$ is non-singular, D being the decoupling matrix of dimension $m \times p$, given by:

$$D = \begin{bmatrix} L_{f_{b,1}} L_{f_a}^{(\rho_1-1)} z_1 & \cdots & L_{f_{b,p}} L_{f_a}^{(\rho_1-1)} z_1 \\ \vdots & \ddots & \vdots \\ L_{f_{b,1}} L_{f_a}^{(\rho_m-1)} z_m & \cdots & L_{f_{b,p}} L_{f_a}^{(\rho_m-1)} z_m \end{bmatrix} \quad (6)$$

Here we use the standard geometric notation for Lie derivatives summarized bellow by its recursive expression:

$$\begin{cases} L_f^{(0)} z_i &= z_i \\ L_f^{(k)} z_i &= L_f L_f^{(k-1)} z_i = \frac{\partial L_{f_a}^{(k-1)} z_i}{\partial q} f = z_i^{(k)} \end{cases}$$

Using this notation, $L_{f_{b,j}} L_{f_a}^{(k)} z_i$ in condition 1) can be read as the Lie derivative of the kth Lie derivative of z_i with respect to f_a with respect to $f_{b,j}$.

Rewriting the expression for $z_i(t+\tau)$ in a matrix form and excluding the remainder term we obtain the following approximation:

$$z_i(t+\tau) \simeq \underbrace{\begin{bmatrix} 1 & \tau & \cdots & \frac{\tau^{\rho_i}}{\rho_i!} \end{bmatrix}}_{\Lambda_i} \begin{bmatrix} z_i(t) & \dot{z}_i(t) & \cdots & z_i^{(\rho_i)}(t) \end{bmatrix}^t$$

Replacing the first matrix by the more compact notation Λ_i and using Lie derivatives, one can write:

$$z_i(t+\tau) \simeq \Lambda_i L_{z_i} \quad (7)$$

where

$$L_{z_i} = \begin{bmatrix} L_f^{(0)} z_i(t) & L_f^{(1)} z_i(t) & \cdots & L_f^{(\rho_i-1)} z_i(t) & L_f^{(\rho_i)} z_i(t) \end{bmatrix}^t$$

which, assuming condition 1) above, can be simplified to:

$$L_{z_i} = \begin{bmatrix} L_{f_a}^{(0)} z_i(t) \\ \vdots \\ L_{f_a}^{(\rho_i-1)} z_i(t) \\ L_{f_a}^{(\rho_i)} z_i(t) + L_{f_b}(L_{f_a}^{(\rho_i-1)} z_i(t)) u(t) \end{bmatrix}$$

This last form of L_{z_i} makes the system input u explicit in the expression of $z_i(t+\tau)$. Functions f, f_a and f_b come from the model in eq. 4.

As for the second term in the prediction error expression, $z_{\text{i,ref}}(t+\tau)$, it can be kept undetermined until the expression for u, i.e. the control law, is found. This is possible because our planner can give its value for any $\tau \mid 0 < \tau \leq T_i$ as long as $T_i \leq T_p - T_c$. That last condition over T_i is needed because $z_{\text{ref}}(t+T_p-T_c)$ is the further in time the planner can output a valid reference trajectory for any given t.

4) Control law equation:

After some algebraic manipulation we derive the final expression for the control law as shown in eq. 8.

$$\begin{aligned} \frac{\partial J}{\partial u} &= 0_{p\times 1} \\ &\Rightarrow \quad u = -(D^t D)^{-1} D^t (K^{ss})^{-1} (K^s L_y - R^s) \end{aligned} \quad (8)$$

where D is the decoupling matrix (eq. 6), K^{ss} and K^s the gain matrices (eq. 9 to 12), L_z the prediction output matrix (eq. 13) and R^s the future reference output matrix (eq. 14 and 15).

$$K^s = \text{diag}([K_1^s \ \cdots \ K_m^s]) \quad (9)$$

$$K^{ss} = \text{diag}([K_1^{ss} \ \cdots \ K_m^{ss}]) \quad (10)$$

with K_i^s the last line of the matrix K_i and K_i^{ss} the last element of the vector K_i^s. K_i being defined as:

$$K_i = \int_0^{T_i} \Lambda_i^t \Lambda_i d\tau \quad (11)$$

which gives the following expression for each element of K_i:

$$K_{i,(a,b)} = \frac{T_i^{(a+b)+1}}{((a+b)+1)a!b!} \quad (12)$$

with $a,\ b \in [0,\ \rho_i] \subset \mathbb{Z}$ the row and column indexes.

$$L_z = \begin{bmatrix} z_1 & \cdots & L_{f_a}^{(\rho_1)} z_1 & \cdots & z_m & \cdots & L_{f_a}^{(\rho_m)} z_m \end{bmatrix}^t \quad (13)$$

$$R^s = [R_1^s \ \cdots \ R_m^s]^t \quad (14)$$

R_i^s being the last element of the row vector R_i, where

$$R_i = \int_0^{T_i} z_{i,\text{ref}} \Lambda_i d\tau \quad (15)$$

5) Finding u for a unicycle-like robot:

In order to find u, matrices D, K^s, K^{ss}, L_z and R^s must be determined. To do so the vector ρ is needed. A way of finding ρ for unicycle robots is by computing $L_{f_{b,j}} L_{f_a}^{(k)} z_i$ using the model in eq. 5 with k beginning at 0 and incrementing it until the conditions 1) and 2) presented before are satisfied.

Applying that procedure, we find $\rho = [2 \ 2 \ 2]$. Consequently, matrices D, K^{ss}, K^s and L_z can be written as bellow:

$$D = \begin{bmatrix} \frac{\cos\psi}{\theta_1} & 0 \\ \frac{\sin\psi}{\theta_1} & 0 \\ 0 & \frac{1}{\theta_2} \end{bmatrix}, \quad K^{ss} = \text{diag}\left(\begin{bmatrix} \frac{T_1^5}{20} & \frac{T_2^5}{20} & \frac{T_3^5}{20} \end{bmatrix} \right),$$

$$K^s = \mathrm{diag}\left(\left[\left[\begin{array}{ccc} \frac{T_1^3}{6} & \frac{T_1^4}{8} & \frac{T_1^5}{20} \end{array}\right]\cdots\left[\begin{array}{ccc} \frac{T_3^3}{6} & \frac{T_3^4}{8} & \frac{T_3^5}{20} \end{array}\right]\right]\right),$$

$$L_z = \begin{bmatrix} x \\ v\cos\psi \\ \left(\frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v\right)\cos\psi - v\omega\sin\psi \\ y \\ v\sin\psi \\ \left(\frac{\theta_3}{\theta_1}\omega^2 - \frac{\theta_4}{\theta_1}v\right)\sin\psi + v\omega\cos\psi \\ \psi \\ \omega \\ -\frac{\theta_5}{\theta_2}v\omega - \frac{\theta_6}{\theta_2}\omega \end{bmatrix}$$

R^s can be found (numerically or analytically) from the planner's output according to the following expression:

$$R^s = \frac{1}{2}\begin{bmatrix} \int_0^{T_1} x_{\mathrm{ref}}(t+\tau)\tau^2 d\tau \\ \int_0^{T_2} y_{\mathrm{ref}}(t+\tau)\tau^2 d\tau \\ \int_0^{T_3} \psi_{\mathrm{ref}}(t+\tau)\tau^2 d\tau \end{bmatrix} \tag{16}$$

According to [14], the stability of the closed-loop system using a nonlinear generalized predictive control depends only on the relative degree ρ and the control order p. For $\rho = [2\ 2\ 2]$ and $p = 2$ the closed-loop system is stable.

IV. Experimental results

A. Simulation setup

Simulations using the planner and controller presented before were carried out using a physical simulation environment called XDE [15]. Its visual environment can be seen in Fig. 4.

The simulated vehicles have specifications similar to the Adept Lynx AIV produced by Adept Technology. The setup consisted of about 7 obstacles and 3 robots in a 15×15 m area. Obstacles radius were chosen randomly between 0.4 and 1.5 m. The pose of obstacles and robots was generated also at random only excluding cases of conflict, for instance, cases where two robots had overlapping initial poses. Each robot traveled a total of 14 m approximately. The overall aspects of the simulations can be seen in the video supplement.

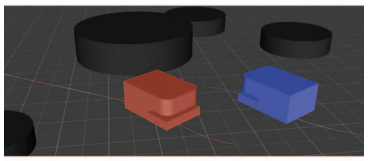


Fig. 4: XDE 3D visual environment during Fig. 2 simulation.

B. Controllers performance

In order to analyse the controller performance, three different control laws were compared: **NCGPC** (Nonlinear Continuous-time Generalized Predictive Control) is the control law presented in [12]; **NCGPC-M** (NCGPC-Modified) is the control law presented in the previous section; **TRVSK** (Tracking Reference Vehicle with Same Kinetics) is presented in [16] and, differently from the other two control laws, it is not predictive.

Fig. 5 shows the result of three identical simulations (same reference trajectory, robot, obstacles) except for the control law adopted to follow the reference trajectory. 6 obstacles were placed in the simulated area as well as 4 waypoints to which the robot passed by before reaching its goal near point $(-4,-5)$. Motion planning main parameters, T_p and T_c, were set to 1.2 s and 0.3 s respectively[2]. The three different paths for each simulation and the reference trajectory are overlapped and their non-coincidence can better be seen in Fig. 5b. Table I shows a comparison of the three control laws based on results of the three simulations in Fig. 5. Additionally, Fig. 6 shows the pose error during the first 20 seconds of the simulations, which is nearly the first half of the robot's path[3].

From Table I and Fig. 6 we can see that NCGPC-M shows the smaller root mean square (RMS) and smaller maximum values for both position and orientation errors. This indicates that the NCGPC-M is the control law that performs the best among the three studied with regard to error minimization.

TABLE I: Comparison of control laws

	TRVSK	NCGPC	NCGPC-M
RMS($\|[x_{\mathrm{err}}\ y_{\mathrm{err}}]\|$) (cm)	6.93	1.17	0.44
max($\|[x_{\mathrm{err}}\ y_{\mathrm{err}}]\|$) (cm)	31.28	4.26	1.92
RMS(ψ_{err}) (deg)	2.78	0.75	0.34
max(ψ_{err}) (deg)	16.29	4.84	1.28

C. Computational cost

Regarding the planner, two different computation routines can be distinguished; the optimization routine computing a parametric optimized solution, and the output evaluation which evaluates that solution at given times.

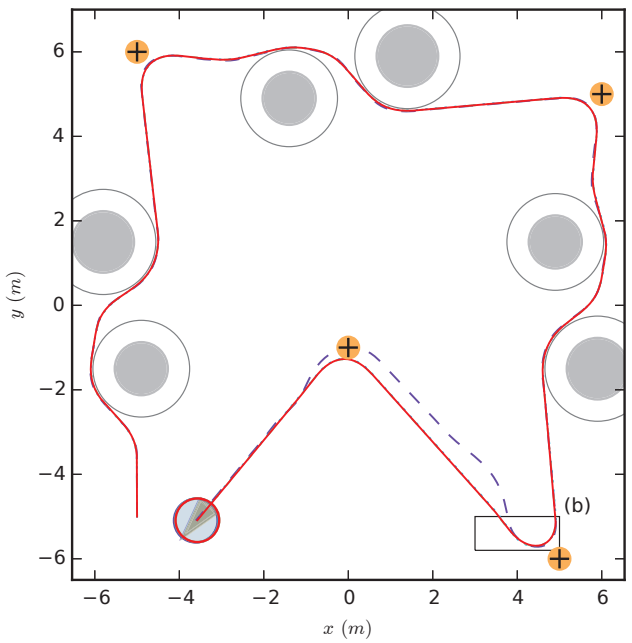
In accordance with the receding horizon principle, the control of the robot is performed based on the output evaluation performed at instants $t \in [\tau_k, \tau_k + T_c]$ which is based on the parametric solution found by the optimization routine at a previous time $t_{\mathrm{pre}} \in [\tau_k - T_c, \tau_k)$. Therefore, the response time of the planner depends only on the output evaluation routine, and not on the optimization routine itself[4].

The output evaluation routine, in turn, depends on how the solution $(\hat{q}(t), \dot{\hat{q}}(t), \ddot{\hat{q}}(t))\ \forall t \in [\tau_k, \tau_k + T_p]$ is parameterized as its only work is to evaluate the solution at a given time. B-splines were used for that purpose. The time for evaluating a b-spline curve at a given moment depends only on three parameters, namely its number of nonzero knot spans, its degree and its dimension. Those parameters do not directly depend on the optimization problem size (number of obstacles and robots). Overall, this suggests that the planner's

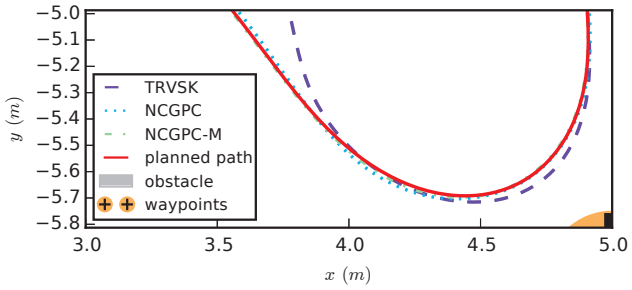
[2]For some discussion about choosing values for T_p and T_c see [10]

[3]in the second half of the path, some high differences between errors make difficult to appreciate the graph

[4]provided enough computational resource for computing in parallel both routines, the output evaluation and optimization for the next time timeslot.



(a)



(b)

Fig. 5: Control laws comparison. (a) General configuration of the simulation. (b) Zoom on the robots' paths stressing the non-coincidence of the planned path and the three executed paths for each control law. Travel time is around 48 s.

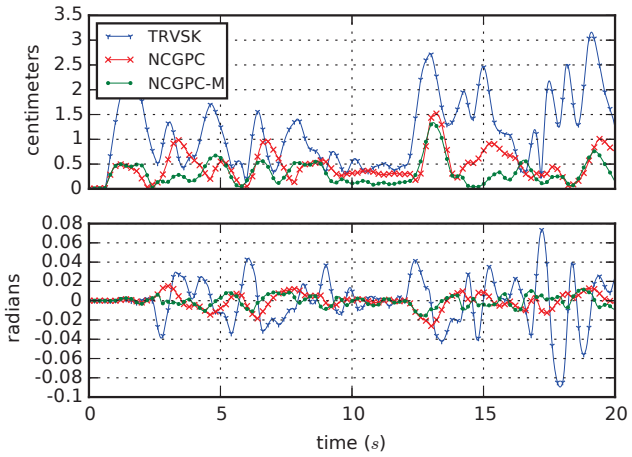


Fig. 6: Errors in position and orientation for the first 20 seconds of the simulation shown in Fig. 5.

response time is independent from the size of the problem which entails a good scalability.

As for the controllers, TRVSK and NCGPC approaches were implemented having constant time complexity while NCGPC-M implementation was $O(n)$ on the number of samples used for integration when approximating matrix R^s (eq. 16). If an analytical solution for R^s would be provided, NCGPC-M could also have constant complexity but it would not necessarily be quicker than the current implementation for relatively small number of samples.

Table II shows the mean and standard deviation of 4778 measurements of elapsed time for each of the four routines: the output evaluation routine in the planner and the three different control routines.

TABLE II: Performance of planning and control implementations on an Intel i7-5600U CPU

	Planner	TRVSK	NCGPC	NCGPC-M
Mean elapsed time (μs)	6.48	1.79	1.89	33.86
Standard deviation (μs)	19.16	1.12	0.59	6.09

All controllers were coded in C++03 STL language and compiled using Visual C++ 10.0 compiler. They were run on an Intel i7-5600U CPU.

The NCGPC-M control which had the best performance for minimizing the position and orientation errors presents the greatest computational cost of the three controllers, which is expected since it computes a numerical integration the others do not.

In Fig. 7a we see a histogram based on the same 4778 calls to the NCGPC-M controller. For 95.44% of the calls the elapsed time was inferior to 35 μs (~29 kHz) and in no case the elapsed time was bigger than 140 μs (~7 kHz).

Likewise, a histogram for the elapsed times of the output evaluation routine in the planner can be seen in Fig. 7b. 96.76% of the calls falls under 11 μs of elapsed time. The second most significant bar in the graph represents the calls where a new thread was spawned responsible for solving the optimization problem.

The total elapsed time for the planner-controller scheme is found by adding both times. Even considering the worst case and adding the longest elapsed times for the NCGPC-M controller and the planner, a response frequency of more than 2 kHz is possible using the processor mentioned before.

Another important time measurement is the one of the optimization routine. Even though it does not impact the response time, it suggests how short the implementation timeslot T_c can be. In our implementation we used the NLopt nonlinear-optimization package [17], more precisely the SLSQP algorithm ([18], [19]) it provides. It is one of the few free algorithms that supports minimization under both equality and inequality constraints.

Fig. 7c shows the distribution of elapsed times for optimization problem solving. In the worst case, optimization took about 0.14 s which represents a lower bound for T_c. In other words, replanning can be done at ~7.14 Hz.

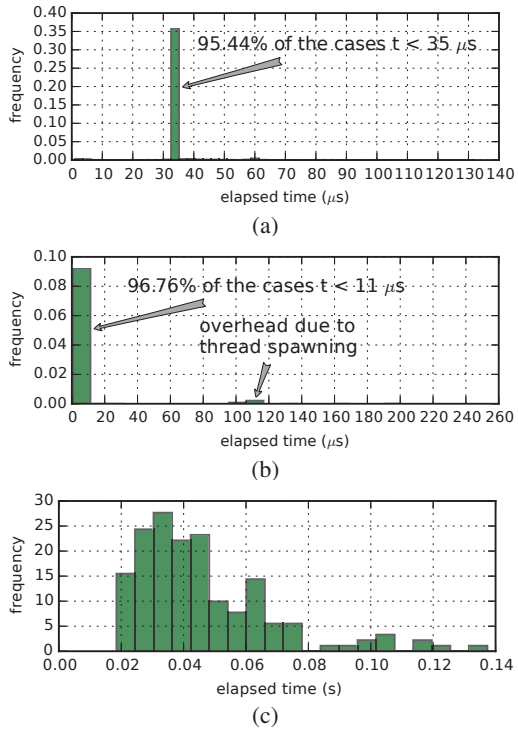


Fig. 7: Histograms of elapsed time of different routines during simulation represented in Fig. 5. (a) Plot based on 4778 calls to the NCGPC-M controller. (b) Plot based on 4778 calls to the output evaluation routine. (c) Plot based on 153 calls to the optimization problem's solver (they took in average 0.046 s)

V. Conclusions

We tackled the problem of taking dynamics into account when planning and executing trajectories for a multi-robot system. Our approach for trying to solve this problem was to improve a DRHMP approach so it could generate high quality plans (or trajectories) and add an MPC that fully exploits the information provided by the planner.

The simulated results suggest that this approach allows a system of multiple unicycle-like robots to navigate collision-free with errors from planned position below 2 cm. Furthermore, the results indicate a response frequency for the planner and controller higher than 2 kHz what would allow for their use in a real-time system.

Small errors and real-time capability in addition to the distributed and near-optimal aspects of this approach motivate further research about this take on the mobile multi-robot navigation problem. Besides, some issues remain to be addressed. In particular, fail-safe measures when no solution trajectory can be found by Step 1 or Step 2 must be implemented. The obvious next step will be to perform experiments with real mobile robots and analyze the robustness of this method with regard to perception, communication and non-synchronization among vehicles. Another interesting subject is the integration of this motion planning and control to task planners. That integration could enable a higher level of autonomy of the system making it possible to operate in a more complex environment.

Acknowledgment

This research and development work was carried out in the scope of the Easily diStributed Personal RapId Transit (ESPRIT) project, funded by the European Union's Horizon 2020 research and innovation program, grant agreement Nº653395.

References

[1] S. Robarts, "Autonomous robots are helping to pack your Amazon orders." [Online]. Available: http://www.gizmag.com/amazon-kiva-fulfillment-system/34999/

[2] "Idea Groupe met en place Scallog pour sa préparation de commandes." [Online]. Available: http://supplychainmagazine.fr/NL/2015/2085/

[3] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," The International Journal of Robotics Research, vol. 20, no. 5, pp. 378–400, 2001.

[4] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," IEEE Robotics & Automation Magazine, vol. 4, no. 1, pp. 23–33, mar 1997. [Online]. Available: http://dx.doi.org/10.1109/100.580977http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=580977

[5] J. Bruce and M. Veloso, "Real-time multi-robot motion planning with safe dynamics," in Multi-Robot Systems. From Swarms to Intelligent Automata Volume III. Springer, 2005, pp. 159–170.

[6] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," Journal of Guidance, Control, and Dynamics, vol. 25, no. 1, pp. 116–129, 2002.

[7] Y. Guo and L. E. Parker, "A distributed and optimal motion planning approach for multiple mobile robots," in Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, vol. 3. IEEE, 2002, pp. 2612–2619.

[8] A. Richards and J. How, "A decentralized algorithm for robust constrained model predictive control," in American Control Conference, 2004. Proceedings of the 2004, vol. 5. IEEE, 2004, pp. 4261–4266.

[9] L. Grüne and K. Worthmann, "A distributed nmpc scheme without stabilizing terminal constraints," in Distributed Decision Making and Control. Springer, 2012, pp. 261–287.

[10] J. M. Mendes Filho and E. Lucet, "Multi-Robot Motion Planning: a Modified Receding Horizon Approach for Reaching Goal States," Acta Polytechnica, vol. 56, no. 1, pp. 10–17, 2016. [Online]. Available: https://ojs.cvut.cz/ojs/index.php/ap/article/view/3440/3334

[11] M. Defoort, A. Kokosy, T. Floquet, W. Perruquetti, and J. Palos, "Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: A distributed receding horizon approach," Robotics and Autonomous Systems, vol. 57, no. 11, pp. 1094–1106, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2009.07.004

[12] M. Krid, F. Benamar, and R. Lenain, "A new explicit dynamic path tracking controller using Generalized Predictive Control," International Journal of Control, Automation and Systems, pp. 1–10, 2016.

[13] C. De La Cruz and R. Carelli, "Dynamic modeling and centralized formation control of mobile robots," IECON 2006-32nd Annual Conference on IEEE Industrial Electronics, pp. 3880–3885, 2006.

[14] W.-H. Chen, D. J. Ballance, and P. J. Gawthrop, "Optimal control of nonlinear systems: a predictive control approach," Automatica, vol. 39, no. 4, pp. 633–641, 2003.

[15] X. Merlhiot, J. L. Garrec, G. Saupin, and C. Andriot, "The XDE Mechanical Kernel: Efficient and Robust Simulation of Multibody Dynamics with Intermittent Nonsmooth Contacts," in Proceedings of the Second Joint International Conference on Multibody System Dynamics - IMSD 2012, 2012.

[16] B. Siciliano and O. Khatib, Springer handbook of robotics. Springer Science & Business Media, 2008.

[17] S. G. Johnson, "The NLopt nonlinear-optimization package." [Online]. Available: http://ab-initio.mit.edu/nlopt

[18] D. Kraft, "A software package for sequential quadratic programming," Technical Report DFVLR-FB 88-28, Institut für Dynamik der Flugsysteme, Oberpfaffenhofen, July 1988.

[19] ——, "Algorithm 733: TOMP–Fortran modules for optimal control calculations," ACM Transactions on Mathematical Software (TOMS), vol. 20, no. 3, pp. 262–281, 1994.