Direct Model Navigation issue shifted in the continuous domain by a predictive control approach for mobile robots

Nicolas Morette, Cyril Novales, Laurence Josserand, Pierre Vieyres

Direct Model Navigation issue shifted in the continuous domain by a predictive control approach for mobile robots

Morette N., Novales C., Josserand L., and Vieyres P.

Abstract— **This paper presents a new navigation method for mobile robots, based on direct kinematics model and predictive control approach. It is adaptable to all types of robots taking into account their specific constraints. The research of the optimal trajectory is shifted in the continuous parameters space, which enables the exploitation of all of the robot's capabilities. The use of a stochastic algorithm enables the determination of obstacle bypass trajectories, and simulation results show that the navigation in cluttered environment is improved.**

I. INTRODUCTION

THIS paper addresses the problem of mobile robots navigation. Indeed, the navigator is key to an autonomous mobile robot since it acts as the link between the robot's motion abilities and its environment. Our contribution is to consider navigation methods based on robot direct kinematics model with a new control approach. These navigation methods are independent of the robot's kinematics constraints; the improvement consists in developing a predictive control to continuously determine an optimal trajectory among a considered class of trajectories.

More generally, research works pertaining to navigation have established three main approaches. A first part of these works does not really make the distinction between the path, the trajectory and the pilot features. This is the case when using neural networks [8][20] or fuzzy logic [7][21], which aim to control the robot's overall behavior in a given environment situation such as the one described in their learning algorithms. These approaches are simple to implement and generate results only in well-defined environments. However, they do not provide long term planning to ensure the convergence toward the final goal, and these methods are subjected to stability issues [13].

The second part of navigation researches treats the issue of following the reference trajectory or the reference path provided by the planning module. Since the majority of robots are non-holonomic, an inverse model does not exist. As a consequence, it is possible that the reference trajectory cannot be systematically followed; some approaches resolve this problem by using transverse functions [10][17] or flat outputs [9]. However, they require the determination of a specific model of the robot, which is non-trivial and non-systematic. Therefore, they do not allow the development of a generic method that works regardless of the considered robot.

The last type of approaches uses the knowledge of the robot direct kinematics, dynamic or geometric model. Among the robot's motion abilities, one can choose the trajectory that will results in the expected behavior [5][12][16]. The advantage of these methods is their adaptability to any kind of robots knowing their direct model; the disadvantage is that they are discrete by nature.

The work, presented here, is directed toward the last type of approach, and concerns the development of a navigator based on the robot's direct kinematics model. Our main contribution is to discard the discrete formalism and to propose a continuous one, which better exploit the kinematic potential of the robot by using a well-tested method of control within regulation processes: the model based control [15][18]. Indeed, the notion of trajectory prediction on a "temporal horizon" in robotics is comparable to the notion of predictive control on a "prediction horizon" when using this control theory [4]. Moreover the research of an optimal control system on this horizon is continuous.

In section II, we define our navigator using direct kinematics model (DKM) to generate the robot's trajectories. In section III, we define the cost function used for the proposed model-based navigator, and discuss the optimization tools used to solve the optimization under constraints problem. In section IV, we show some simulations results of our method applied to different mobile robots.

II. FORMULATION OF THE NAVIGATOR USING THE MODEL BASED CONTROL

There are two types of entries provided to the navigator. The first one is a path computed by a path planning module, in the form of a continuous path or a list of waypoints. In this paper, we have considered a path planning module providing successive waypoints. The second entry is a local map of the environment provided by a real time mapping module. We assume that the robots are programmed to map out their local environment.

The navigator has to determine the admissible trajectories for the robot to move safely in a potentially clustered area. It is therefore a trajectory generating module, not a trajectory following controller.

In this first section, we present the generation of trajectories using the direct kinematics model principle.

Contact: cyril.novales@bourges.univ-orleans.fr (+33 248 23 84 70)
PRISME 63, av. de Lattre de Tassigny F-18020 Bourges cedex, France

A. *Classical DKM Navigation*

1) *Trajectory generation using direct kinematics model*

We consider a prediction horizon T_p, and a control function u defined on this $[t_0, t_0 + T_p]$ horizon. In order to be admissible, this control function has to fulfill the robot actuators constraints. By injecting this control function in the robot DKM, the corresponding evolutions of the robot in the cartesian space $\dot{X} = (x, y, \theta)^T$ are determined.

By integration of these evolutions on T_p, the robot's motions relatively to its initial position $X(t_0) = (x(t_0), y(t_0), \theta(t_0))^T$ are computed. The whole trajectory $\Gamma[t_0, t_0 + T_p]$ corresponding to the control function $u[t_0, t_0 + T_p]$ is generated (Fig. 1).

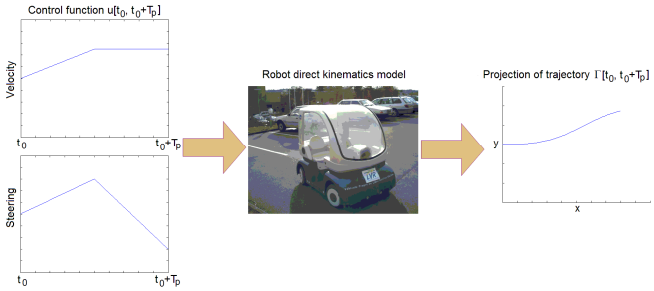

Fig. 1 : the trajectory generation using DKM

2) *Navigation using the trajectory projection principle*

The robot is considered to be equipped with exteroceptive sensors and is able to draw a local obstacle map. By projecting the previous computed trajectory on the local obstacle map, the robot is able to determine if the trajectory is safe or not (taking into account the errors on the obstacle position and the slipping of the robot on $[t_0, t_0 + T_p]$).

These trajectories must follow a desired path generated by a path planner: classically a reference path or a set of waypoints. DKM methods must thus follow 3 steps (Fig. 2):

- to generate a finite number of trajectories,
- to project them in a local map and eliminate the ones colliding with obstacles,
- to select the trajectory the nearest to the desired path.

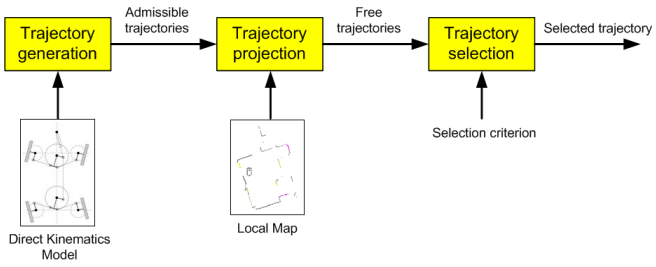

Fig. 2 : the DKM navigation principle

The control function associated to the selected trajectory is sent as an input to the actuators, and applied on a sampling time inferior to the prediction time. The whole process is then repeated periodically with the new updates on the obstacle and robot position, and on the reference path.

All DKM navigation developments are based on this method considering:

- some focus on a specific control function and associated trajectory[6],
- some focus on the selection criteria [16],
- some focus on the projection in a neural network map [5].

Their main advantage is the easy adaptation to any mobile robot whose the DKM is known. However, their application is limited to a finite number of trajectories due to the computation time limits for on-line constraints. As a consequence, the robot motion abilities are limited and so is the accuracy with respect to the reference path. That is the reason why we propose a new control approach associated with it.

B. *The proposed method*

1) *DKM navigation in a continuous control functions space*

The proposed navigation method is based on the same trajectory projection principle, but extends this principle to a continuous control function space. We propose to use parameterized control function, such as linear functions with a slope p_0, or parabolic functions defined by two parameters p_1 and p_2. By making a continuous variation of these parameters, and injecting the corresponding control functions in the robot DKM, it leads to the generation of an infinite number of trajectories. As a consequence, we do not work on a discrete control space and we exploit more thoroughly the robot motion abilities.

Then, we have to determine among this continuous space of trajectories, the one that best fits the navigation problem on $[t_0, t_0 + T_p]$. In order to compare the trajectories, a cost function Z is determined. Finding the set of parameters that minimizes the cost function corresponds to an optimization problem.

2) *Cost function and optimal trajectory*

The navigation problem can be translated into an optimization problem, whose cost function Z is structured with 3 parts (eq. (1)), Fig. 3:

$$Z = Z_{planif} + Z_{robot} + Z_{env} \qquad \textbf{(1)}$$

With:

- Z_{planif} quantifies the gap between the reference trajectory and the evaluated trajectory of the robot,
- Z_{robot} is a penalty considering the constraints inherent to the robot (actuators saturations dynamic constraints...),
- Z_{env} is a penalty considering the constraints inherent to the environment (fixed or mobile obstacles).

A set of parameters p_i corresponds to a control function on $[t_0, t_0 + T_p]$ and corresponds to a trajectory $\Gamma[t_0, t_0 + T_p]$. The set of parameters p_i minimizing Z corresponds to the trajectory which is the nearest to the reference trajectory,

including all the constraints on the robot and on the environment. The navigation problem at t_0 and on the prediction horizon T_p becomes an optimization one under constraints.

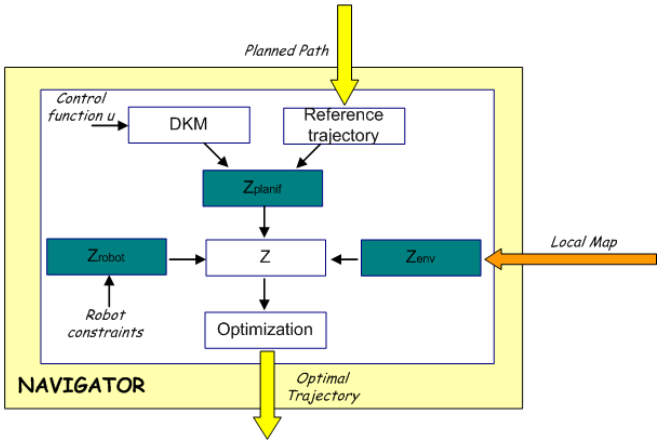

Fig. 3 : the cost function $Z = Z_{planif} + Z_{robot} + Z_{env}$

This problem has to be solved on-line by an optimization algorithm, which can be deterministic (convex problems) or stochastic (non convex problems) as we will see in section III.E.

The trajectory selected by the optimization algorithm is defined on the $[t_0, t_0 + T_p]$ interval. The control function $u[t_0, t_0 + T_p]$ corresponding to the optimal parameters p_i is applied during a sampling time T_e. The problem is updated every T_e, taking into account the changes on the obstacles position, on the planned path and the drift on the robot position. The control loop of this system is inherent to this updating process at the period T_e.

III. THE NAVIGATION PROBLEM

A. *Class of robot considered*

We consider a car-like vehicle whose DKM is classically given by (eq. (2), Fig. 4):

$$\begin{cases} \dot{x} = v.\cos(\theta) \\ \dot{y} = v.\sin(\theta) \\ \dot{\theta} = v.\dfrac{\tan(\xi)}{L} \end{cases} \quad (2)$$

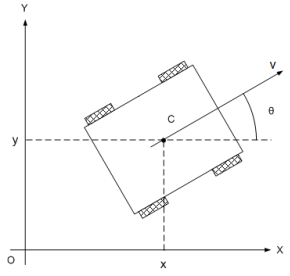

Fig. 4 : a considered class of robot

x and y are the abscissa and ordinate of the robot's control point, θ is its orientation in the basis $(0, \vec{x}, \vec{y})$, v its velocity, ξ its steering angle and L its wheelbase.

We suppose that the robot circulates on a flat surface, so its movements are confined to a plan parallel to the ground.

We define the robot's state by the vector X:

$$X[t_0, t_0 + T_p] = (x\ y\ \theta\ \dot{s})^T_{[t_0, t_0+T_p]} \quad (3)$$

with T_p is the prediction horizon, and $\dot{s}$ its curvilinear velocity along the curve.

Then, we define a control u vector with m lines containing the articular inputs, which are then applied to the robot:

$$u[t_0, t_0 + T_p] = (\dot{q}_1\ \dot{q}_2\ \ldots\ \dot{q}_m)^T_{[t_0, t_0+T_p]} \quad (4)$$

B. Z_{planif} : *trajectory tracking criteria*

1) *The reference curve*

We assume that the path provided to the robot by the path planner is given in the form of a series of successive waypoints including the speeds and orientations at each point. Since the robot drifts from the planned path, a reference curve has to be computed in order to link the current state of the robot X_0 and the desired state X_d corresponding to the next waypoint to be reached. This reference curve has to be defined based on these points' velocities and orientations. Hence, a Bezier curve is used in order to keep the continuity on the robot's positions and velocities, and is defined by four controls parameters, R_1 to R_4. It passes through the points R1(x1, y1) and R4(x4, y4), and the vectors $\overrightarrow{R_1R_2}$ and $\overrightarrow{R_3R_4}$ are tangent to the curve at R_1 and R_4, respectively, where:

$$R1: \begin{cases} x_1 = x_0 \\ y_1 = y_0 \end{cases}, R2: \begin{cases} x_2 = x_1 + \dfrac{1}{3}.\dot{s}_0.\cos(\theta_0) \\ y_2 = y_1 + \dfrac{1}{3}.\dot{s}_0.\sin(\theta_0) \end{cases},$$
$$R3: \begin{cases} x_3 = x_4 - \dfrac{1}{3}.\dot{s}_d.\cos(\theta_d) \\ y_3 = y_4 - \dfrac{1}{3}.\dot{s}_d.\sin(\theta_d) \end{cases}, R4: \begin{cases} x_4 = x_d \\ y_4 = y_d \end{cases} \quad (5)$$

This Bezier curve is geometric and contains no information concerning the vehicle's kinematics. In order to obtain a reference trajectory, we conduct a parameterization depending on the times shown on this curve (Fig. 5) and developed in 2).

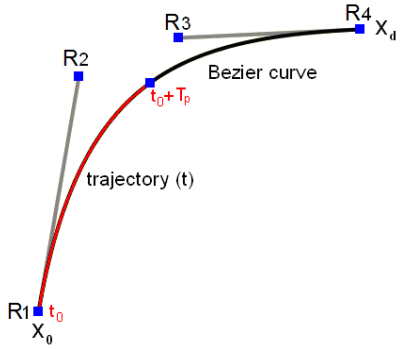

Fig. 5 : the reference curve and reference trajectory

2) Reference trajectory

D is designated as the length of the Bezier curve. When considering the distance $D_{ref} = \dot{s}_d * T_p$ that the robot is likely to travel during the prediction horizon T_p, the equation of the reference trajectory is given by:

$$\begin{cases} x_r(t') = (1-t')^3 x_1 + 3t'(1-t')^2 x_2 + 3t'^2(1-t')x_3 + t'^3 x_4 \\ y_r(t') = (1-t')^3 y_1 + 3t'(1-t')^2 y_2 + 3t'^2(1-t')y_3 + t'^3 y_4 \end{cases} \quad (6)$$

with $t' = \frac{\dot{s}_d}{D}.t$ and $0 < t < T_p$.

The complete reference trajectory equation is:

$$X_{ref}(t) = \begin{pmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \end{pmatrix} = \begin{pmatrix} x_r(\frac{\dot{s}_d.t}{D}) \\ y_r(\frac{\dot{s}_d.t}{D}) \\ \arctan_2(\dot{x}_r, \dot{y}_r) \end{pmatrix} \quad (7)$$

with $t \in [0, T_p], \frac{\dot{s}_d}{D}.t \le 1$

This reference trajectory does not take into account the local environmental constraints and those related to the robot's movements.

3) The robot's admissible trajectories

There is an infinite number of control functions u defined on the prediction horizon $[t_0, t_0+T_p]$ that fulfills the robot's kinematic constraints. Among these functions, we consider a class of control families, which are parameterized by a finite number of parameters n_p. Therefore, each control function will be entirely defined by the set of parameters p_1 to p_{n_p}.

To be applied to a car-type mobile robot, the control functions are then defined by two parameters p_{v1} and p_{v2} for the velocity control and two parameters $p_{\xi 1}$ and $p_{\xi 2}$ for the steering radius (eq. (8)(9), Fig. 6).

$$\begin{cases} v(t) = v(t_0) + p_{v1}.(t-t_0) \\ \xi(t) = \xi(t_0) + p_{\xi 1}.(t-t_0) \end{cases}, t \in [t_0, t_0 + T_p/2] \quad (8)$$

$$\begin{cases} v(t) = v(t_0) + p_{v1}.(T_p/2 - t_0) + p_{v2}.(t - T_p/2 - t_0) \\ \xi(t) = \xi(t_0) + p_{\xi 1}.(T_p/2 - t_0) + p_{\xi 2}.(t - T_p/2 - t_0) \end{cases}, \quad (9)$$

$$t \in [t_0 + T_p/2, t_0 + T_p]$$

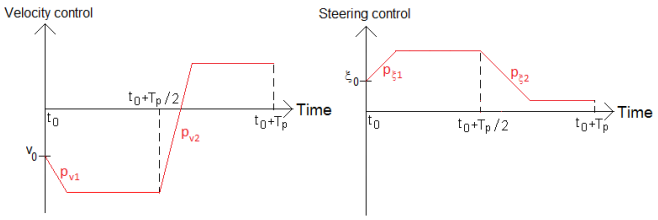

Fig. 6 : The u parameterized control family

The use of this control family into the robot's kinematic model enables the generation of a trajectories family; we then obtain an infinite number of admissible trajectories defined by the set of parameters. A few of these trajectories among this infinite number are represented in Fig. 7. These trajectories show that 2 parameters on the velocity control and 2 parameters on the steering control are sufficient to generate bypass trajectories.

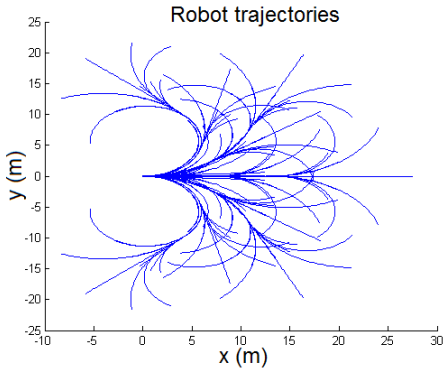

Fig. 7 : wheel prints of a sample of admissible trajectories with (0,0) initial position and a zero initial velocity

Among this infinite number of trajectories, we need to determine the one, which will be the closest to the reference trajectory previously determined with respect to a criterion described in the following section.

4) The cost function

The difference between the planned trajectory of the robot according to the control applied and the references curve is quantified by a cost function, Z_{planif}. It assigns a $Z_{planif}(X_0, t_0, T_p, u)$ value to each control function u over $[t_0, t_0+T_p]$ and is written in the following general form:

$$Z_{planif}(X_0, t_0, T_p, u) = \psi(x(t_0 + T_p)) + \int_{t_0}^{t_0+T_p} l(X(\tau), u(\tau), \tau) d\tau \quad (10)$$

The cost function is composed of a function quantifying the stage cost l and a function quantifying the terminal cost ψ, relying on the quadratic distances between the planned trajectory X(t) of the robot and the reference trajectory $X_{ref}(t)$:

$$\begin{aligned} \psi(x(t_0 + T_p)) &= k_{final} * (X(t_0 + T_p) - X_{ref}(t_0 + T_p))^2 \\ l(X(t), u(t), t) &= k_{stage} * (X(t_0 + t) - X_{ref}(t_0 + t))^2 \end{aligned} \quad (11)$$

with k_{final} and k_{stage} of the decoupled weighting matrix.

C. Z_{robot} : Constraints on the robot

Z_{robot} is defined as the penalty function considering the robot's constraints. These constraints are noted as:

$$C_i(X_0, t_0, t, u) \le 0, \; i \in \{1, \ldots, n_r\}, \quad (12)$$

where n_r is the number of constraints. The constraints are related to kinematic constraints and actuator saturations (the wheels maximum rotation velocity, maximum acceleration and deceleration). The saturations on the set of parameters

are given by:

$$
\begin{aligned}
&C_1, C_2: \quad v_{\min} \le p_{v1}, p_{v2} \le v_{\max} \\
&C_3, C_4: \quad \xi_{\min} \le p_{\xi 1}, p_{\xi 2} \le \xi_{\max} \\
&C_5, C_6: \quad v_0 - decel_{\max} * \frac{T_p}{2} \le p_{v1} \le v_0 + accel_{\max} * \frac{T_p}{2} \\
&C_7, C_8: \quad v(t_0 + \frac{T_p}{2}) - decel_{\max} * \frac{T_p}{2} \le p_{v2} \le v(t_0 + \frac{T_p}{2}) + accel_{\max} * \frac{T_p}{2} \\
&C_9, C_{10}: \quad \xi_0 - \dot{\xi}_{\max} * \frac{T_p}{2} \le p_{v1} \le \xi_0 + \dot{\xi}_{\max} * \frac{T_p}{2} \\
&C_{11}, C_{12}: \quad \xi(t_0 + \frac{T_p}{2}) - \dot{\xi}_{\max} * \frac{T_p}{2} \le p_{v2} \le \xi(t_0 + \frac{T_p}{2}) + \dot{\xi}_{\max} * \frac{T_p}{2}
\end{aligned}
\tag{13}
$$

These constraints are integrated into the cost function using an hyperbolic functions B_i defined by :

$$B_i = -\frac{1}{C_i - C_{i\max}} \tag{14}$$

Thus, the Z_{robot} cost function associated with the robot's constraints is given by:

$$Z_{robot} = \sum^{n_r} B_i(C_i) \tag{15}$$

D. Z_{env} : contraints on the environnent

The second type of constraints is related to the environment. We assume that a mapping of the robot local environment is known, e.g. an on-line refreshed occupancy grid. The idea is to assign a value of 1 or 0 to each square depending on whether or not it is occupied [19]. In order to be able to assimilate the robot's trajectory to a single point trajectory, the obstacles' size is extended to a dimension corresponding to the robot's length.

The family of trajectories generated is projected into this grid. A penalty threshold $Z_{threshold}$ is added in case the robot's trajectories intersect any occupied squares.

$$Z_{env} = \begin{cases} 0 \quad if \quad free \\ Z_{threshold} \quad else \end{cases} \tag{16}$$

E. Optimization

The whole navigation issue is now contained in the criterion Z (1), in the *continuous* parameters space. The optimization algorithm has to find all the parameters p_{v1}, p_{v2}, $p_{\xi 1}$ and $p_{\xi 2}$ minimizing this Z criterion.

$$\underset{/p_i}{Min}\, Z(u(p_i)) = \underset{/p_i}{Min}(Z_{planif} + Z_{robot} + Z_{env}) \tag{17}$$

The optimization algorithms under constraints are classified into two categories: the deterministic and the stochastic algorithms.

The deterministic algorithms generally present the best convergence properties and enable to obtain more precise and quicker solutions; however, they are very sensitive to local minima problems. Moreover, introducing discrete constraints, due to obstacles, has the effect of breaking the cost function convexity and introducing local minima.

Among deterministic algorithms, the widely used Levenberg-Marquardt algorithm was chosen; its convergence step is defined by:

$$\Delta_i = (H_i + \lambda I)^{-1} G_i \tag{18}$$

with H the Hessien and G the gradient of the cost function for the considered parameters, and λ a setting parameter which increases when the cost function diverges.

The second algorithm category is stochastic algorithms, which are based on probabilistic approaches. They are less accurate and are slower to converge; however, they have the advantage of being less sensitive to local minima problems, which enables them to find obstacle bypass trajectories in cluttered surroundings. The stochastic algorithm considered is a simulated annealing [1] whose transition rule probability is defined by:

$$P = 1 - e^{-\frac{\Delta Z}{T_k}} \tag{19}$$

With T_k respecting the following cooling function:

$$T_k = (1 - \frac{k}{k_{\max}}).T_0 + T_{\min} \tag{20}$$

Fig. 8 illustrates a situation for which the robot encounters an U-trap shape obstacle, unplanned for by the path planner (Fig. 8 upper left and right figures). In this case, the local minima of the cost function $Z(p_i)$ are generated in the parameters space representation planner (Fig. 8 lower left and right figures). The deterministic algorithm is trapped in this local minimum in the parameters space (Fig. 8 - top left and bottom left) and cannot find the p_i parameters corresponding to the global minimum of the Z cost function. The stochastic algorithm manages to find the parameters regardless the initial settings (Fig. 8- top right and bottom right); this algorithm's drawback is that there is no convergence clue.

The convergence of the deterministic algorithm to find the global minimum highly depends on the parameters initialization and on local minima. The simulated annealing algorithm enables to escape from local minima without warranty to reach the global minimum within a finite time.

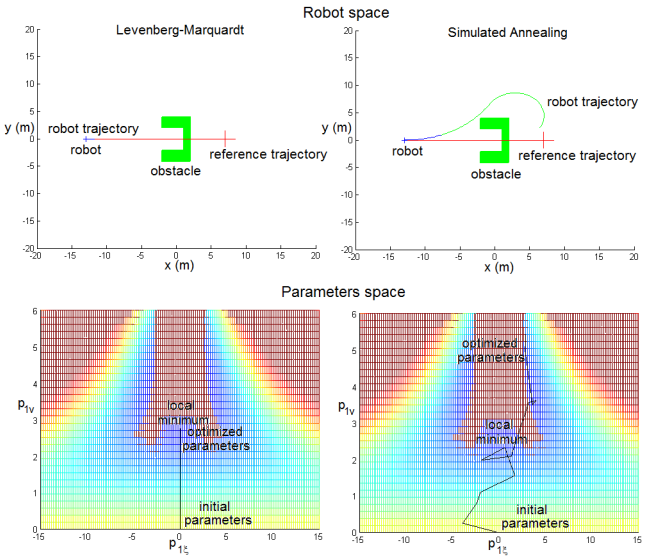

Fig. 8 : comparison between deterministic and stochastic algorithms in the presence of local minima

The complete bypass trajectory on several iterations of the navigator is shown in Fig. 9.

Therefore, choosing either a stochastic or deterministic algorithm depends on the type of terrain covered (open or cluttered), as well as on whether or not the path planner takes into account any potential obstacles. The simulated annealing stochastic algorithm [1] was used to proof the robustness of the navigator when faced with an incomplete planning in cluttered surroundings.

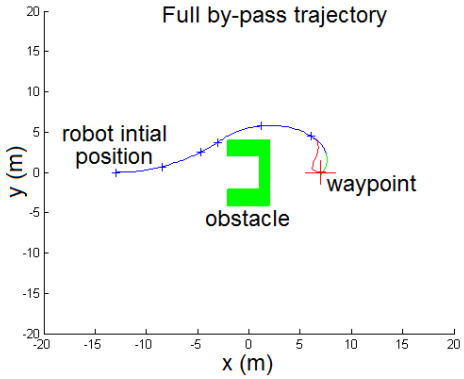

Fig. 9 : bypass trajectory performed through simulated annealing algorithm

IV. RESULTS

A. *Application: the CyCab mobile robot*

The proposed predictive control navigator was applied to a CyCab vehicle (Fig. 10). The particularity of this vehicle is its double steering radius (front and rear axle) which improves its maneuverability in cluttered places, such as urban areas.

Fig. 10 : the CyCab robot

Fig. 11 shows the scheme of the CyCab robot. In [16], we have developed the CyCab kinematic model:

$$\begin{cases} \dot{x}_C = V_C.\cos(\theta) \\ \dot{y}_C = V_C.\sin(\theta) \\ \dot{\theta} = \dfrac{V}{\rho} = V_C \dfrac{\tan(\xi)}{\chi.L} = V_C \dfrac{\tan(\xi)+\tan(k\xi)}{L} \end{cases} \quad (21)$$

where C is the control point of the vehicle, ρ the steering radius and L the wheelbase. When the steering angle on the front axle equals ξ, that of the rear axle equals kξ.

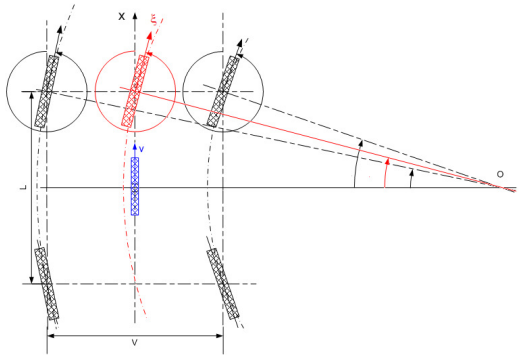

Fig. 11 : the CyCab model

Fig. 12 and Fig. 13 show the navigation simulations for the CyCab robot circulating around obstacles. The path planner indicates four successive waypoints (large crosses); the square zones on an occupancy grid measuring 40x40 m² represent the obstacles' position. The continuous line represents the robot's trajectory; each cross indicates a new iteration of the navigation algorithm navigation (period T_e). At the end of the robot's trajectory, two trajectories are represented: the reference trajectory computed from the current state of the robot and the next waypoint, and the trajectory determined by the optimization algorithm on the next prediction horizon.

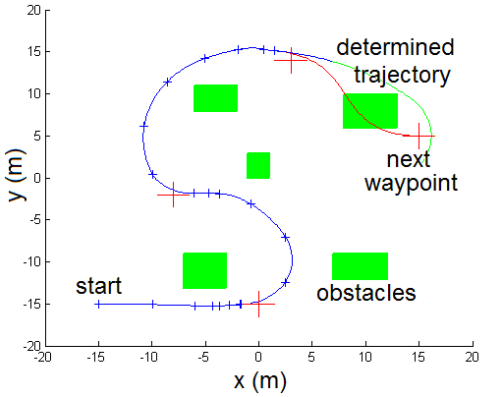

Fig. 12 - scenario 1: navigation of the CyCab robot in a cluttered environment

Fig. 12, shows the case in which the path planner does not take into account any obstacles; this engenders reference trajectories colliding into them. This may occurs for example when dynamic obstacles are present in the environment. Despite this occurrence, the navigator is able to determine the obstacle bypass trajectories, enabling the robot to reach its various waypoints without causing a collision. Indeed, using a simulated annealing algorithm helps us to find solutions avoiding the local minima (induced by the breaking of the convexity of Z due to the presence of obstacles). Besides, using an adapted temporal horizon enables the generation of trajectories, which are long enough to bypass obstacles. We propose the following relationship:

$$\frac{v_{ref} * T_p}{D_o} \approx 3 \qquad (22)$$

where is D_0 the average length of the obstacles' diagonal, chosen in order to determine the appropriate horizon.

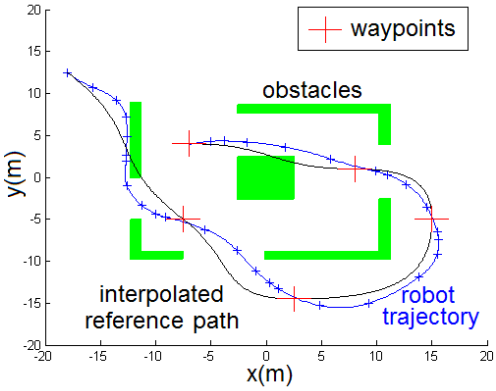

Fig. 13 - scenario 2: navigation of the CyCab robot in a cluttered environment with U traps

Fig. 13 shows an even more constraining scenario (with several traps in U). We note that the robot's complete trajectory avoids obstacles and follows the interpolated reference path, validating the overall navigation method in cluttered surroundings.

Fig. 14 presents results on this thirty-second navigation scenario: the distance measures between the robot's trajectory and the nearest obstacle (red line), and the distance between the robot's trajectory and a reference path computed from the waypoints provided by the path planner (blue line). This comparison shows that the navigator presents enough autonomous properties to diverge from the reference path when needed (i.e. due to obstacles presence).

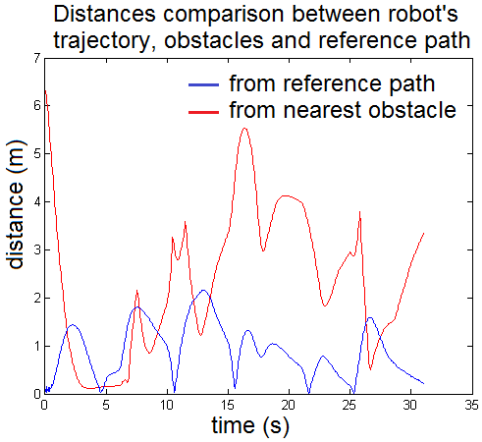

Fig. 14: comparison between the robot's distance from the initial reference path and from the nearest obstacles in scenario 2

B. Application to a differential wheeled robot

The navigation method was also applied to a differential wheeled robot with the same two scenarii (section IV.A) in order to test the adaptability of the method to different types of kinematics. Fig. 15 and Fig. 16 show these simulation results. The robot moves safely among obstacles, using specific trajectories inherent to its movement abilities.

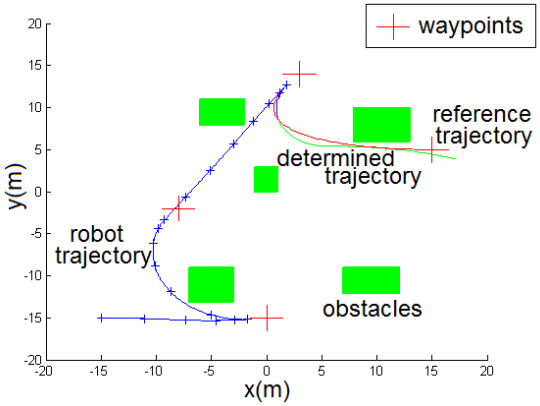

Fig. 15 - scenario 1: navigation of a differential wheeled robot in the same environment

We notice in Fig. 15 and Fig. 16 that the robot makes maneuvers by itself. It is due to the control family used that enables the reverse motions. This behavior can be controlled through the trajectory criteria Z_{planif}.

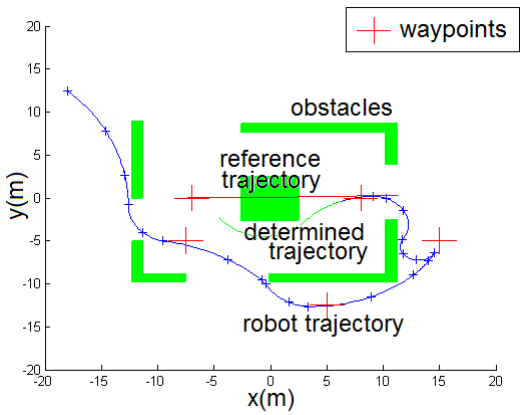

Fig. 16 - scenario 2: navigation of a differential wheeled robot in a cluttered environment with U traps

V. CONCLUSION

In this article, we have developed a new generation of mobile robot navigators thanks to the predictive control approach, which we have adapted to solve navigation issues. This method is based upon a direct kinematic model, which typically can be adapted to all types of robots taking into account their specific constraints. Furthermore, using the predictive control method, which shifts the optimal trajectory research in the continuous parameters space in lieu of the discrete approach, enables the exploitation of all the robot's kinematics capabilities. In addition, by integrating the parameterized functions we are able to master the robot's behavior throughout its motion. By choosing a stochastic algorithm to solve the optimization problem under constraints we are also able to determine any obstacle bypass trajectories as well as to remedy an inadequate path planning.

This navigator is a pivot between the reactive and deliberative parts of the robot control architecture. From a deliberative point of view, the simulations showed that the robot is capable of anticipating obstacles bypass trajectories on its predictive horizon T_p. From a reactive point of view, it takes into account unplanned obstacles during its sampling period near T_e. Therefore, these simulations show that the proposed navigator is capable of circulating correctly in cluttered surroundings by avoiding obstacles not accounted for by the path planner. These capabilities facilitate the link between the deliberative part (path planning) and the reactive part of the control architecture. The next phase of this work involves its application to different robots, which will enable us to test proof the compatibility of this method with respect to computing time as well as the perception and environment's model.

REFERENCES

[1] Aarts, E. H. L., and v. Laarhoven, Statistical cooling: A general approach to combinatorial optimization problems, Philips J. Res., 40(4), 193-226, 1985

[2] Adachi, Y.; Saito, H.; Matsumoto, Y.; Ogasawara, T., *"Memory-based navigation using data sequence of laser range finder"*, Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International symposium on Volume 1, Issue , 16-20 July 2003 Page(s): 479 - 484 vol.1

[3] Agirrebeitia, J., Aviles, R., de Bustos, I.F. and Ajuria, G. , *«A new APF strategy for path planning in environments with obstacles »* ; Mechanism and Machine Theory, Volume 40, Issue 6, June 2005, Pages 645-658

[4] Alamir M.,*"Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes : A Parametrized Approach for Fast Systems*". Lecture Notes in Control and Information Sciences, Springer, London, ISBN 1-84628-470-8. 2006

[5] Belker, T., Schulz, D. « Local Action Planning for Mobile Robot Collision Avoidance», I*ntelligent Robots and System, 2002. IEEE/RSJ International Conference on Volume 1, 30 Sept.-5 Oct. 2002 Page(s):601 - 606 vol.1.*

[6] Bonnafous, D. Lacroix, S. Simeon, T. , « Motion generation for a rover on rough terrains", IROS 2001, Volume: 2, On page(s): 784-789 vol.2, Meeting Date: 10/29/2001 - 11/03/2001 Location: Maui, HI, USA ISBN: 0-7803-6612-3

[7] Chatterjee, R., Matsuno, F., *« Use of single side reflex for autonomous navigation of mobile robots in unknown environments »* ; Robotics and Autonomous Systems, Volume 35, Issue 2, 31 May 2001, Pages 77-96

[8] Er, M.J., Tien Peng Tan, Sin Yee Loh, *« Control of a mobile robot using generalized dynamic fuzzy neural networks »* ; Microprocessors and Microsystems, Volume 28, Issue 9, 2 November 2004, Pages 491-498

[9] Fraisse, P., Gil. A., Zapata, R., *« Stratégie de commande décentralisée d'une flottille de robots mobiles terrestres téléopérés. »*, JNRR 05, Guidel, October 5-7

[10] Fruchard, M., « Méthodologies pour la commande de manipulateurs mobiles non-holonomes », thèse soutenue le 23 septembre 2005 à l'école nationale supérieure des mines de paris - sophia antipolis , préparée à l'INRIA Antipolis

[11] Ibrahim, M.Y., Fernandes, A., « Study on mobile robot navigation techniques », 2004 IEEE International Conference on Industrial Technology , Volume 1, 8-10 Dec. 2004 Page(s):230 - 236 Vol. 1

[12] Klancar, G., Skrjanc, I., "Tracking-error model-based predictive control for mobile robots in real time", Robotics and Autonomous Systems, Volume 55 , Issue 6 (June 2007), Pages 460-469, ISSN:0921-8890, 2007

[13] Koren, Y.; Borenstein, J. **, "**Potential field methods and their inherent limitations for mobile robot navigation"; *Robotics and Automation*, Sacramento, California, April 7-12, 1991, pp. 1398-1404 vol.2

[14] Marquardt , D., "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". *SIAM Journal on Applied Mathematics* **11**: 431–441, 1963

[15] Morette, N, "Contribution à la navigation de robots mobiles : approche par modèle direct et commande prédictive", Phd Thesis, universite d'Orleans, France,, december 18, 2009

[16] Morette N., Novales C., Vieyres P., « Inverse versus Direct Kinematics Model based on Flatness and Escape Lanes to control CyCab Mobile Robot », IEEE-ICRA 2008, Pasadena, USA, May 19-23, 2008

[17] Morin, P. et Samson, C. « Trajectory tracking for non-holonomic vehicles : overview and case study ». Dans Kozlowski, K., éditeur, 4th Inter. Workshop on Robot Motion Control, pages 139–153.

[18] Richalet, J., Rault, A., Testud, J.L., Papon, J., *"Model Predictive Heuristic Control : Applications to Industrial processes"*, Automatica, 14, 413-428 (1978)

[19] Tengda Sun, Jinfeng Wang, "A traffic cellular automata model based on road network grids and its spatial and temporal resolution's influences on simulation Simulation Modelling Practice and Theory", Volume 15, Issue 7, August 2007, Pages 864-878

[20] Wang, X.; Hou, Z.; Zou, A.; Tan, M.; Cheng, L. , "A behavior controller based on spiking neural networks for mobile robots" Neurocomputing Volume 71 , Issue 4-6 (01 2008) Pages 655-666

[21] Xu, W.L., « A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behavior-based mobile robot » ; *Institute of Technology and Engineering, College of Sciences, Massey University, Palmerston North, New Zealand ;* Received 22 June 1998 ; accepted 20 August 1999