Randomized Model Predictive Control for Robot Navigation

Jorge L. Piovesan and Herbert G. Tanner

Abstract— **The paper suggests a new approach to navigation of mobile robots, based on nonlinear model predictive control and using a navigation function as a control Lyapunov function. In this approach, the nonlinear optimal control problem is treated using randomized algorithms. The advantage of the proposed combination of navigation functions for robot motion planning with randomized algorithms within an MPC framework, is that the control design offers stability by design, is platform independent, and allows the designer to trade-off performance for (computation) speed, according to the application requirements.**

I. Introduction

Robot navigation is an instance of a motion planning problem in the presence of constraints. Obtaining provable obstacle avoidance and convergence results in this context is a difficult problem on its own merit, which is possibly why issues of optimality and uncertainty have received less attention. This paper presents an approach that aims at obtaining *local optimal, combined motion planning and control* designs, within a methodology that can potentially handle model and environment uncertainty.

Mobile robot navigation has been approached mainly from two different perspectives: the first is to plan and generate a reference trajectory and then track this trajectory; the second is to use feedback based on the gradient of a navigation function [7]. The former approach consists on a planning phase (which specifies a path for the robot [8]–[10]), and a tracking phase (which drives the robot through the pre-specified path [11]–[14]). The latter approach consists of creating a potential function that captures the topology of the obstacle free space and the robot's destination. Contrary to typical potential field methods, navigation functions can be tuned so that all critical points except for the destination are not attractive (saddles).

Optimal feedback control design for general nonlinear systems is prohibitively complex, both analytically and computationally. One approach that is gaining momentum due to the availability of more powerful computation platforms is that of model predictive, or receding horizon optimization. Model Predictive Control (MPC), uses online, finite-time, iterative optimization to obtain a feedback controller for a given system [1]–[3]. In general, one iteration consists of performing a finite time optimization, with initial condition the current state of the system, and computes the controller that optimizes a cost functional over the finite interval (prediction horizon); then the obtained controller is applied over a much shorter finite horizon (control horizon) before the process is repeated at the new state. Challenges include the solution of the finite time optimization, usually done numerically, and ensuring stability for the overall closed loop system.

The solution to the finite horizon optimization problem has been usually obtained using model based techniques [2]. The challenge there is that the analytical effort required for their application increases with the complexity of the model. On the other hand, randomized algorithms have been used for robust control design [4], [5], with some basic assumptions on the system model. The objective of the randomized methods is to obtain a solution that is very close to the optimal one, with probability *arbitrarily close to one*. This is achieved by *sampling* the set of potential solutions sufficiently many times, and choosing the sample that yields the best performance. Randomized techniques have been previously used to solve some robust control problems that are NP-hard [4], [6]. The advantage of these probabilistic methods is that they simplify the analysis and design tasks at the cost of not being able to guarantee the optimality of the solution.

This paper introduces of a randomized version of MPC applied to a mobile robot navigation problem. The idea is to use a navigation function as a control Lyapunov function within an MPC framework. State constraints are encoded in the navigation function, while input constraints are enforced through appropriate sampling. Basing the receding horizon control on a scheme that uses a control Lyapunov function as a terminal cost estimate provides stability for the closed loop trajectories [1]. Introducing control input randomization allows the application of the method independently of the system's model structure —the latter is accounted for in the control Lyapunov function.

Section II recasts a general nonlinear MPC problem as an instance of a randomized optimization problem. Section III outlines a general solution to the randomized model predictive control problem. It is shown that if the controller obtained from the randomized optimization stabilizes the system for each control horizon, the resulting MPC controller asymptotically stabilizes the closed loop system. In Section V the gradient of the navigation function which is in the role of the control Lyapunov function is parameterized and sampled in a way that guarantees stability and constraint satisfaction within the finite MPC horizon. An algorithm is given in which the gradient directions are stochastically optimized. A numerical example is given in Section VI and the paper concludes with Section VII.

Jorge L. Piovesan is currently with K&A Wireless LLC, 2617 Juan Tabo NE Suite A, Albuquerque NM 87112, jpiovesan@ka-wireless.com. Herbert Tanner is with the Department of Mechanical Engineering at the University of Delaware, Newark DE 19711, btanner@udel.edu

Authors were supported by CTA MAST # W911NF-08-2-0004.

II. MPC Design for Nonlinear Systems

The model predictive control design problem is stated similarly to [1]. Consider the nonlinear dynamical system $\dot{x} = f(x, u)$, where $x \in \mathbb{R}^n$ is the state of the system and $u \in U \subseteq \mathbb{R}^m$ is the control input, with $n, m \in \mathbb{Z}$, and f satisfying the standard local Lipschitz continuity conditions.

Define a cost functional

$$J(t, x, u, T) = \int_t^{t+T} q(x(\tau), u(\tau))d\tau + M(x(t + T)) \ ,$$

where $q(x, u)$ is a positive definite function of x and u, $M(x)$ is a terminal cost, t denotes time, and T is the (fixed) prediction horizon. Consider the *finite horizon optimal control problem* (FHOCP):

$$J^*(t, x, T) = \inf_{u[t,t+T)} J(t, x, u, T) \quad \text{(1a)}$$

$$\text{s.t.} \ \ \dot{x} = f(x, u), \quad x \in X \subset \mathbb{R}^n, \quad u \in U \subseteq \mathbb{R}^m \quad \text{(1b)}$$

where $u[t, t + T)$ denotes the control input on the interval of time between t and $t + T$, and X is the admissible region of the state space. The iterative procedure is as follows:

1) Given the current state of the system $x(t)$, solve the FHOCP (1) obtaining the controller $u^*[t, t + T)$, which optimizes $J(t, x, u, T)$ over the finite interval $[t, t+T)$.
2) Apply the u^* over a finite interval $[t, t + T_c)$ where $T_c \leq T$ is the *control horizon*.
3) Update the current state of the system $x(t)$ with $x(t + T_c)$ and go back to 1).

The objective is to solve (1) for each iteration in finite time, and obtain a controller that asymptotically stabilizes the system in the infinite horizon, and respects the state and input constraints, $x \in X$ and $u \in U$ respectively.

Problem 1 *Given a system dynamics $f(x, u)$, a cost functional $J(t, x, u, T)$, and horizons T and T_c, find a control input $u^*(\tau) \in U$ that solves* (1) *at each iteration $[t, t + T)$, and asymptotically stabilizes $f(x, u^*)$ to the origin.*

III. Randomized MPC for Nonlinear Systems

A. Randomized optimization algorithms

Among the technical challenges in implementing a nonlinear MPC algorithm are computational complexity and the need for special numerical optimization libraries. These problems are circumvented here by relaxing optimality in FHOCP and applying randomized algorithms [4], [5]. In a randomized algorithm, one generates a sufficient number of samples from a set of potential solutions, and selects the one that performs the best. The procedure does not ensure the optimality but it can guarantee, however, that it is close to the optimum with very high probability. The advantage of randomized techniques in this setting is that the analytical complexity of the solution is independent of the system's model: the optimization is performed by *simulating* the dynamics using candidate controllers. In contrast, a model-based approach has to take into account the dynamics in the form of a constraint equation. In addition, in the proposed randomized approach state and input constraints are enforced automatically by a combination of sampling and terminal cost construction.

Let $\sigma \in \Sigma$ be a decision vector and $R : \Sigma \to \mathbb{R}$ a cost functional. One is interested in the decision parameter σ^* that yields the best performance $R^* = \inf_{\sigma \in \Sigma} R(\sigma)$. Take N_s independent and identically distributed random samples σ_i, $i = 1, \ldots, N_s$ from Σ according to a probability distribution $P(\sigma)$. Then $R^0 = \min_i R(\sigma_i)$ can approximate R^* [5]:

Definition 1 (Probable Near Minimum [4]) *Given $R(\sigma)$, $\delta \in (0, 1)$, $\alpha \in (0, 1)$, a number $R^0 \in \mathbb{R}$ is said to be a probable near minimum of $R(\sigma)$ to level α and confidence $1 - \delta$ if there exists a set $\tilde{\Sigma} \subseteq \Sigma$ measuring $\mathbb{P}\{\tilde{\Sigma}\} \leq \alpha$ such that $\mathbb{P}\Big\{ \inf_{\Sigma} R(\sigma) \leq R^0 \leq \inf_{\Sigma \setminus \tilde{\Sigma}} R(\sigma) \Big\} \geq 1 - \delta$*

In the above, $\mathbb{P}$ denotes probability. Parameter α (level) quantifies the set of solutions that may not be represented in the samples. For small α, the probability of finding a solution better than R^0 is small. Confidence $1 - \delta$ expresses the probability that the solution is not worse than the actual minimum in the set that is represented in the samples. The desired level and confidence is achieved by adjusting the number of samples N_s:

Lemma 1 (Number of samples [4], [5]) *The number of samples N_s that guarantees that R^0 is a probable near minimum of $R(\sigma)$ to level α and confidence $1 - \delta$ satisfies $N_s \geq \frac{\ln(1/\delta)}{\ln(1/(1-\alpha))}$.*

B. Randomized MPC problem

Now FHOCP is recast in a randomized framework:

Problem 2 *: Given a system $f(x, u)$, a cost functional $J(t, x, u, T)$, horizons T ad T_c, and level and confidence parameters α and δ, find a control input $u^0(\tau) \in U$ such that $J(t, x, u^0, T)$ is a probable near minimum to level α and confidence $1 - \delta$ of $J^*(t, x, T)$ in* (1) *for each $[t, t + T)$, and $f(x, u^0)$ is asymptotically stable for all $t \in [0, \infty)$.*

Although Problems 1 and 2 have different optimality conditions, both aim at an asymptotically stable closed loop system in the infinite horizon.

IV. Stability Under Randomized Control

A procedure to perform a randomized optimization of a cost functional $R(\sigma)$ is given in Algorithm 1. The algorithm is based on Lemma 1 to guarantee that the obtained solution satisfies the conditions of Definition 1.

To solve Problem 2 using Algorithm 1, one needs to parameterize the control input and sample the parameter space for the given prediction horizon: $U_{[t,t+T)} = \{u[t, t + T) : u(\tau) \in U, \ \forall \tau \in [t, t + T)\}$. Let $u : \Sigma \times \mathbb{R}^n \to U$ be a parameterized control input function, where Σ is the (closed and bounded) parameter space. A sequence of N_s i.i.d. samples $\{\sigma_1, \sigma_2, \ldots, \sigma_{N_s}\}_{[t,t+T)}$ drawn from a p.d.f. $P(\sigma)$ for the time interval $[t, t + T)$, gives rise to N_s

Algorithm 1 Randomized optimization [4], [5]

```
Require: $\alpha$, $\delta$, $R(\sigma)$, $P(\sigma)$, $\Sigma$, $T$, $T_c$.
Ensure: $R^0$ and $\sigma^0$.
1: Compute $N_s$ according to Lemma 1.
2: Generate $N_s$ i.i.d. samples $\sigma_1, \sigma_2, \dots, \sigma_{N_s}$ from $\Sigma$ according to $P(\sigma)$
3: Choose $R^0 = \min_{\sigma_1,\dots,\sigma_{N_s}} R(\sigma_i)$ and $\sigma^0 = \arg\min_{\sigma_1,\dots,\sigma_{N_s}} R(\sigma_i)$
```

i.i.d. input function samples $\{u_1, u_2, \dots, u_{N_s}\}_{[t,t+T)}$, where $u_i[t, t+T) = u(\sigma_i; [t, t+T))$. Then (1) is reformulated as:

$$J^*(t, x, T) = \inf_{\sigma[t,t+T)} J(t, x, \sigma, T) \tag{2a}$$

$$\text{s.t.} \quad \dot{x} = f(\sigma; x), \quad x \in X \subset \mathbb{R}^n, \quad u \in U \subseteq \mathbb{R}^m \tag{2b}$$

where $J(t, x, \sigma, T) = J(t, x, u, T) \circ u(\sigma; x)$, and $f(\sigma; x) = f(x, u) \circ u(\sigma; x, t)$. Proposition 1 that follows provides a mechanism to ensure that the concatenation of input solutions obtained through the randomized algorithm for each interval $[t, t+T)$ asymptotically stabilizes the system $f(x, \sigma)$ on the infinite horizon.

Proposition 1 *If Algorithm 1 is applied to* FHOCP (2) *with samples* $\{\sigma_1, \sigma_2, \dots, \sigma_{N_s}\}_{[t_j,t_j+T)}$, $j \in \mathbb{N}_+$, *drawn such that* $f(x, \sigma_i[t_j, t_j+T))$ *admits a common Lyapunov function* $V_{[t_j,t_j+T)}(x)$ *on each finite interval* $[t_j, t_j + T)$ *then the solution to Problem 2 is given by:*

$$u^0[sT] = u^0[t_1, t_2)|u^0[t_2, t_3)|u^0[t_3, t_4)\dots, \tag{3}$$

where $u^0[t_j, t_{j+1}) = u(x, \sigma^0[t_j, t_{j+1}))$ *and* $\sigma^0[t_j, t_{j+1})$ *is the output of Algorithm 1 applied to* FHOCP (2)*, where* $t_{j+1} = t_j + T_c$.

Proof: The randomized optimization part of this result follows from Lemma 1. The stability part of this proof is based on [15, Theorem 4.1]. Without loss of generality, consider $x = 0$ as the equilibrium point. For all $t \geq 0$, $x(t)$ is absolutely continuous, $V(x)$ is continuous on x, and therefore $V\big(x(t)\big)$ is continuous for all $t \geq 0$.

Function $V\big(x(t)\big)$ is monotonically decreasing since $\dot{V}_{[t_j,t_{j+1})}(x) < 0$ for all $x \neq 0$, and for all $t \in (t_j, t_{j+1})$ and $j \in \mathbb{N}_+$. If one sets $t_j^- \triangleq t_j$ and $t_j^+ \triangleq t_{j+1}$, then one writes $V\big(x(t_{j+1}^-)\big) = V\big(x(t_j^+)\big)$ for all $j \in \mathbb{N}_+$. This implies that the sequence $\{V\big(x(t_k)\big)\}$ satisfies $V\big(x(t_k)\big) > V\big((t_{k+1})\big) > V\big(x(t_{k+2})\big) > \cdots$, where $t_k > t_{k+1} > t_{k+2} > \dots$ is a sequence $t_k \to \infty$ as $k \to \infty$. Function $V\big(x(t)\big)$ is bounded below from zero since it is assumed to be positive definite.

To prove stability consider an $\epsilon > 0$ and choose $r \in (0, \epsilon]$ such that $B_r = \{x \in \mathbb{R}^n | \|x\| \leq r\}$. Define $\alpha \triangleq \min_{\|x\|=r} V(x) > 0$. Take $\beta \in (0, \alpha)$ and let $\Omega_\beta = \{x \in B_r | V(x) \leq \beta\}$. Then Ω_β is in the interior of B_r. The set Ω_β is positively invariant since $V\big(x(t)\big)$ is decreasing implying that $V\big(x(t)\big) \leq V\big(x(0)\big)$ for all $t \geq 0$.

With Ω_β being compact, $\dot{x} = f(x, \sigma[Ts])$ has a unique solution for all $t \geq 0$ whenever $x(0) \in \Omega_\beta$. As $V(x)$ is continuous and $V(0) = 0$, there is a $\hat{\delta} > 0$ such that $\|x\| \leq \hat{\delta} \Rightarrow V(x) < \beta$. Then $B_{\hat{\delta}} \subset \Omega_\beta \subset B_r$, and $x(0) \in B_{\hat{\delta}} \Rightarrow x(0) \in \Omega_\beta \Rightarrow x(t) \in \Omega_\beta \Rightarrow x(t) \in B_r$.

Similarly one shows that for every $a > 0$ there is a $b > 0$ such that $\Omega_b \subset B_a$. Therefore it suffices to show that $V\big(x(t)\big) \to 0$ as $t \to \infty$. Since $V\big(x(t)\big)$ is monotonically decreasing and bounded below from zero, $V\big(x(t)\big) \to c \geq 0$ as $t \to \infty$. Suppose $c > 0$; then by continuity of $V\big(x(t)\big)$ there exists a $d > 0$ such that $B_d \subset \Omega_c$. Then the limit $V\big(x(t)\big) \to c \geq 0$ as $t \to \infty$ implies that $x(t)$ lies outside B_d for all $t \geq 0$.

In the set $\overline{\Omega}_c \backslash \text{int}B_d$ (which denotes the closure of Ω_c less the interior of B_d), there will be a maximum for $\dot{V}$ for each $\sigma \in \Sigma$, which always exists because $\overline{\Omega}_c \setminus \text{int}B_d$ is compact. Let this minimum be $\gamma < 0$. Then $V(x(t)) <= V(x(0)) - \gamma t$, and it follows that there will be a finite time for which V will become negative if $x(t)$ remains outside the set B_d, causing a contradiction. Thus $c = 0$, and the proof is completed. ■

V. Robot Navigation with RMPC

Consider the problem of taking a mobile robot from its initial position to a goal position, avoiding obstacles located in its motion environment, which we assume is perfectly known. In the context of Problem 2, stability is understood as uniform convergence to the goal configuration along collision free trajectories. The robot's workspace and the associated motion planning constraints, are encoded in a navigation function [7]. The implementation details of the navigation function construction are not important for this discussion; any navigation function can be considered. Assume therefore that such a navigation function, denoted $\varphi(x)$, is constructed. Function $\varphi(x)$ will serve as a control Lyapunov function, and the cost functional is constructed as

$$J(t, x, u, T) = \int_t^{t+T} q(x(\tau), u(\tau))d\tau + \varphi(x(t+T)) \ , \tag{4}$$

and control input samples are drawn from the set U, which now becomes state dependent and is defined as $U(x) = \{u \in \mathbb{R}^m \mid -\nabla\varphi(x)^T u > 0\}$. The parameterized family of inputs $u \in U(x)$ at state x is constructed, and expressed in polar coordinates as:

$$u(\sigma; x) = \big(-\|\nabla_x \varphi(x)\|, \arg(\nabla_x \varphi(x)) + \sigma\big) \ , \tag{5}$$

where $\arg(\cdot)$ denotes the angle of its argument. By setting $\sigma = (-\frac{\pi}{2}, \frac{\pi}{2})$ and selecting $u(\sigma; x, \tau)$ from $U(x)$, $\varphi(x, u)$ becomes a common control Lyapunov function, and Proposition 1 applies, at the expense of optimality in minimizing functional J. Then Algorithm 1 returns the solution $\sigma^0[t_j, t_{j+1})$, which corresponds to the input $u^0(\sigma^0, [t_j, t_{j+1})$ that approximately minimizes functional (4).

Relaxing the requirement for optimality in this setting does not affect the stability properties of the closed loop system [16], as long as one can guarantee the existence of an *improving* control law $u(\sigma; x)$; in this case, the existence of a navigation function implies the existence of this improving control law. Relaxing optimality by forcing negative definiteness in $\dot{\varphi}$ is also dictated by the fact that $\varphi(x)$ encodes applicable state constraints as configuration space obstacles.

The evaluation of the control law is summarized in Algorithm 2. In step 3 of Algorithm 2, the control input is used for a shorter, constant time horizon T_c, which is typical in MPC applications.

Algorithm 2 Randomized MPC for Robot Navigation

```
Require: $f(\sigma; x)$, $T$, $T_c$, $J(t, x, \sigma, T)$, $\alpha$, $\delta$, $P(\sigma)$, $\Sigma$.
1: Define $\mathcal{CP}$ as the FHOCP (2).
2: Obtain $\sigma^0[t, t+T)$ using Algorithm 1 to solve $\mathcal{CP}$
3: Apply $u^0[t, t+T_c) = u(\sigma^0[t, t+T); x)$, where $u(x, \sigma)$ is given by (5).
4: $t \Leftarrow (t + T_c)$, $x(t) \Leftarrow x(t + T_c)$.
5: Go back to 1.
```

VI. Numerical Example

A. Simulation Set-up

The system on which the proposed methodology is tested is a single-legged robot, which navigates in a two-dimensional workspace between an initial and a final positions. For this example we use the planar and continuous averaged model of this system's kinematics in [17]. The robot's movement is composed of periodic locomotion cycles illustrated in Fig. 1. A locomotion cycle is composed by a stance phase (between touching ground and lifting), and a flight phase (between lifting and touching ground).

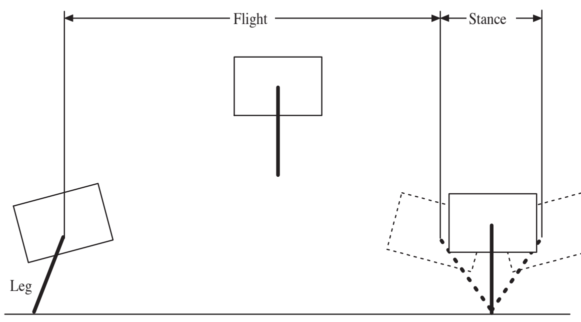

Fig. 1. One dimensional locomotion cycle for an ARL-Monopod II. One locomotion cycle is composed by a flight phase and a stance phase [17].

In [17], the (single-dimensional) average speed of the ARL-Monopod II between flight and stance phases is analytically computed. Here, this average kinematic model is extended in two dimensions for a planar environment as follows:

$$\dot{x} = -\frac{2r_0}{T_s}\sin\left[\theta_x \sin((1-\rho)\pi)\right] \tag{6a}$$

$$\dot{y} = -\frac{2r_0}{T_s}\sin\left[\theta_y \sin((1-\rho)\pi)\right] \tag{6b}$$

where $(x, y) \triangleq \mathbf{x}$ is the position of the robot in the plane, and θ_x, and θ_y are the amplitudes of the hip oscillations along the x and y directions respectively. The latter are considered to be the control inputs. The remaining parameters are constants : r_0 is the length of the robot's leg, T_s is the time that the stance phase last during a locomotion cycle of the robot, T_{step} is the total time of the locomotion cycle composed of a stance phase and a flight phase, and $\rho = T_s/T_{step}$. The parameters r_0, T_s, and ρ are the same on both dimensions for simplicity. The parameter values used in our numerical example are the same as those in [17].

The robot moves in a environment that includes a rectangular obstacle as shown in Fig 3. The circular workspace is centered at $(x_w, y_w) = (-3, 3)$ and has a radius of $r_w = 3$. The robot's initial position is $(x_r, y_r) = (-3, 7)$, while the destination point is at $(x_g, y_g) = (-4, 3)$. The obstacle is centered at $(x_o, y_o) = (-2, 5)$, and its dimensions are $l_o = 2$ and $w_o = 1$. Let $s_o \triangleq \sqrt{\left(\frac{x-x_o}{l_o}\right)^6 + \left(\frac{y-y_o}{w_o}\right)^6}$, $s_l \triangleq \sqrt{(x-x_w)^2 + (y-y_w)^2}$, $h(z) \triangleq \exp\left(\frac{\lambda}{z^2}\right)$ if $z \geq 0$ and $h(z) \triangleq 0$ if $z < 0$, and define the potential function

$$\varphi(\mathbf{x}) = \tanh\left(\frac{\varphi_g}{1 - \tanh(\varphi_o + \varphi_l)}\right) \tag{7}$$

where

$$\varphi_g = \frac{(x-x_g)^2 + (y-y_g)^2}{20}, \quad \varphi_o = \frac{\mu\, h(2\gamma - s_o)}{h(2\gamma - s_o) + h(s_o)},$$

$$\varphi_l = \frac{\mu\, 2h(s_l - r_w - 2\gamma)}{h(s_l - r_w - 2\gamma) + h(s_l)},$$

and λ, γ, μ are constant tuning parameters that shape the graph of the potential function (chosen here $\lambda = \gamma = 1$ and $\mu = 10$ for this example). A control horizon of $T_c = 0.25$ seconds, and a prediction horizon of $T = 1$ second is selected. The sample space is defined as $\Sigma = [-0.9\pi/2, 0.9\pi/2]$. The minimum number of samples N_s for the randomized optimization follows from Lemma 1.

The cost functional for each prediction horizon is selected as

$$J(t, \mathbf{x}, u, T) = \int_t^{t+T} \varphi(\mathbf{x}) d\tau + \varphi(x(T)) \ , \tag{8}$$

in which the incremental cost penalizes time spent at high potential locations (close to obstacles), but not input magnitude —the latter is adjusted here automatically during sampling. The simulation starts setting the initial condition of the robot $(-3, 7)$ as the initial condition for the prediction phase.

Prediction phase: Compute the negated gradient of $\varphi((x))$, and then generate N_s random deviations from the nominal negated gradient direction. For each sample σ, derive a planar control direction $(\dot{x}_d, \dot{y}_d) \triangleq \big(-\|\nabla\varphi(\mathbf{x})\|, \arg(\nabla\varphi(\mathbf{x})) + \sigma\big)$. This direction is normalized and then used to generate the system's input $u(\sigma; \mathbf{x})$ according to [17, Eqn 1],

$$\theta_x = -\frac{\arcsin(T_s \dot{x}_d / 2r_0)}{\sin((1-\rho)\pi)}, \quad \theta_y = -\frac{\arcsin(T_s \dot{y}_d / 2r_0)}{\sin((1-\rho)\pi)}.$$

These control signals are then applied to (6) for the prediction interval T, and he system's performance as quantified by (8) for every control sample is evaluated. The sample that performs best is selected as the input for the control phase.

Control phase: The control phase starts at the same initial condition as the prediction phase. During this phase the selected controller is applied to the system only during T_c seconds, recording the performance of the system according to (8). After this phase the system ends at a final position (x_f, y_f) which is set as the new initial condition of the next prediction phase. The performance of the control phase is stored and the algorithm returns to the prediction phase with the new initial condition (x_f, y_f).

Final steps: The simulation ends by adding up the performance of all control phases during the simulation period (this is done iteratively). This final value for the performance cost is used to quantify the performance of the randomized MPC algorithm in the next section.

B. Simulation results

The performance of the proposed randomized MPC algorithm is evaluated by testing different values for the level and confidence parameters α and δ. For each (α, δ) combination we take ten simulations runs. With this data the average and the standard deviation of the cost for each (α, δ) combination is computed. This statistics capture the relationship between the computational complexity of the algorithm (number of tested samples) and the performance of the obtained solution.

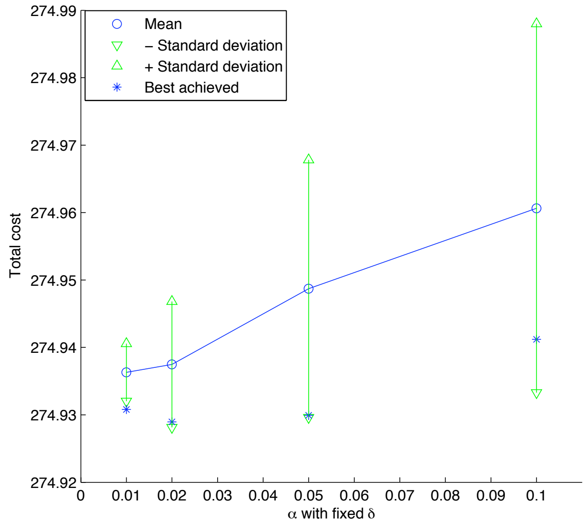

Fig. 2. Overall performance of randomized MPC for different values of α with $\delta = 0.05$. Note that as α decreases, the controller performs better on average and becomes more consistent. However, it is possible to obtain very good solutions even with large values of α.

For the first set of simulations a fixed value for the confidence parameter $\delta = 0.05$ is used. Four different values for the level parameter α: 0.1, 0.05, 0.02, and 0.01 are considered. The corresponding number of samples are 29, 59, 149, and 299. The average and standard deviation of the cost for each value of α is depicted in Fig. 2. Note that as α decreases (number of samples increases), the cost average and its standard deviation also decrease, making the result better on average. Fig. 2 suggests that results may become more consistent as α decreases, but the current (arbitrarily) chosen number of simulations per (α, δ) pair is too small for a statistically confident conclusion. It is quite possible, however, to achieve an acceptable solution even with a very small number of samples. There is an obvious trade-off between accuracy and complexity, which the designer can exploit based on available hardware and application requirements. Similar results were obtained for a fixed α and different values of δ.

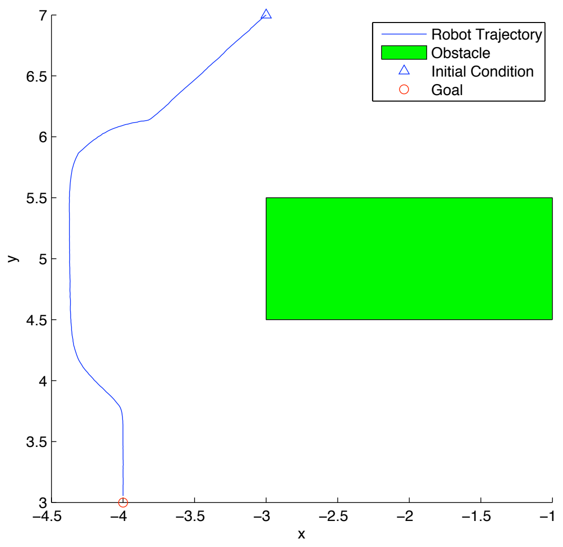

Fig. 3. Robot's trajectory during a simulation test. The robot starts at an initial condition $(-3, 7)$ avoids the rectangular obstacle centered at $(-2, 5)$ with dimensions 2×1, and reaches the goal at $(-4, 3)$.

The robot's behavior during an algorithm test is depicted in Figures 3, 4, and 5. All figures show the robot navigating in the environment, reaching its goal, while avoiding the obstacle. Figure 3 illustrates this behavior in a two dimensional illustration. Figure 4 shows the robot's trajectory on a plane in reference to the level sets of the navigation function, plotted in the third dimension, and Fig. 5 shows a detail of the same trajectory on an inverted section of Fig. 4.

Figure 6 shows the control inputs θ_x^0 and θ_y^0 obtained after calculating the gradient of the navigation function and including the deviation σ^0.

VII. Conclusion

Mobile robot navigation problems can be treated within a model predictive control framework, using navigation functions and combined with randomized optimization algorithms. Such an approach is almost model-independent, to the degree that a control Lyapunov function is available for the system at hand. In addition, it offers the control designer the ability to trade performance for computational speed, without reliance on specialized nonlinear optimization algorithms. The proposed framework combines randomized algorithms with CLF/navigation function-based receding horizon optimization to yield closed loop systems with provable

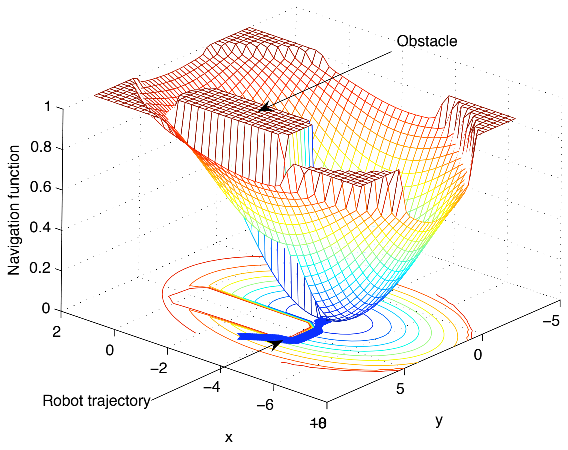

Fig. 4. Robot's trajectory during a simulation test shown with the navigation function. The robot navigates through the environment avoiding the obstacle and reaching goal at the global minimum of the navigation function.

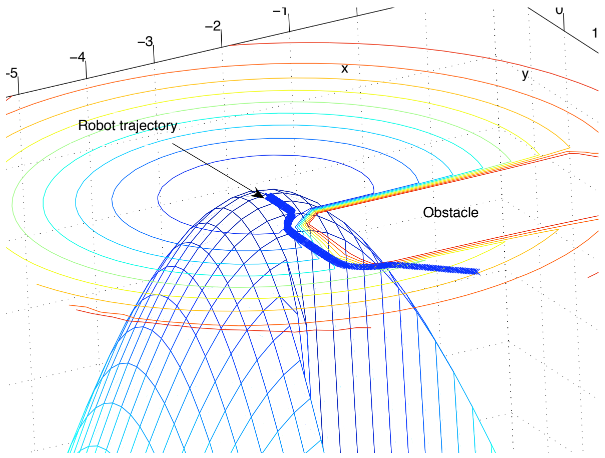

Fig. 5. Robot's trajectory during a simulation test shown with the navigation function. Zoomed, inverted section of Figure 4.

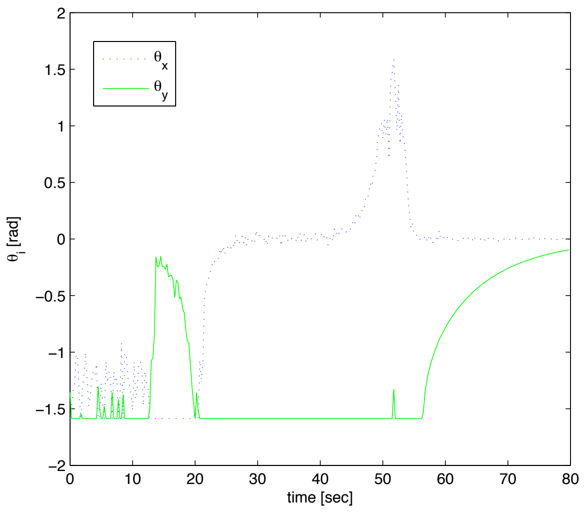

Fig. 6. Optimal control signals θ_x^0 and θ_y^0 obtained from the randomized MPC for a single simulation.

collision avoidance and convergence properties. Numerical results depict the mechanism with which performance is increased at the expense of computational load, although further numerical testing is needed to statistically confirm this hypothesis. Ongoing work focuses on incorporating model and environmental uncertainty.

References

[1] A. Jadbabaie, J. Yu, and J. Hauser, "Stabilizing receding horizon control of nonlinear systems: A control Lyapunov function approach," in *Proceedings of the American Control Conference*, San Diego, CA, USA, 1999, pp. 1535–1539.

[2] J. Primbs, V. Nevistić, and J. Doyle, "Nonlinear optimal control: A control Lyapunov function and receding horizon perspective," *Asian Journal of Control*, vol. 1, pp. 14–24, 1999.

[3] G. De Nicolao, L. Magni, and R. Scattolini, "Stabilizing receding-horizon control of nonlinear time-varying systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 7, pp. 1030–1036, 1998.

[4] M. Vidyasagar, "Randomized algorithms for robust controller synthesis using statistical learning theory," *Automatica, Elsevier Science Ltd.*, vol. 37, pp. 1515–1528, 2001.

[5] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*, ser. Communications and Control Engineering. London, UK: Springer, 2003.

[6] V. Koltchinskii, C. Abdallah, M. Ariola, P. Dorato, and D. Panchenko, "Improved sample complexity estimates for statistical learning control of uncertain systems," *IEEE Transactions on Automatic Control*, vol. 45, no. 12, pp. 2383–2388, December 2000.

[7] E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, October 1992.

[8] J.-C. Latombe, *Robot Motion Planning*. Boston, MA, USA: Kluwer Academic Publishers, 1991.

[9] J. Canny, *The Complexity of robot motion planning*. Cambridge, MA, USA: MIT Press, 1988.

[10] R. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, pp. 700–716, 1993.

[11] G. Oriolo, A. D. Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 835–851, Nov. 2002.

[12] D. Wang and G. Xu, "Full-state tracking and internal dynamics of nonholonomic wheeled mobile robots," *IEEE/ASME Transactions on Mechatronics*, vol. 8, no. 2, pp. 203–214, June 2003.

[13] R. Fierro and F. Lewis, "Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics," *Journal of Robotic Systems, by John Wyley & Sons, Inc.*, vol. 14, no. 3, pp. 149–163, 1997.

[14] J. Yang and J. Kim, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 578–587, June 1999.

[15] H. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[16] A. Jadbabaie, "Receding horizon control of nonlinear systems: A control lyapunov function approach," Ph.D. dissertation, California Institute of Technnology, Pasadena, California, October 2000.

[17] M. Ahmadi and M. Buehler, "Controlled passive dynamic running experiments with the ARL-Monopod II," *IEEE Trans. on Robotics*, vol. 22, no. 5, pp. 974–986, Oct. 2006.