Receding Horizon Control for Convergent Navigation of a Differential Drive Mobile Robot

Marija Seder, *Member, IEEE,*
Mato Baotić, *Member, IEEE,*
and Ivan Petrović, *Member, IEEE*

***Abstract*—A receding horizon control (RHC) algorithm for convergent navigation of differential drive mobile robots is proposed. Its objective function utilizes a local-minima-free navigation function to measure the cost-to-goal over the robot trajectory. The navigation function is derived from the path-search algorithm over a discretized 2D search space. The proposed RHC navigation algorithm includes a systematic procedure for generation of feasible control sequences. The optimal value of the objective function is employed as a Lyapunov function to prove a finite-time convergence of the discrete-time nonlinear closed-loop system to the goal state. The developed RHC navigation algorithm inherits fast re-planning capability from the D* search algorithm, which is experimentally verified in changing indoor environments. The performance of the developed RHC navigation algorithm is compared to the state-of-the-art sample-based motion planning algorithm based on lattice graphs which is combined with a trajectory tracking controller. The RHC navigation algorithm produces faster motion to the goal with significantly lower computational costs and it does not need any controller tuning to cope with diverse obstacle configurations.**

***Index Terms*—Motion planning, Receding horizon control, Lyapunov function, Path planning, Graph searching, Obstacle avoidance.**

I. Introduction

Convergent navigation of a mobile robot refers to a control algorithm that moves the robot from an initial state to a given goal state in an environment, while avoiding obstacles, and guaranteeing finite-time convergence of the closed-loop system to the goal state. The computation time is the main concern in many real-world applications where a robot needs to avoid slow-moving obstacles, such as people or other mobile robots. Some examples are delivery tasks in offices, hospitals, supermarkets, shop floors, warehouses, etc. In general, to find the optimal control sequence one has to optimize over the whole feasible state-input space, which is defined by the geometry of the environment and by the robot dynamics and state-input constraints. In practice, applying the computed optimal control sequence in an open-loop manner would be unwise (or even infeasible), since such control system could not cope with disturbances (e.g., sudden changes in the environment) or discrepancies between the behavior of the model and the real system. One solution that introduces feedback in the control system and allows for quick reaction to the new conditions is to recompute the optimal control input at each time step over a shifted horizon, i.e., to use the receding horizon control.

The receding horizon control (RHC) (also known as the model predictive control (MPC)) is an on-line optimization algorithm that predicts system outputs based on its current states and mathematical model, finds the best control sequence by optimization, and applies the first control input from the optimal control sequence to the system [1]. In this paper we are interested in application of the RHC principle to navigate a differential drive mobile robot, which is the most commonly used type of nonholonomic mobile robots [2]. There are a number of stabilizing RHC methods for navigation of nonholonomic mobile robots reported in the literature. From Brockett's theorem it is well known that a nonholonomic system can not be stabilized with a continuous control law [3, 4]. Several methods design piecewise-smooth control laws with polar representation of the state space variables [5–8], while more recent methods use canonical chained form of the system to design robust time-varying control laws [9, 10]. Besides the stability issue, the receding horizon controllers are computationally expensive and thus not always applicable for real-time use. Computation times can be lowered while maintaining the closed-loop stability by shortening the time horizon, by using suboptimal controllers [7] or by reducing the frequency of the control law computation [11].

All previously mentioned control methods assume that the robot state space is convex and collision free. However, in real environments the state constraint set (the configuration space) is generally non-convex which makes it unsuitable for the standard, linear MPC formulation [12]. This problem can be resolved by splitting the configuration space into convex parts and optimizing them separately part by part [6, 13]. Another approach is to compute a navigation function, which is a special case of the classical potential field function without local minima [14, 15], and employ it as a control Lyapunov function. In [16] the RHC is combined with the navigation function proposed in [15] to guarantee asymptotic stability of a holonomic mobile robot. Discrete navigation functions[1] are computationally less demanding, since they are calculated as the cost of the path to the goal on a discrete set of grid points (occupancy grid map [17]) by a graph search algorithm [18]. In [19] a convergent dynamic window approach (CDW) with the RHC and an interpolated continuous version of the discrete navigation function as a control Lyapunov function is used to establish asymptotic stability of a holonomic mobile robot. Inspired by the CDW of [19], in [20] we obtained fast and convergent navigation for holonomic anyshape mobile robots in dynamic environments by employing the RHC and an interpolated continuous navigation function created over the pose configuration space (x, y, θ). Unfortunately, the above mentioned approaches cannot be applied directly to nonholonomic robots since the gradient of navigation function is not guaranteed to be non-zero, except for trivial cases. For nonholonomic robots one usually utilizes switching controls or combines several different control laws [13, 21, 22].

A complementary problem to the convergent navigation is the trajectory tracking or path-following, which focuses on stabilizing a robot to a predefined path independent of time, while convergent navigation cares only about reaching the goal. High quality paths can be obtained by rapidly-exploring

M. Seder, M. Baotić and I. Petrović are with the Department of Control and Computer Engineering, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia e-mail: {marija.seder, mato.baotic, ivan.petrovic}@fer.hr.

[1]In the following, by the term *discrete* navigation function we consider function whose domain is a discrete set, while *continuous* navigation function will refer to the function that is defined over a continuous domain.

random tree algorithms such as RRT* [23], kinodynamic RRT* [24], or state lattice graphs built of motion primitives [25, 26]. Since all these methods search in a high-dimensional state space, they are time consuming when used at high resolutions, while at lower resolutions they may miss paths in very narrow spaces. Moreover, some approaches require additional, computationally intensive, path smoothing to have a smooth change of accelerations along the path [27]. Since stabilization is in a local vicinity of the computed trajectory, motion in changing environments can be inefficient. Some approaches start with a simple trajectory, which is adapted in the presence of detected obstacles [28]. However, it may be difficult to maintain collision avoidance guarantees in confined and changing environments.

The stabilizing RHC algorithm that we propose in this paper is based on a fast and efficient method for constructing the navigation function (for the differential drive mobile robot) with the global minimum at the goal state and no local minima. We extend the idea of our previous approach for holonomic mobile robots [20] to nonholonomic differential drive mobile robots. Even though the design of the navigation function in [20] is calculated for the pose configuration space (x, y, θ), it cannot be used for a differential drive mobile robot since it does not take into account the constraint on the lateral motion. To avoid local minima caused by the lateral motion constraint, here we introduce an additional cost term in the navigation function that ensures non-zero value of the gradient of the navigation function (outside of the goal state). Our navigation algorithm includes generation scheme of feasible control sequences that are then optimized according to the navigation function. Hence, our approach does not require switching controls or combination of several different control, as is the case in [13, 21, 22]. To the best of authors' knowledge, this is the first single-control-law navigation algorithm for differential drive mobile robots in environments with obstacles (non-convex constraint set) whose convergence is proved by applying the stability theory of discontinuous discrete-time nonlinear systems. The proposed RHC navigation algorithm produces robot trajectories that take less time to reach the goal than the state-of-the-art sample-based motion planning algorithm based on lattice graphs [26], with significantly lower computational costs. Furthermore, opposed to [26], the proposed RHC navigation algorithm does not depend on the map resolution, and it considers velocity and acceleration constraints in a systematic manner.

The paper is organized as follows. Formulation of the RHC navigation algorithm is given in Section II. The proposed navigation function is described in Section III. A generation scheme of feasible control sequences that are used in the RHC optimization is developed in Section IV. Section V gives the proof of the finite-time convergence to the goal state, while Section VI presents simulation and experimental test results.

II. Receding Horizon Control for Robot Navigation

In this Section we briefly review the discrete-time kinematic model of a differential drive mobile robot and present the concept of the proposed RHC navigation algorithm based on that model.

A. Robot's Kinematic Model and Constraints

The kinematic model of the differential drive mobile robot in the discrete-time form assuming that translational (v) and rotational (ω) velocity control inputs are constant within each sampling interval Δt is as follows [29]:

$$\begin{array}{rcl} x(t+1) & = & x(t) + v(t)\Delta t \cos\theta(t) \\ y(t+1) & = & y(t) + v(t)\Delta t \sin\theta(t) \\ \theta(t+1) & = & \theta(t) + \omega(t)\Delta t. \end{array} \tag{1}$$

where $t \in \mathbb{N}_0$, (x, y) are the Cartesian coordinates of the center of the robot rotation (the position of the mobile robot), and θ is the orientation of the robot with respect to the positive x axis. The time-invariant state space model of the robot motion can be written compactly as

$$s(t+1) = f(s(t), u(t)),\ t \in \mathbb{N}_0, \tag{2}$$

where the robot state and control input are defined by $s = (x, y, \theta) \in \mathcal{S} \subset \mathbb{R}^3$, and $u = (v, \omega) \in \mathcal{U} \subset \mathbb{R}^2$, respectively. $\mathcal{S}$ is the state constraint set containing collision free robot states, $\mathcal{U}$ is the velocity constraint set, and $f : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$ is the time-invariant nonlinear function given by (1).

At time t the control input $u(t)$ is subject to kinematic and dynamic constraints of the form

$$\begin{array}{rcl} u(t) & \in & \mathcal{U} = [0, v_{\lim}] \times [-\omega_{\lim}, \omega_{\lim}] \\ \dfrac{u(t) - u(t-1)}{\Delta t} & \in & \mathcal{A} = [-\dot{v}_{\lim}, \dot{v}_{\lim}] \times [-\dot{\omega}_{\lim}, \dot{\omega}_{\lim}] \end{array} \tag{3}$$

where $\mathcal{A}$ is the acceleration constraint set, while $(\cdot)_{\lim}$ denotes limits on velocities and accelerations. Note that the backward motion is forbidden because sensors' field of view is in the forward direction.

B. RHC Navigation Algorithm

We exploit the principle of the RHC, cf. [1], to develop an algorithm for convergent navigation of a differential drive mobile robot modeled with (1)–(3).

Let $\{s_k\}_0^N$ be the evolution of the robot state over a fixed horizon N (as a function of the control sequence $\{u_k\}_0^{N-1}$), according to the model (2), starting with $s_0 = s(t)$. At the current time t, and for the current state $s := s(t)$, the receding horizon control problem is the following:

$$P_N(s): \quad J^*(s) := \min_{\{u_k\}_0^{N-1}} J(\{s_k\}_0^N, \{u_k\}_0^{N-1}), \tag{4}$$

subject to:

$$\{u_k\}_0^{N-1} \in \mathcal{F}, \tag{5}$$

$$s_0 = s, \tag{6}$$

$$s_{k+1} = f(s_k, u_k), \quad \text{for } k = 0, \ldots, N-1, \tag{7}$$

$$\phi(s_N) \le \phi(s_k), \quad \text{for } k = 0, \ldots, N-1, \tag{8}$$

where the objective function J is given by

$$J(\{s_k\}_0^N, \{u_k\}_0^{N-1}) := \sum_{k=0}^{N} \phi(s_k) + \rho \sum_{k=0}^{N-1} \|u_k\|_1, \tag{9}$$

$\rho > 0$, J^* is the value function and ϕ is the navigation function measuring the cost-to-goal over states (cf. Section III), with the constraint (8) to have that the path cost at the last state does not increase, $\mathcal{F} \subset \mathbb{R}^2 \times \ldots \times \mathbb{R}^2 = \mathbb{R}^{2N}$ is the set of feasible control sequences that is computed to ensure tractable on-line optimization (cf. Section IV). Notice that the objective function encourages faster motion to the goal.

Any control sequence that attains the minimum in (4)–(9) is called an optimal control sequence, and is denoted with $\{u_k^*\}_0^{N-1}$. Without loss of generality we can assume that $\{u_k^*\}_0^{N-1}$ is unique. Then, the control applied to the system at time t is the first element of this sequence:

$$u(t) = u_0^*. \tag{10}$$

The same procedure (4)–(10) is repeated at the time $t+1$ for the new state $s(t+1)$. Since the model of the system and objective function are time invariant, this procedure defines a time invariant state feedback control $u(t) = \kappa(s(t))$, resulting with the closed-loop system $s(t+1) = f(s(t), \kappa(s(t)))$.

III. Navigation Function

In the rest of the paper, without loss of generality, we assume that the goal state is at the origin, i.e., $(x_G, y_G, \theta_G) = (0, 0, 0)$. The navigation function ϕ is derived from the D* graph search algorithm [30], which computes for every graph node the optimal path to the goal and the associated optimal cost, h, as the sum of weights along the path. The graph that is searched is created from the occupancy grid map representation of the environment. In particular, the whole environment, free space and obstacles, is divided into square cells of constant size e_{cell} (map resolution). For simplicity, we describe the mobile robot as a single point in the environment, while the real obstacles are enlarged for the integer number of cells $\lceil r_r / e_{\text{cell}} \rceil$, where r_r is the robot radius, to account for the robot's dimensions. This procedure is common in practice because it allows an easy detection of trajectory/obstacle collisions (see Section IV).

The set of nodes in the graph, denoted with $\mathcal{N}$, corresponds to the set of computed cells. With a slight abuse of the notation, in the following we will use index of the node in a graph also as a shorthand notation for the coordinates of the corresponding cell center. The weights $w_{i,j}$ between neighboring nodes i and j are defined as $w_{i,j} := e_{\text{cell}} \max(o(i), o(j))$, where $o \geq 1$ is the occupancy value. Similar to the potential field function, as suggested in [31], o has small values for nodes far from obstacles, higher values for nodes close to obstacles, and value ∞ for nodes at obstacles. The optimal path that is calculated on a graph defined in this manner will steer away from the obstacle cells, if possible. We chose 4-connected grid neighbors to avoid difficult maneuvers through diagonally connected cells between obstacles that can appear with 8-connected grid neighbors.

The next step is to create a continuous navigation function $\phi : \mathcal{S} \to [0, \infty\rangle$ from the discrete navigation function h computed by the D* search algorithm. A simple, straightforward extension where ϕ has the same value for every coordinate in the cell (equal to the cost h of the corresponding node, see the left part of Fig. 1) is not a good choice. In such a case there would be little distinction, in terms of the values of objective function (9), between two close trajectories. Furthermore, the discontinuity of ϕ would introduce unreasonable jumps in the penalty for trajectories that move to cells with higher cost. To overcome this problem we introduce a continuous simplex-based interpolation navigation function (see the right part of Fig. 1).

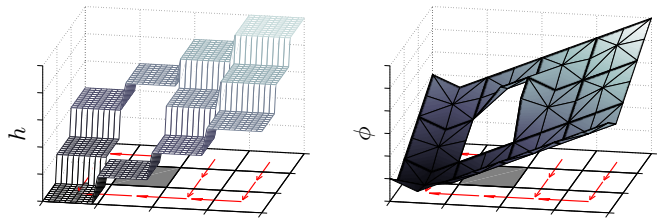

Fig. 1. 3D view of the cost h assigned to all points within a grid cell (left) and simplex-based interpolated cost ϕ (right). The grid map is drawn in the zero cost plane with one cell occupied (gray) and D* paths denoted with arrows.

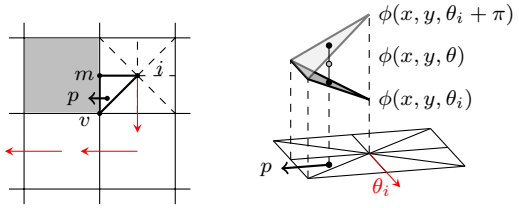

Fig. 2. Left: Division of a cell into eight triangles. Right: The cost ϕ at the state $p = (x, y, \theta)$ (θ is denoted with arrow) is interpolated between costs at vertices v, m, and i, where only at i the cost depends on the robot orientation θ with the smallest cost for orientation in the direction of the D* pointer θ_i.

The navigation function ϕ is built in the form of a simplicial complex composed of 2-simplexes (triangles) glued together [32]. Each cell is divided into eight equally shaped triangles with common vertex at the cell center (Fig. 2). Let $p = (x, y, \theta)$ be a state with location (x, y) contained in one of the eight triangles with vertices at the cell center $i = (x_1, y_1)$, the cell vertex $v = (x_2, y_2)$ and the mid point of the cell edge $m = (x_3, y_3)$. The cost $\phi(x, y, \theta)$ is calculated as the interpolation of the costs at the vertices i, v, and m:

$$\begin{aligned}
\phi(x, y, \theta) &= a_1\phi_1(i, \theta) + a_2\phi_2(v) + a_3\phi_3(m), \\
\phi_1(i, \theta) &= h(i) + \lambda o(i) \text{dist}(\theta, \theta_i), \\
\phi_2(v) &= \min_{j \in \mathcal{N}} \{h(j) + e_{\text{cell}} o(j) \;\big|\; \|v - j\| = \tfrac{\sqrt{2} e_{\text{cell}}}{2}\}, \\
\phi_3(m) &= \min_{j \in \mathcal{N}} \{h(j) + \tfrac{e_{\text{cell}}}{2} o(j) \;\big|\; \|m - j\| = \tfrac{e_{\text{cell}}}{2}\},
\end{aligned} \tag{11}$$

where θ_i is the orientation of the D* pointer, which points to the next cell on the optimal path (for the goal cell we use the desired orientation of the robot, $\theta_i = \theta_G$), $0 \leq \text{dist}(\theta, \theta_i) \leq \pi$ is the minimal distance between two orientations, and λ is the orientation weight defined as

$$\lambda = \frac{e_{\text{cell}}}{3\pi}. \tag{12}$$

The interpolation coefficients $0 \leq a_1, a_2, a_3 \leq 1$ are obtained by solving the simplex equations

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{13}$$

By construction, the vertices i, v, and m are not co-linear, hence the matrix in (13) is always invertible. In general, we can extend the domain of ϕ so that it has value ∞ at obstacles.

Lemma 1. *The navigation function ϕ, defined by* (11)–(13)*, is a continuous real-valued function that has a strict global minimum at the goal state and no (other) local minima.*

For brevity, we omit the proof of Lemma 1. We note that the lemma statement is equivalent to the claim that at each state, $s = (x, y, \theta) \neq 0$, there exists a feasible trajectory for (initially non-moving) differential drive robot that will lower the value of the navigation function ϕ. In particular, for a given state s one first finds a point in the current cell(s) that has the smallest value – the so-called *cell exit point*, denoted with g. If (x, y) belongs to the goal cell then $g = (0, 0)$, otherwise g is one of the 8 specific points available for every cell (x, y) belongs to: 4 vertices and 4 mid points of the edges of the cell. Now we can construct the following sequence of feasible control actions: i) rotate (from orientation θ) towards g; ii) translate from (x, y) to g; iii) if $g = (0, 0)$ then rotate towards θ_G and exit procedure, otherwise deduce the next exit-point, g^+, and rotate towards g^+; iv) make one translational step from g to g^+ with velocity $v = \min(\frac{e_{\text{cell}}}{2\Delta t}, \dot{v}_{\lim}\Delta t)$. The procedure i)–iv) defines a finite sequence of feasible control actions – henceforth referred to as the *cell exit control sequence*, denoted with $\{u_k^g(s)\}_0^{N_g-1}$. This control sequence moves the robot from the initial state s to a new state, $\tilde{s}$ in N_g steps. In general there is a complex dependence of N_g on s and system constraints, but one can compute a trivial upper bound

$$N_g \leq 2\left\lceil \frac{\pi}{\dot{\omega}_{\lim}(\Delta t)^2} \right\rceil + \left\lceil \frac{\sqrt{2}e_{\text{cell}}}{\dot{v}_{\lim}(\Delta t)^2} \right\rceil + 1. \tag{14}$$

Furthermore, we know that either $\tilde{s}$ is equal to the goal state, or we can can guarantee that $\phi(s) - \phi(\tilde{s}) \geq \Delta\phi$, for some fixed $\Delta\phi > 0$. Clearly, with a repetitive application of the above procedure, we can ensure that the robot moves from an arbitrary initial state s to the goal state in *finite time*. However, we can achieve better closed-loop system performance by considering an extended set of feasible control sequences.

IV. Feasible control sequences and robot trajectories

To solve the receding horizon control problem one would have to carry out the search for the optimal control sequences among all feasible ones of horizon N, which is a highly computationally demanding task and thus not suitable for the real-time implementation. Here this problem is circumvented by introducing a reduced set of feasible control sequences $\mathcal{F}$, resulting, in general, with a suboptimal solution.

The procedure of creating feasible control sequences is based upon the dynamic window approach (DWA) [33], which is an efficient trajectory search generation scheme. The search space is a discrete set of uniformly distributed velocities around the current robot velocity that accounts for kinematic and dynamic constraints of the robot. The DWA considers constant velocities (v, ω) for N time intervals, and thus defines circular trajectories. The optimal trajectory is found by maximizing the objective measure composed of several different criteria [34, 35]. The first element of the optimal trajectory defines the control move u^* that is applied to the robot. The search is repeated at each sampling instant. Therefore, the DWA can be considered as a special case of the RHC. However, the constraints in the DWA are not satisfied for the full horizon N since trajectories can contain collisions.

The proposed algorithm for creating feasible control sequences ensures that all constraints are satisfied along the horizon N. Each control sequence defines a circular arc shaped collision-free trajectory, whose velocity is zero at the end of the prediction horizon (see Fig. 3). Stopping at the end of the prediction horizon is necessary for proving the finite-time convergence of the closed-loop system, while keeping the computational requirements low.

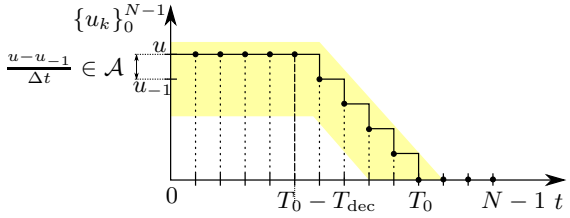

Fig. 3. An example of the created control sequence in $\mathcal{F}$ for fixed u and T_0. Shaded area represents other trajectories in $\mathcal{F}$.

The generation procedure of the set of feasible control sequences $\mathcal{F}$ is given by Alg. 1. The trajectory generation depends on the current state $s = s(t)$ and on the last optimal control sequence at time $t-1$, noted as $\{u_k^{*\text{old}}\}_0^{N-1}$, which is initially filled with zeros. The first element of the last control sequence, which was applied to the system at time $t-1$, becomes the current velocity u_{-1}, and determines possible control inputs that can be applied at time t, according to the dynamic and kinematic constraints (3) (Alg. 1, line 4), where $\mathcal{A}$ is uniformly quantized set of accelerations. The first zero velocity element of the last optimal control sequence defines the time index T_0^{old}, which determines the stopping time T_0 of the control sequences at the current time t. We have constrained T_0 to four values $\{T_0^{\text{old}}-2, T_0^{\text{old}}-1, T_0^{\text{old}}, T_0^{\text{old}}+1\}$ assuming that the optimal solution will be close to the last optimal solution. If $T_0 = T_0^{\text{old}} - 1$, we can get a state trajectory completely aligned with the old state trajectory starting from the new state $s_0 = s_1^{*\text{old}}$ and repeating the last state. With $T_0 = T_0^{\text{old}} - 2$ or $T_0 \geq T_0^{\text{old}}$ we can get shorter or longer state trajectories, respectively. With $T_0 = T_0^{\text{old}} + 1$ we enable extension of the non-zero control sequence towards N. Starting with the zero trajectory, stopping time T_0 of the control sequence can increase from 0 to full horizon length N during motion, or again decrease among obstacles to obtain shorter trajectories.

The control sequence $\{u_k\}_0^{N-1}$ is calculated such that the start velocity is the chosen control input $u = [v, \omega]$, which is being held and then ramp down until the velocity reaches zero

Algorithm 1 Generation of $\mathcal{F} = \mathrm{traj}(s(t), \{u_k^{*\mathrm{old}}\}_0^{N-1})$

Require: Current state $s(t)$, the last optimal control sequence $\{u_k^{*\mathrm{old}}\}_0^{N-1}$ (initially $u_k^{*\mathrm{old}} = 0$, for $k = 0, \ldots, N-1$)
Ensure: The set $\mathcal{F}$ of control sequences $\{u_k\}_0^{N-1}$

1: $s_0 \leftarrow s(t),\ \mathcal{F} \leftarrow \emptyset$ // *initialization*
2: $T_0^{\mathrm{old}} \leftarrow \min k \mid u_k^{*\mathrm{old}} = 0,\ k = 0, \ldots, N-1$ // *zero-time*
3: $u_{-1} \leftarrow u_0^{*\mathrm{old}}$ // *last applied control input*
4: **for** $\forall u \in \mathcal{U}$ such that $\dfrac{u - u_{-1}}{\Delta t} \in \mathcal{A}$ **do**
5: $\quad u = (v, \omega)$
6: $\quad T_{\mathrm{dec}} \leftarrow \max\left\{ \left\lceil \dfrac{v}{\dot{v}_{\lim} \Delta t} \right\rceil, \left\lceil \dfrac{|\omega|}{\dot{\omega}_{\lim} \Delta t} \right\rceil \right\}$ // *decelerating*
7: $\quad$ **for** $\forall T_0 \in \{T_0^{\mathrm{old}} - 2, T_0^{\mathrm{old}} - 1, T_0^{\mathrm{old}}, T_0^{\mathrm{old}} + 1\} \mid T_{\mathrm{dec}} \leq T_0 < N$ **do**
8: $\quad\quad \{u_k\}_0^{N-1} \leftarrow \begin{cases} u & |k \leq T_0 - T_{\mathrm{dec}}, \\ u\dfrac{T_0 - k}{T_{\mathrm{dec}}} & |T_0 - T_{\mathrm{dec}} < k < T_0, \\ 0 & |T_0 \leq k \leq N-1, \end{cases}$
9: $\quad\quad$ **if** $|u_k| < (\Delta v, \Delta\omega)$ for $k = 0, \ldots, N$ **then**
10: $\quad\quad\quad u_k = (0, 0)$ // *velocity dead zone*
11: $\quad\quad$ **end if**
12: $\quad\quad s_{k+1} = f(s_k, u_k)$ for $k = 0, \ldots, N-1$
13: $\quad\quad$ **if** $\phi(s_k) < \infty$ for $k = 0, \ldots, N$ **then**
14: $\quad\quad\quad \mathcal{F} \leftarrow \mathcal{F} \cup \{u_k\}_0^{N-1}$ // *obstacle free trajectories*
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end for**

at the step T_0. The duration of the ramp down is calculated according to the limit deceleration that can be applied to the robot, and is noted as T_{dec} (Alg. 1, line 6). Since both v and ω decrease with the same negative slope, the control sequence produces points on circular arc of radius $r = \frac{v}{\omega}$ (Alg. 1, line 8). By this procedure, all control inputs satisfy constraints (3). Finally, a sequence $\{u_k\}_0^{N-1}$ is inserted into the set $\mathcal{F}$ if its trajectory is without collision with obstacles, which is true if all trajectory points have $\phi < \infty$ (Alg. 1, line 13).

V. Convergence of the Navigation Algorithm

Lemma 2. *Let the set $\mathcal{S}_N$ be defined as $\mathcal{S}_N = \{s \in \mathcal{S} \mid \phi(s) < \infty\}$. Then, $\forall s \in \mathcal{S}_N$ there exist feasible state and control sequences for the RHC problem $P_N(s)$ in* (4)–(9)*. Furthermore, the set $\mathcal{S}_N$ is positively invariant for the closed-loop system $s^+ = f(s, \kappa(s))$.*

Proof. We need to show that for any $s \in \mathcal{S}_N$ the set $\mathcal{F}$ contains a trajectory which satisfies the constraint (8). At the beginning of the motion there exists a trajectory in the set $\mathcal{F}$ with $T_0 = 1$ that produces minimal translation or rotation. We have already shown that rotation or translation in the direction of the D* pointer lowers the cost ϕ (cf. Lemma 1) and therefore (8) is satisfied. Denote the optimal control sequence and corresponding state sequence at time t for the state $s = s(t) \in \mathcal{S}_N$ as follows:

$$\{u_k^*\}_0^{N-1} = \{u_0^*, u_1^*, \ldots, u_{N-1}^*\}, \tag{15}$$

$$\{s_k^*\}_0^{N} = \{s_0^*, s_1^*, \ldots, s_N^*\}, \tag{16}$$

where $s_0^* = s$, and $\{s_k^*\}_0^N$ satisfies (8). Then the control move applied to the system at time t is the first element of (15), i.e., $u(t) = \kappa(s) = u_0^*$. Assume that at time $t+1$ the robot moves to the state s_1^*, i.e., $s^+ = f(s, \kappa(s)) = f(s_0^*, u_0^*) = s_1^*$. At time $t+1$, there exists a control sequence $\{u_k^+\}_0^{N-1} \in \mathcal{F}$, with starting control move $u_0^+ = u_1^*$, and the parameter $T_0 = T_0^{\mathrm{old}} - 1$ that will produce the state trajectory $\{s_k^+\}_0^N$ starting from the s_1^* and repeating the last state at s_N^* (cf. Alg. 1):

$$\{u_k^+\}_0^{N-1} = \{u_1^*, \ldots, u_{N-1}^*, 0\}, \tag{17}$$

$$\{s_k^+\}_0^{N} = \{s_1^*, \ldots, s_N^*, s_N^*\}. \tag{18}$$

If trajectory $\{s_k^*\}_0^N$ satisfies (8), it is clear that $\{s_k^+\}_0^N$ also satisfies (8). Therefore control sequence (17), and corresponding state sequence (18) are feasible for the RHC problem $P_N(s^+)$ in (4)–(9) and hence $s^+ \in \mathcal{S}_N$. By this it is shown that $\mathcal{S}_N$ is positively invariant for the closed-loop system $s^+ = f(s, \kappa(s))$. □

Theorem 3. *Let the system $s(t+1) = f(s(t), u(t))$ be controlled by the receding horizon algorithm* (4)–(10)*, i.e. $u(t) = \kappa(s(t))$ except when $\kappa(s(t)) = 0$. If $\kappa(s(t)) = 0$ is detected at time t_0 execute the full cell exit control sequence, i.e., $u(t_0 + k) = u_k^g(s(t_0))$, for $k = 0, \ldots, N_g - 1$, before switching back to $u(t) = \kappa(s(t))$ for $t \geq t_0 + N_g$. Then the closed-loop system converges to the origin in finite-time from any s in $\mathcal{S}_N$.*

For brevity we omit the proof of Theorem 3. It relies on the use of the value function J^* as a Lyapunov function. To prove that the origin is reached in finite-time from any $s \in \mathcal{S}_N$ one must show that whenever the RHC control (4)–(10) is used (i.e., $\kappa(s) \neq 0$) the Lyapunov function J^* satisfies the following property:

$$J^*(s) - J^*(f(s, \kappa(s))) \geq \Delta J^*, \tag{19}$$

for some fixed $\Delta J^* > 0$. In particular, we can show that $J^*(s) - J^*(f(s, \kappa(s))) \geq \rho \|u_0^*(s)\|_1$. Thus, by limiting the smallest allowed non-zero value for $u = (v, \omega)$ in the generation of $\mathcal{F}$, as is achieved by a velocity dead zone (Alg. 1, line 9), we guarantee that $\Delta J^* \geq \rho \min\{\Delta v, \Delta\omega\} > 0$.

VI. Test results

A. Simulations

The proposed RHC navigation algorithm was tested and compared to the state-of-the-art sample-based motion planning algorithm based on the lattice graph [26] (SBPL package under ROS [36] was used) on different shaped maps U-shaped, Corridor, and S-shaped maps to test the algorithm behavior in case of local minima and symmetry, behavior in very narrow passages, and sharp (slalom) turns, respectively. The SBPL algorithm generates a shortest path by combining a series of *motion primitives*, which are feasible motions with constant velocity that satisfy kinematic constraints. The path is followed by DWA. Both algorithms used higher costs around obstacles to push the planned path for 0.3 m between the robot contour and obstacles if possible, where the robot diameter is 0.5 m. Limits on translation and rotation velocities and

accelerations were set to $v_{\lim} = 1$ m/s, $\omega_{\lim} = 100$ °/s, $\dot{v}_{\lim} = 0.6$ m/s^2, $\dot{\omega}_{\lim} = 100$ °/s^2, and the length of horizon $N = 50$. Duration of the time step is $\Delta t = 0.1$ s. The smallest allowed non-zero velocities were set to $\Delta v = 0.006$ m/s and $\Delta\omega = 1$ °/s. We purposely put low acceleration limits to make maneuvering more difficult. We used mostly default parameter configurations for the SBPL algorithm: motion primitives for unicycle without backward and side motions, 8 samples for translational velocity space and 20 samples for rotational velocity space, while in the RHC algorithm we used coarse velocity sampling: 3 samples for translational and 3 samples for rotational velocity space, i.e., acceleration set comprises only minimal and maximal values. Coarse sampling is chosen to keep the computations of control inputs per time step below 10 ms, while with more velocity samples RHC obtains slightly faster motion. On the contrary, high sampling for SBPL influence on better tracking of the planned path, with slightly increased control computations to about 20 ms. The RHC algorithm is stopped when the robot reaches the goal cell and orients within 5 ° from the goal orientation. It needs to be mentioned that in all simulation tests and experiments the RHC algorithm never switched to the cell exit control sequence.

Table I presents the results obtained by the RHC algorithm and the SBPL algorithm on all three map shapes for the grid resolution of 10 cm, 5 cm and 2.5 cm. It can be seen that for all three grid resolutions both algorithms generate similar trajectory lengths, but RHC gives around 10% shorter goal reaching times. For the sake of visual comparison, robot trajectories and velocity profiles on all three map shapes for the grid resolution of 10 cm are shown in Fig. 4 and Fig. 5, respectively. In terms of calculation times per time step, we compared both algorithms according to the path planning time, since this is the major calculation defining the algorithm behavior in changing environments, while the rest of computations for control inputs depend only on the number of velocity samples used for trajectory generation. RHC algorithm significantly outperforms SBPL algorithm (at least 20 times[2]). The values of planning times clearly show that, contrary to the SBPL algorithm, the RHC algorithm is suitable for real-time navigation in changing environments. Because of that, we conducted experimental tests on a real robot only with the RHC algorithm.

TABLE I
COMPARISON OF RHC AND SBPL ACCORDING TO THE MAP RESOLUTION

	resolution	10 cm	5 cm	2.5 cm
U-shaped map		planning time [ms] (expanded nodes [#])		
	RHC	6 (7,402)	19 (30,648)	48 (123,184)
	SBPL	180 (24,713)	420 (54,812)	1,330 (175,368)
		trajectory length [m] (goal reaching time [s])		
	RHC	12.42 (17.6)	12.21 (17.1)	12.67 (17.1)
	SBPL	12.59 (19.0)	12.32 (18.9)	12.14 (17.8)
Corridor map		planning time [ms] (expanded nodes [#])		
	RHC	5 (3,934)	17 (17,438)	38 (69,708)
	SBPL	110 (13,461)	370 (54,718)	1,340 (191,728)
		trajectory length [m] (goal reaching time [s])		
	RHC	13.91 (23.0)	13.95 (22.4)	14.0 (22.5)
	SBPL	13.19 (25.1)	13.25 (25.2)	13.18 (24.9)
S-shaped map		planning time [ms] (expanded nodes [#])		
	RHC	1 (1,326)	8 (6,120)	21 (25,022)
	SBPL	70 (11,403)	230 (37,554)	820 (117,032)
		trajectory length [m] (goal reaching time [s])		
	RHC	8.34 (12.4)	8.49 (12.7)	8.64 (13.7)
	SBPL	7.77 (14.7)	7.44 (12.3)	7.12 (12.4)

B. Experiments

The experiments were done on the Pioneer P3-DX differential drive mobile robot in a known environment populated with unknown and dynamically changing obstacles. The experiments were repeated for a long list of randomly chosen goal states, and three of them were chosen here for the algorithm representation (Fig. 6). The occupancy grid map is created from the given map with resolution 10 cm, and the laser range data are used for updating the occupancy grid map needed for path cost re-calculation. Limits on translation and rotation velocities and accelerations used in experiments were set to $v_{\lim} = 0.5$ m/s, $\omega_{\lim} = 50$ °/s, $\dot{v}_{\lim} = 0.3$ m/s^2, $\dot{\omega}_{\lim} = 100$ °/s^2. Each update of the occupancy grid map initiates D* calculation for new D* pointers and cost values needed for the navigation function creation. In case of no path, the robot follows the chosen trajectory to the end and tries again. One such case can be noticed on the way to the goal 3 (Fig. 6), in which moving obstacle blocked completely the path to the goal (the spot where the trajectory changes color from blue to green). Velocity profile, Lyapunov function and planning times during motion are shown in Fig. 7. The maximal translational velocity is achieved most of the time, even in turning maneuvers and regardless of re-planning cases. The changes of costs ϕ in the robot vicinity due to re-planning for obstacle avoidances are reflected in abrupt changes of J^* (increase or decrease, depending on the change in occupancy). However, between re-planning cases the value of J^* always decreases.

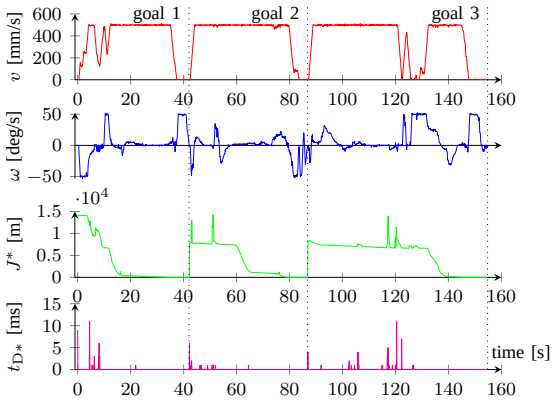

Fig. 7. Velocity profiles, Lyapunov function and D* planning times during motion between three goals. Jumps of Lyapunov function can be noticed at time steps in which re-planning time is not zero.

[2]Planning times were measured in optimization mode (opt with -03 flag on GNU C compiler) on Intel Core i5-2557M CPU @ 1.70GHz processor

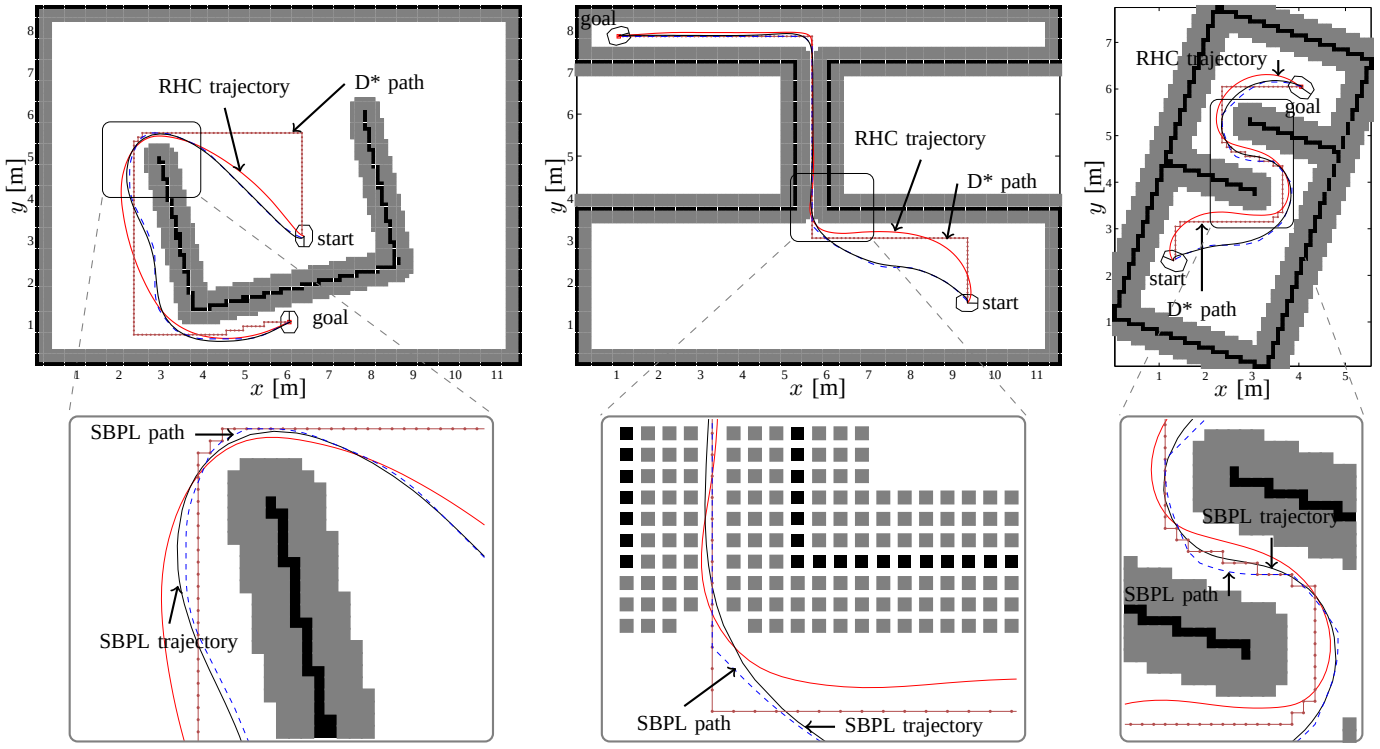

Fig. 4. Robot trajectories in the U-shaped (left), Corridor (middle), and S-shaped (right) maps with grid resolution of 10 cm and magnified part of each map (down row) with SBPL trajectory (solid black) and path (dashed blue) and RHC trajectory (solid red) and path (dotted red).

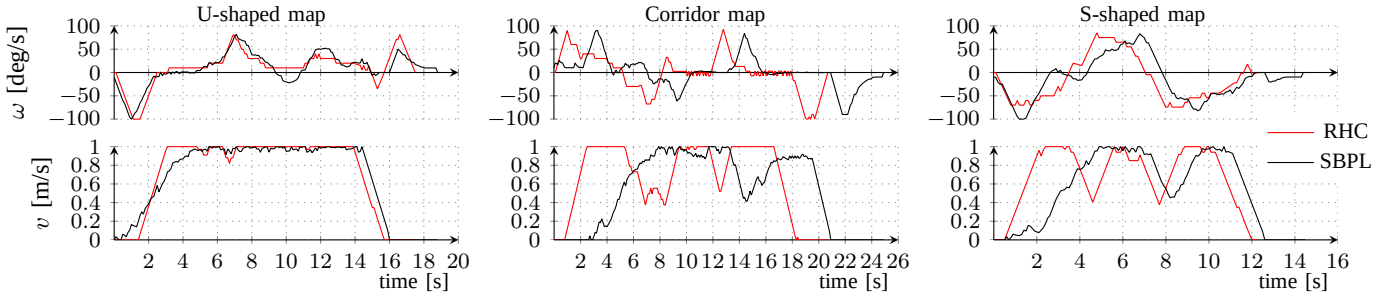

Fig. 5. Velocity profiles of the RHC and SBPL trajectory for the U-shaped, Corridor and S-shaped map for grid resolution of 10 cm.

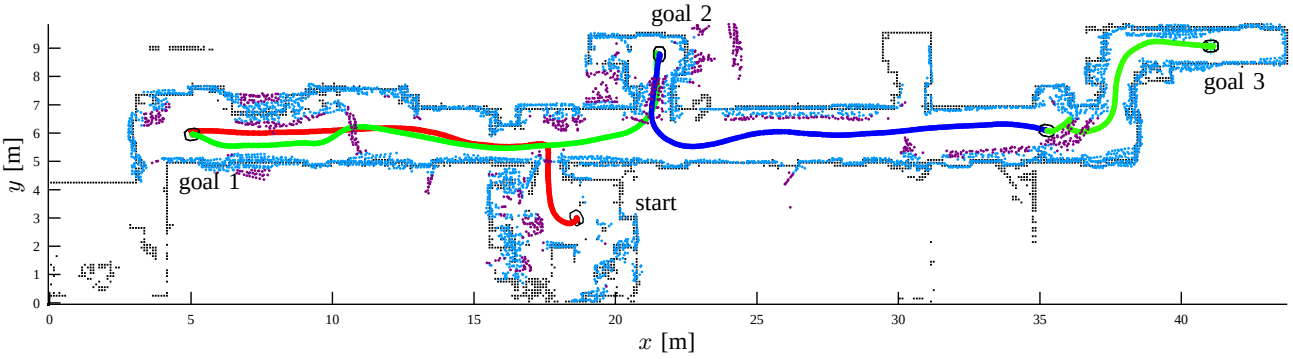

Fig. 6. Robot trajectories during motion between three goals and accumulated laser readings representing static (blue) and dynamic (purple) obstacles.

VII. Conclusion

We presented a receding horizon control based navigation algorithm for a differential drive mobile robot. The objective function is proposed that integrates a navigation function and local robot trajectories. It measures the contribution of each control sequence with respect to the navigation function obtained by interpolation of the D* path costs in the grid.

The optimal value of the objective function is employed as a Lyapunov function to prove the finite-time convergence of the closed-loop system to the goal state. The test results show good performances of the proposed RHC navigation algorithm even for a very coarse set of control sequences and/or robot motion in changing environments. We compared the RHC algorithm to the state-of-the-art SBPL algorithm with respect to the trajectory lengths and the time needed to reach the goal state. Although the SBPL has somewhat shorter trajectories, the RHC has about 10% shorter goal reaching times and at least 20 times faster path computations. Furthermore, SBPL is highly dependent on the DWA scaling parameters, which a user should tune, and does not have the finite-time convergence guarantee. On the other hand, the proposed RHC navigation algorithm uses single criterion based on the navigation function, without need of any parameter tuning, and considers velocity and acceleration constraints in a systematic manner.

ACKNOWLEDGMENT

This research has been partly supported by the Ministry of Science, Education and Sports of the Republic of Croatia under the grant "Centre of Research Excellence for Data Science and Cooperative Systems".

REFERENCES

[1] G. C. Goodwin, M. M. Seron, and J. A. De Doná, *Constrained control and estimation: an optimisation approach*. Springer Verlag, 2004.

[2] P. Mellodge and P. Kachroo, *Model abstraction in dynamical systems: application to mobile robot control*, m, Ed. Springer Verlag, 2008.

[3] R. Brockett, "Asymptotic stability and feedback stabilization," *Differential Geometry Control Theory*, pp. 181–191, 1983.

[4] F. A. C. C. Fontes, "Discontinuous feedbacks, discontinuous optimal controls, and continuous-time model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 13, no. 3-4, pp. 191–209, 2003.

[5] M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, "Closed loop steering of unicycle like vehicles via lyapunov techniques," *Robotics & Automation Magazine, IEEE*, vol. 2, no. 1, pp. 27–35, 1995.

[6] F. Kuhne, W. Lages, and J. da Silva Jr, "Point stabilization of mobile robots with nonlinear model predictive control," in *Mechatronics and Automation, 2005 IEEE International Conference*, vol. 3. IEEE, 2005, pp. 1163–1168.

[7] D. Gu and H. Hu, "A stabilizing receding horizon regulator for nonholonomic mobile robots," *Robotics, IEEE Transactions on*, vol. 21, no. 5, pp. 1022–1028, 2005.

[8] F. Xie and R. Fierro, "First-state contractive model predictive control of nonholonomic mobile robots," in *American Control Conference, 2008*. IEEE, 2008, pp. 3494–3499.

[9] Z.-Y. Liang and C.-L. Wang, "Robust stabilization of nonholonomic chained form systems with uncertainties," *Acta Automatica Sinica*, vol. 37, no. 2, pp. 129–142, 2011.

[10] B. Li, H. Chen, and J. Chen, "Global finite-time stabilization for a class of nonholonomic chained system with input saturation," *Journal of Information & Computational Science*, vol. 11, no. 3, pp. 883–890, 2014.

[11] A. Eqtami, S. Heshmati-alamdari, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Self-triggered model predictive control for nonholonomic systems," in *Control Conference (ECC), 2013 European*. IEEE, 2013, pp. 638–643.

[12] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, pp. 789–814, 2000.

[13] S. Lindemann and S. LaValle, "Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 600–621, 2009.

[14] O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[15] 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, 1992.

[16] H. G. Tanner and J. L. Piovesan, "Randomized receding horizon navigation," *Automatic Control, IEEE Transactions on*, vol. 55, no. 11, pp. 2640–2644, 2010.

[17] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*, m, Ed. Cambridge, Massachusetts: MIT Press, 2005.

[18] J. Latombe, *Robot Motion Planning*, m, Ed. Dodrecht, Netherlands: Kluwer Academic Publishers, 1991.

[19] P. Ogren and N. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, 2005.

[20] M. Đakulović (Seder), C. Sprunk, L. Spinello, I. Petrović, and W. Burgard, "Efficient navigation for anyshape holonomic mobile robots in dynamic environments," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 2644–2649.

[21] L. Adouane, A. Benzerrouk, and P. Martinet, "Mobile robot navigation in cluttered environment using reactive elliptic trajectories," in *18th IFAC World Congress*, 2011, pp. 13 801–13 806.

[22] L. Valbuena and H. G. Tanner, "Hybrid potential field based control of differential drive mobile robots," *Journal of intelligent & robotic systems*, vol. 68, no. 3-4, pp. 307–322, 2012.

[23] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1478–1483.

[24] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5054–5061.

[25] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Optimal, smooth, nonholonomic mobile robot motion planning in state lattices," *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-15*, 2007.

[26] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[27] A. M. Lekkas and T. I. Fossen, "Integral LOS path following for curved paths based on a monotone cubic hermite spline parametrization," *Control Systems Technology, IEEE Transactions on*, vol. 22, no. 6, pp. 2287–2301, 2014.

[28] L. Gracia, F. Garelli, and A. Sala, "Reactive sliding-mode algorithm for collision avoidance in robotic systems," *Control Systems Technology, IEEE Transactions on*, vol. 21, no. 6, pp. 2391–2399, 2013.

[29] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*, m, Ed. Springer Verlag, 2008.

[30] A. Stentz, "Optimal and efficient path planning for partially-known environments," *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3310–3317, 1994.

[31] M. Đakulović (Seder) and I. Petrović, "Two-way D* algorithm for path planning and replanning," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 329–342, 2011.

[32] J. Sack and J. Urrutia, *Handbook of computational geometry*, m, Ed. North-Holland, 2000.

[33] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics & Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23–33, 1997.

[34] O. Brock and O. Khatib, "High-speed navigation using the Global Dynamic Window Approach," *ICRA'99, IEEE International Conference on Robotics and Automation*, 1999.

[35] B. P. Gerkey and K. Konolige, "Planning and control in unstructured terrain," in *ICRA Workshop on Path Planning on Costmaps*, 2008.

[36] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.