Combined stochastic-deterministic predictive control using local-minima free navigation

Gregor Klančar and Marija Seder, *Member, IEEE*

***Abstract*— This work proposes a model predictive control approach of a wheeled mobile robot based on a local-minima free navigation function. The constructed navigation function includes information on a goal location and obstacles. Novel conservative navigation is introduced that is simple to compute and yields convergent control behavior. To solve the optimization problem the combined optimization is proposed by a fixed candidate set and particle swarm optimization. The efficiency of the proposed approaches is validated on simulations, by experimental results on the Husky A200 mobile robot and comparisons to Stable Sparse RRT planner.**

I. INTRODUCTION

Operation in occupied dynamically changing environments is a common scenario in mobile robots. Several navigation strategies were proposed that include finding a solution to the global path planning and a feedback controller to drive the robot on the path. Usual approaches tackle the task complexity by separating trajectory planning from the control. Graph search algorithms based on a discrete grid map of the environment can produce computationally efficient paths [1]. But additional path smoothing is needed by further path optimizations or motion primitives inserts to make such a path drivable by a wheeled robot such as in [2], [3], [4], [5]. A smooth path can as well be obtained directly by considering a discrete set of kinodynamic motion primitives as in [6], [7], [8], [9], [10], [11], [12].

During robot control on a previously defined path, any tracking error (initial, transient control), any disturbance in robot operation, or any environment change may result in the previous reference path not being suitable anymore. Therefore navigation approaches are preferred where planning and control are combined in receding dynamic window approaches or trajectory roll-out algorithms considering a convergent navigation function such as in [13], [14], [15]. Here the obtained performance depends on a control law and used objective function which needs to incorporate an environment with mapped obstacles to find feasible optimal trajectories in a prediction horizon. In contrast to the MPC-based algorithms, the RRT-based algorithms ([9], [10], [11]) usually calculates the whole trajectory to the goal before robot starts to execute it.

The main contributions of this work are twofold. Firstly, we propose collision safe navigation functions that are local-minima free with the minimum in the goal location. Note that any optimal discrete graph search algorithm combined with an appropriate interpolation can provide a continuous navigation with a single minimum [13]. We derived them by a smooth polynomial interpolation and by a conservative time-step approach from a discrete potential function. The former has a smoother surface while the latter features simple computation and guarantees convergent control behavior.

Secondly, contribution relates to the novel optimization strategy for optimal controls determination in the applied model predictive control. Optimal control actions are efficiently searched by a combined deterministic and stochastic approach. It upgrades the optimization with a defined fixed set of control action candidates as proposed in [15] with a particle swarm optimization. Proposed approaches are illustrated in several simulations and by experimental results.

G. Klančar is with the Department of System, Control and Cybernetics, Faculty of Elecrical Engineering, University of Ljubljana, Slovenia e-mail: gregor.klancar@fe.uni-lj.si.

M. Seder is with the Department of Control and Computer Engineering, Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia e-mail: marija.seder@fer.hr.

II. NAVIGATION FUNCTION

Navigation function N is used to safely drive a wheeled mobile robot among obstacles towards the goal. A control algorithm, therefore, controls the robot towards locations where N decreases. The unimodal potential function is a viable choice for N as it has a single minimum ($N = 0$) at the goal and without local minima where the control algorithm may get trapped. Additionally, N needs to have the highest values at the obstacles. A graph search algorithm such as D* for dynamic environments can be used to obtain such a potential function U as shown in Fig. 1. The value of U represents the distance to the goal cell computed as the sum of inter-cells distances (d_c) along the path. Such a search is computationally efficient since it is performed on a discrete grid of the environment but is not appropriate for a control algorithm since U is constant for arbitrary robot pose inside a discrete cell. Therefore the grid-based navigation function needs to be modified to obtain unique value for any position within a cell, which retain a single minima property [13]. In the following, we propose a bicubic interpolation and a conservative time-step approach.

A. Bicubic interpolation

Bicubic interpolation is used to a obtain smooth potential $P(x, y)$ from a discrete potential $U(x, y)$. According to a given arbitrary point $[x, y]^T$ location in a cell $\mathbf{N}$ locate four neighbor cells where centers form a square as illustrated in Fig. 2 by a dashed line. Then normalized coordinates inside the selected square are $x_n = \frac{x-x_0}{d_c}$, $y_n = \frac{y-y_0}{d_c}$ where $[x_0, y_0]$ is the lower-left corner of the square. Potential is

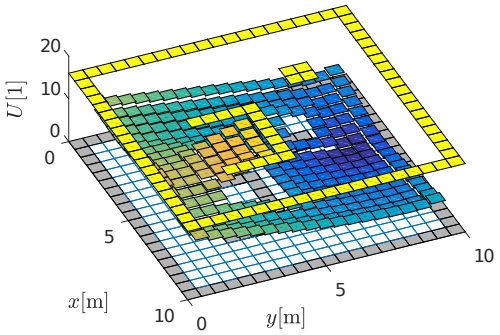


Fig. 1. 3D view of discrete potential function obtained from a grid-based search with 0.5 m resolution, a goal position at $x = 4.25$ m, $y = 8.25$ m and occupied cells with $U = \infty$ (set to $U = 18$ for clarity).

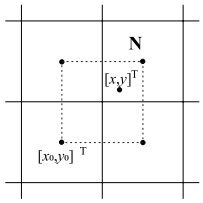


Fig. 2. Region selection for bicubic interpolation of potential based on a point $[x, y]^T$ location in cell $\mathbf{N}$.

expressed as $P_n(x_n, y_n) = P(x, y)$. For the neighbor cells centers define potential and its partial derivatives

$$p_{rc} = U_n(x_n, y_n)\Big|_{x_n=r,\ y_n=c}$$

$$f_{xrc} = \frac{\partial P_n}{\partial x_n}\Big|_{x_n=r,\ y_n=c} \approx \frac{U_n(r+1,c) - U_n(r-1,c)}{2}$$

$$f_{yrc} = \frac{\partial P_n}{\partial y_n}\Big|_{x_n=r,\ y_n=c} \approx \frac{U_n(r,c+1) - U_n(r,c-1)}{2}$$

$$f_{xyrc} = \frac{\partial^2 P_n}{\partial x_n \partial y_n}\Big|_{x_n=r,\ y_n=c} \approx \frac{U_n(r+1,c+1) - U_n(r-1,c+1) - U_n(r+1,c-1) - U_n(r-1,c-1)}{4},$$

where $r, c \in \{0, 1\}$ and $U_n(x_n, y_n) = U(x, y)$.

For a given arbitrary position the interpolated potential is then defined using bicubic interpolation as follows

$$P_n(x_n, y_n) = \begin{bmatrix}1 & x_n & x_n^2 & x_n^3\end{bmatrix} \mathbf{A} \begin{bmatrix}1 & y_n & y_n^2 & y_n^3\end{bmatrix}^T, \quad (1)$$

where matrix of coefficients is

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & f_{y00} & f_{y00} \\ p_{10} & p_{11} & f_{y10} & f_{y11} \\ f_{x00} & f_{x01} & f_{xy00} & f_{xy01} \\ f_{x10} & f_{x11} & f_{xy10} & f_{xy11} \end{bmatrix} \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

And negative gradient of P in $[x, y]^T$ is computed in a closed-form as

$$\begin{aligned} \overrightarrow{\mathbf{s}}(x,y) &= -\nabla P(x,y) = -\left[\frac{\partial P(x,y)}{\partial x}, \frac{\partial P(x,y)}{\partial y}\right]^T = \\ &= -\frac{1}{d_c}\left[\frac{\partial P(x_n,y_n)}{\partial x_n}, \frac{\partial P(x_n,y_n)}{\partial y_n}\right]^T \end{aligned}.$$

Final navigation function depends on $P(x, y)$ and on the orientation φ

$$\begin{aligned} N(x,y,\varphi) &= P(x,y) + \xi e(\varphi) \\ e(\varphi) &= \min_{k=\{0,1,-1\}} |\angle \overrightarrow{\mathbf{s}}(x,y) - \varphi + 2k\pi|, \end{aligned} \quad (2)$$

where $e(\varphi)$ is absolute orientation error, $\xi > 0$ and $\angle \overrightarrow{\mathbf{s}}(x, y)$ is negative gradient orientation.

An example of interpolated navigation function with a smooth surface is shown in Fig. 3. A closer look reveals some small anomalies near the obstacles such as surface oscillations and discontinuities in gradient direction which may affect control performance as discussed in [15]. These anomalies appear due to the estimation of partial derivatives f at obstacles borders or corners and cubic polynomial approximation.

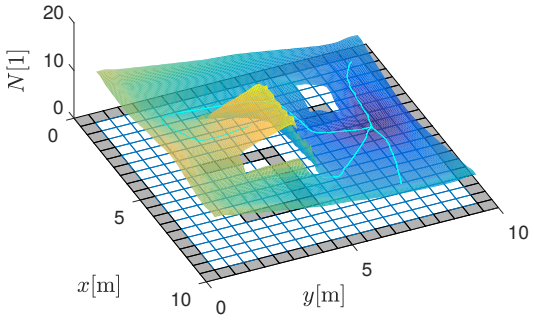


Fig. 3. 3D view of the interpolated navigation function using bicubic interpolation of P with $e(\varphi)$ set to zero for a clear presentation. Four paths are drawn from starting locations following the negative gradient.

B. Conservative time-step approach

As a conservative approach, we selected the criterion that takes into account the worst-case motion consisting of translation and rotation to move across the cells and reach the goal. We defined the so-called time-step function $\tau(x, y, \varphi)$ which measures the number of discrete control steps that robot needs to perform from its current state $[x, y, \varphi]^T$ to the goal $\mathbf{G} = [x_G, y_G]^T$ to obtain a motion through the exit points $\mathbf{E_N} = [x_E, y_E]^T$ of each cell $\mathbf{N}$, $U(\mathbf{N}) < \infty$. The exit point of the cell $\mathbf{N}$ is defined according to the negative gradient of a discrete potential $U(x, y)$. This exit point is at the border between the cells $\mathbf{N}$ and $\mathbf{M}$, where $\mathbf{M}$ is the next cell on the path followed by gradient descent (see Fig. 4). The negative gradient $\mathbf{p}(\mathbf{N})$ for the cell $\mathbf{N}$ can be estimated from the neighbor cells $\mathbf{A}$ and $\mathbf{B}$ with the lowest potential along y and x axis, respectively, as follows

$$\overrightarrow{\mathbf{p}}(\mathbf{N}) = \begin{bmatrix} (U(\mathbf{N}) - U(\mathbf{B})) \cdot \dfrac{x_B - x_N}{d_c} \\ (U(\mathbf{N}) - U(\mathbf{A})) \cdot \dfrac{y_A - y_N}{d_c} \end{bmatrix}. \quad (3)$$

The exit point of the goal cell is the goal position, i.e.,

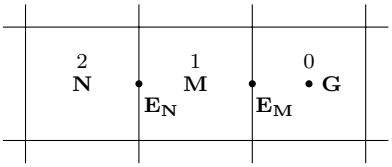


Fig. 4. An example of exit points at the cell borders on the path to the goal, where the cost of the path U is noted by a number inside a cell.

$\mathbf{E_G} = [x_G, y_G]^T$. For all other cells, the exit point is defined

as the intersection of the line defined with the gradient vector and the point within the cell, and the line segment on the cell borders.

The time-step function τ for the robot position $[x, y]^T$ inside the cell $\mathbf{N}$, $U(\mathbf{N}) < \infty$, and for the robot orientation φ is defined as follows:

$$\begin{aligned} \tau(x, y, \varphi) &= \frac{e(\varphi)}{\Delta\omega T_s} + \frac{||[x, y]^T - \mathbf{E_N}||}{\Delta v T_s} + \overline{\tau}(\mathbf{E_N}) \\ e(\varphi) &= \min_{k=\{0,1,-1\}} |\angle(x_E - x, y_E - y) - \varphi + 2k\pi| \\ \overline{\tau}(\mathbf{E_N}) &= C_{\text{rotation}} + C_{\text{translation}} + \overline{\tau}(\mathbf{E_M}), \end{aligned} \tag{4}$$

where T_s is the control sampling time interval, $\Delta v = a_{max} T_s$, $\Delta\omega = \alpha_{max} T_s$, and $\overline{\tau}$ is the upper bound of the function τ calculated only at exit points, with the integer constants $C_$ defining the maximal number of steps for rotation and translation between the exit points

$$C_{\text{rotation}} = \left\lceil \frac{\pi}{\Delta\omega T_s} \right\rceil, C_{\text{translation}} = \left\lceil \frac{\sqrt{2} d_c}{\Delta v T_s} \right\rceil. \tag{5}$$

This upper bound function $\overline{\tau}$ considers the worst-case transitions between cell exit points with the maximal orientation and translation differences. The upper bound at the goal position is

$$\overline{\tau}(\mathbf{E_G}) = C_{\text{rotation}}, \tag{6}$$

which accounts for the orientation of the robot at the goal position in the opposite direction from the desired goal orientation as the worst case.

The explicit form of the recursive equation of the upper bound function $\overline{\tau}$ in (4) can be derived as

$$\overline{\tau}(\mathbf{E_N}) = \frac{U(\mathbf{N})}{d_c} (C_{\text{rotation}} + C_{\text{translation}}) + C_{\text{rotation}}, \tag{7}$$

which can be verified by the mathematical induction from the goal position where $\frac{U(\mathbf{N})}{d_c}$ define the approximation of the number of cell transitions.

Figure 5 shows the 3D view of the time-step function τ with $e(\varphi)$ set to zero for clarity. We can notice discontinuities at the cell borders, but such discontinuities are local-minima free, i.e., there always exists incremental translation or rotation that can lower the cost τ.

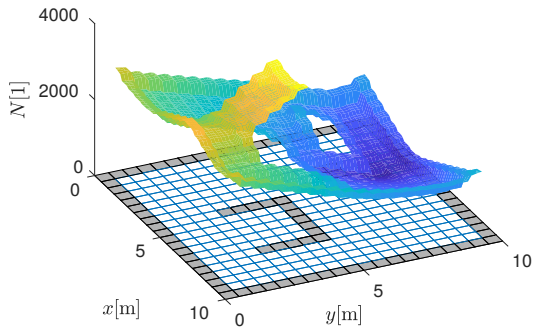


Fig. 5. 3D view of the conservative time-step navigation function $N = \tau$.

The final navigation function for the conservative approach is exactly the time-step function τ:

$$N(x, y, \varphi) = \tau(x, y, \varphi). \tag{8}$$

III. MODEL PREDICTIVE CONTROL

Model predictive control (MPC) is applied to find optimal controls $\mathbf{u}(t) = [v(t), \omega(t)]^T$ for a differential robot over a prediction horizon h that minimize objective function at current state $\mathbf{s}(t) = [x, y, \varphi]^T$

$$J(\mathbf{s}(t)) = \min_{\mathbf{u}(i-1)} \sum_{i=1}^{h} \left(N(\mathbf{s}(i)) + \mathbf{u}^T(i-1) \mathbf{R} \mathbf{u}(i-1) \right). \tag{9}$$

where v and ω are translational and angular velocity, i is a short notation for time $t + iT_s$ and T_s is sample time. Note that MPC considers the robot motion model and environment model. In horizon interval future robot state $\mathbf{s}(i)$ is predicted using differential drive kinematics

$$\begin{aligned} x(i+1) &= x(i) + v(i) T_s \cos\left(\varphi(i) + \tfrac{\omega(i) T_s}{2}\right) \\ y(i+1) &= y(i) + v(i) T_s \sin\left(\varphi(i) + \tfrac{\omega(i) T_s}{2}\right) \\ \varphi(i+1) &= \varphi(i) + \omega(i) T_s. \end{aligned} \tag{10}$$

Control actions are constrained by maximal velocities and accelerations by

$$\begin{aligned} 0 \le v(i) \le v_{\max} \quad &, \quad |\omega(i)| \le \omega_{\max} \\ \tfrac{|v(i) - v(i-1)|}{T_s} \le a_{\max} \quad &, \quad \tfrac{|\omega(i) - \omega(i-1)|}{T_s} \le \alpha_{\max}, \end{aligned} \tag{11}$$

where $v_{\max}$, $\omega_{\max}$, $a_{\max}$, $\alpha_{\max}$ are maximal allowed translational and rotational speeds and accelerations. In horizon navigation function must lower by fulfilling

$$N(\mathbf{s}(i)) \ge N(\mathbf{s}(h)),\ i = 1, \ldots, h. \tag{12}$$

During the horizon, the robot is supposed to drive on a circular arc where $v(i)/\omega(i)$ is its radius. The minimal horizon length is chosen so that robot driving at maximal speeds could safely decelerate to a stop at the end of the prediction horizon

$$h = \left\lceil \max\left(\frac{v_{\max}}{a_{\max} T_s}, \frac{w_{\max}}{\alpha_{\max} T_s} \right) \right\rceil + 1. \tag{13}$$

Controls $\mathbf{u}(i)$, $i \in 0, 1, \ldots, h-1$ in horizon therefore linearly decreases to zero as follows

$$\mathbf{u}(i) = \mathbf{u}(0) \left(1 - \frac{i}{h-1} \right). \tag{14}$$

Optimal control sequence $\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(h-1)$ that minimizes (9) defines the best feasible future trajectory and its first control action is applied to the robot in current time. In next time sample the procedure repeats.

To prove the convergence of the MPC scheme the value function J (employed as a Lyapunov function) needs to decrease each time sample [15].

A novel optimization strategy for solving (9) is proposed that combines optimization with a fixed candidate set of control action candidates and particle swarm optimization.

A. Fixed candidate optimization

To lower computational burden a fixed candidate optimization (FCO) is introduced in [15] to solve the MPC problem. At current velocities $\mathbf{u}_c = [v_c, \omega_c]^T$ (applied to the robot at time $t - T_s$) a course set of possible discrete accelerations $a_c \in \{-a_{max}, 0, a_{max}\}$, $\alpha_c \in \{-\alpha_{max}, 0, \alpha_{max}\}$ is defined to produce a set of 9 velocity candidates for optimization

$$\mathbf{u}(t) = \mathbf{u}_c + T_s[a_c, \alpha_c]^T \tag{15}$$

constrained by (11).

The main strengths of the proposed MPC with FCO are low computational complexity and generation of near-optimal trajectories with a guaranteed convergence as shown in [15]. While the obtained velocity profile includes higher noise due to the coarse set of possible accelerations used.

B. Particle swarm optimization

Particle swarm optimization (PSO) uses a stochastic strategy with a swarm of randomly perturbed particles to find a solution. Applying PSO to the MPC problem yields arbitrary velocities $\mathbf{u}(t)$ sampled from a continuum and constrained by (11). Each particle k is parametrized by a parameter vector $\mathbf{p}_k = [v_k, \omega_k]^T$ defining its velocities and increment vector $\Delta\mathbf{p}_k$ defining the change of the velocities. During MPC optimization the population of all particles is iteratively updated and validated according to objective function (9). Each particle keeps track of its parameters and remembers its best parameter $\mathbf{pB}_k$ achieved so far together with associated objective function $J_k = f(\mathbf{pB}_k)$. During optimization also the global best parameter vector of the entire swarm $\mathbf{gB}$ is stored.

In each sample time particles iteratively update by means of the following rules

$$\begin{aligned} \Delta\mathbf{p}_k &\leftarrow \gamma\Delta\mathbf{p}_k + c_1 \underset{(0,1)}{\mathbf{rand}}(\mathbf{pB}_k - \mathbf{p}_k) + c_2 \underset{(0,1)}{\mathbf{rand}}(\mathbf{gB} - \mathbf{p}_k) \\ \mathbf{p}_k &\leftarrow \mathbf{p}_k + \Delta\mathbf{p}_k, \end{aligned} \tag{16}$$

where $\gamma > 0$ is the inertia factor, $c_1 > 0$ is the self-cognitive constant, $c_2 > 0$ is the social constant and $\mathbf{rand}_{(0,1)}$ is a vector of uniformly distributed values in range $(0, 1)$. At the end of the optimization the best parameter is applied to the robot $\mathbf{u}(t) = \mathbf{gB}(t)$.

MPC with PSO produces smoother velocity profiles and can find better solutions because no velocity discretization is used. However the computational complexity gets much higher (than in MPC with FCO) due to several required iterations of more particles. Due to a random nature PSO the solution as well as search convergence are not guaranteed.

C. Combined deterministic-stochastic optimization

The main idea is to combine FCO and PSO in so-called combined deterministic-stochastic optimization (CDS) and inherit the advantages of both algorithms and thus produce trajectories with a smooth velocity profile, with guaranteed convergence, and low computational complexity.

CDS is a modified PSO algorithm (illustrated in Alg. 1) that runs in parallel $K_F = 9$ fixed particles and K_C changing particles. Fixed particles are initialized by (15) and they do not update during optimization. These fixed particles provide good starting parameters which can be used by other changing particles through $\mathbf{gB}$ when iteratively update by (16). In this way, CDS provides a better solution (more optimal and smoother) or at least as good as FCO itself. MPC with CDS is guaranteed to converge to the goal in a finite time from any non-occupied location in the environment where the goal is reachable ($N(x, y, \varphi) < \infty$). The algorithm is computationally efficient since the number of changing particles K_C in CDS can be much lower than in the equivalent PSO.

Algorithm 1 Combined deterministic-stochastic optimization.

Require: List of particles $k = 1, \cdots, K$ where first $k = 1, \ldots, K_F$ are fixed particles.
  for each particle $k = 1, \cdots, K_F$ **do**
    Initialize $\mathbf{p}_k$ by (15).
  end for
  for each particle $k = K_F + 1, \cdots, K$ **do**
    Randomly initialize $\mathbf{p}_k$, $\Delta\mathbf{p}_k = [0, 0]^T$, $\mathbf{pB}_k = \mathbf{p}_k$.
  end for
  $J_{\text{best}} = \infty$, *iter* $= 1$
  repeat
    for each particle $k = 1, \cdots, k$ **do**
      if $k > K_F$ ∣ *iter* $== 1$ **then**
        Compute objective J_k by (9) considering ramp down (14) and add penalty for (11) violat
        ion.
        Set *convergent* condition by (12).
        if $J_k < f(\mathbf{pB}_k)$ **then**
          $\mathbf{pB}_k = \mathbf{p}_k$
        end if
        if $f(\mathbf{pB}_k) < J_{\text{best}}$ & *convergent* **then**
          $\mathbf{gB} = \mathbf{pB}_k$, $J_{\text{best}} = f(\mathbf{gB})$
        end if
      end if
    end for
    for each particle $k = K_F + 1, \cdots, K$ **do**
      Update $\mathbf{p}_k$ and $\Delta\mathbf{p}_k$ by (16) constrained by (11).
    end for
  until *iter* $\leq$ *MAXiter*

IV. RESULTS AND COMPARISONS

The results of model predictive control with the proposed combined deterministic-stochastic optimization (MPC-CDS) are compared to the results of fixed candidate optimization (MPC-FCO) and particle swarm optimization (MPC-PSO). Additionally, MPC-based algorithms are compared to the kinodynamic Stable Sparse RRT planning approach (SST) [10]. Results are shown in Figs. 6 and 7 for the U-shaped map and a random obstacle map with two different starting poses. All the MPC trajectories are computed for the bicubic-interpolation and the conservative navigation function and collected also in Tab. I. Results are compared in terms of obtained trajectories, velocity profiles, length of the trajectory L, driving time t_{goal}, accumulated navigation A_N and normalized computational efficiency E_{comp} (according to the MPC-FCO).

TABLE I

ALGORITHMS VALIDATION ON THREE DIFFERENT MAPS

	Inter.	Alg.	L [m]	t_{goal} [s]	A_N [1]	E_{comp} [1]
U-shaped map	Tao	MPC-CDS	8.73	9.80	15581	1.64
		MPC-FCO	8.88	10.00	16393	1.00
		MPC-PSO	8.69	9.70	15340	11.03
	Bic	MPC-CDS	9.30	10.30	62.33	1.21
		MPC-FCO	9.29	10.30	63.93	1.00
		MPC-PSO	9.29	10.30	56.54	9.67
	n/a	SST	11.70	13.22	n/a	n/a
Random map 1	Tao	MPC-CDS	27.67	28.80	150772	2.18
		MPC-FCO	27.77	28.80	151872	1.00
		MPC-PSO	27.65	28.70	150444	15.85
	Bic	MPC-CDS	27.91	29.00	424.62	3.55
		MPC-FCO	28.31	29.40	435.58	1.00
		MPC-PSO	27.97	29.00	410.05	33.86
	n/a	SST	30.11	35.28	n/a	n/a
Random map 2	Tao	MPC-CDS	22.21	23.30	98487	2.14
		MPC-FCO	22.36	23.30	99643	1.00
		MPC-PSO	22.26	23.30	98036	18.31
	Bic	MPC-CDS	22.33	23.40	271.62	3.22
		MPC-FCO	22.32	24.10	274.30	1.00
		MPC-PSO	22.25	23.30	260.69	34.17
	n/a	SST	22.09	26.54	n/a	n/a

In simulations the following constraints are considered $v_{\max} = 1$ m/s, $\omega_{\max} = 6$ rad/s, $a_{\max} = 1$ m/s^2 and $\alpha_{\max} = 1$ m/s^2. Conservative navigation (Tao) outperforms bicubic interpolation by producing a bit shorter and faster trajectories and enables convergent control which is not guaranteed in the bicubic interpolation due to the mentioned surface anomalies near the obstacles. Regarding the optimization approach (CDS, FCO or PSO) used in the MPC the best (shorter, faster, and with smooth velocity profile) trajectories are obtained using PSO while the computational burden gets much higher. MPC-PSO uses 25 particles and 20 iterations while MPC-CDS uses 9 fixed and only 2 changing particles. CDS produces compromised results with smoother velocity profiles than FCO and much lower computational complexity than PSO. SST produces similar length of trajectories but with much slower velocity profiles due to the randomness of the velocity selection during the search process.

The experiments were performed on the Husky A200 mobile robot equipped with the Velodyne Lidar, and Xsens IMU sensors, in an outdoor environment populated with arbitrary-shaped unknown obstacles (cars and humans). To meet the real-time requirements of the control algorithm the MPC-FCO variant was deployed on the onboard Intel NUC computer running ROS. The localization of the robot is calculated with the EKF standard ROS package, which fuses the wheel encoders and the IMU sensor. The grid map of resolution 0.2 m was used for path (re)planning with the D* algorithm, which was filled continuously with the Lidar's points within a certain height range. These Lidar's points accumulated during the experiment scenario are presented in Fig. 8, where the moving obstacles (humans) can be distinguished from the static obstacles (cars, stairs) according to the reading's density. The velocity constraints for the Husky robot were $v_{\max} = 0.5$ m/s, $a_{\max} = 0.3$ m/s^2, $\omega_{\max} = 0.52$ rad/s, and $\alpha_{\max} = 0.87$ rad/s^2 with the backward motion allowed due to the 360° field of view. The experiment scenario consisted of three goals, named A, B, and C. Between goals A and B the person was blocking the robot's way to test the algorithm responsiveness to dynamic changes. The algorithm performance is presented in Fig. 9. The main computation burden goes to the path planning and replanning taking on average 30 ms for replanning and 60 ms for the initial search. On the other hand, the MPC optimization takes less than 10 ms for the calculation of the navigation function for each candidate trajectory. To check the convergent control performance the function J, employed as a Lyapunov function, needs to decrease each time step during the motion. This is fulfilled except if new obstacles appear and increase the path cost values and consequently the navigation function, as was the case with a moving obstacle between goals A and B. However, even after the increase of the values J due to the new obstacles, the values decreased to zero eventually. The control method showed fast and smooth motion with good performances in the presence of unknown and moving obstacles.

V. CONCLUSIONS

We presented two navigation functions deployed in a model predictive control with an efficient deterministic-stochastic optimization strategy. We showed how the conservative time-step navigation function, which is discontinuous but simple to compute outperforms the continuous navigation function which uses bicubic interpolation. The proposed combined optimization strategy enables convergent control performance, produces similar quality trajectories than computationally more demanding particle swarm optimization and with smother velocity profiles than the fixed candidate set optimization at only a bit higher complexity. Proposed algorithm has no limitation on obstacles shape and can handle dynamic environments if navigation function is computed from D* discrete graph search or similar.

ACKNOWLEDGMENT

This work was supported by the Slovenian Research Agency under Grant P2-0219 and the Croatian Ministry of Science and Education through the European Regional Development Fund (DATACROSS) under Grant KK.01.1.1.01.0009.

REFERENCES

[1] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/

[2] M. Brezak and I. Petrović, "Real-time approximation of clothoids with bounded error for path planning applications," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 507–515, April 2014.

[3] B. Sencer, K. Ishizaki, and E. Shamoto, "A curvature optimal sharp corner smoothing algorithm for high-speed feed motion generation of NC systems along linear tool paths," *International Journal of Advanced Manufacturing Technology*, vol. 76, no. 9, pp. 1977–1992, 2015.

[4] A. Zdešar and I. Škrjanc, "Optimum Velocity Profile of Multiple Bernstein-BéZier Curves Subject to Constraints for Mobile Robots," *ACM Trans. Intell. Syst. Technol.*, vol. 9, no. 5, Jun. 2018. [Online]. Available: https://doi.org/10.1145/3183891

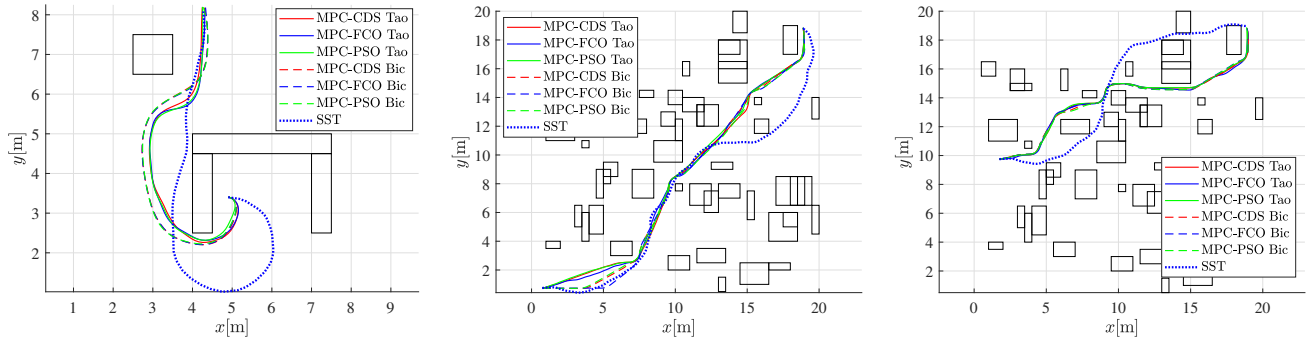


Fig. 6. Trajectory comparison for the U-shaped map (left), Random-obstacle map with different start location (middle and left) using the conservative navigation (Tao), the bicubic interpolated navigation (Bic), and the SST algorithm.

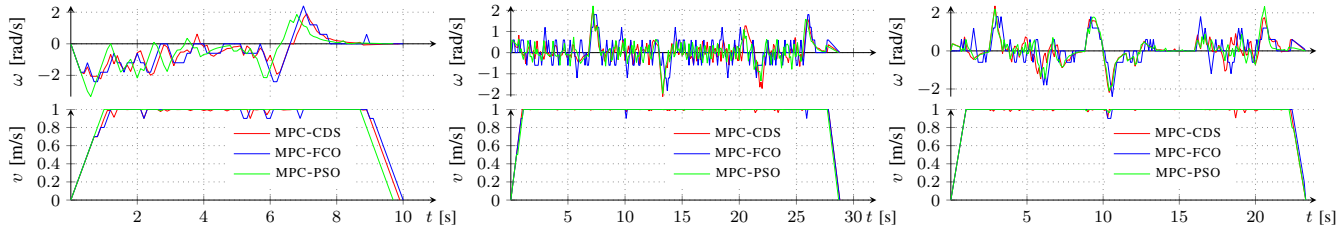


Fig. 7. Velocity profile comparison for the U-shaped map (left), Random-obstacle map with different start location (middle and left) using the conservative navigation.

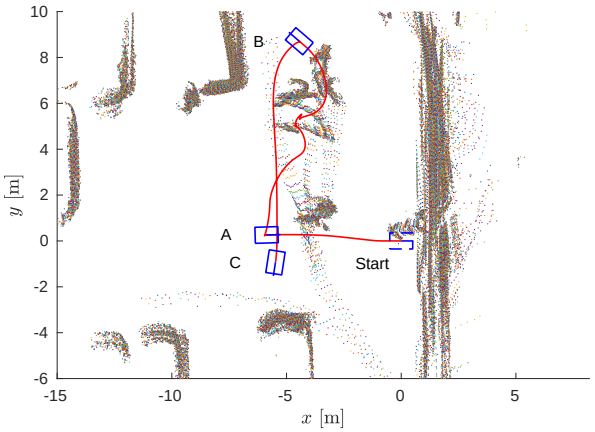


Fig. 8. Robot's trajectories with accumulated laser readings in an outdoor environment.

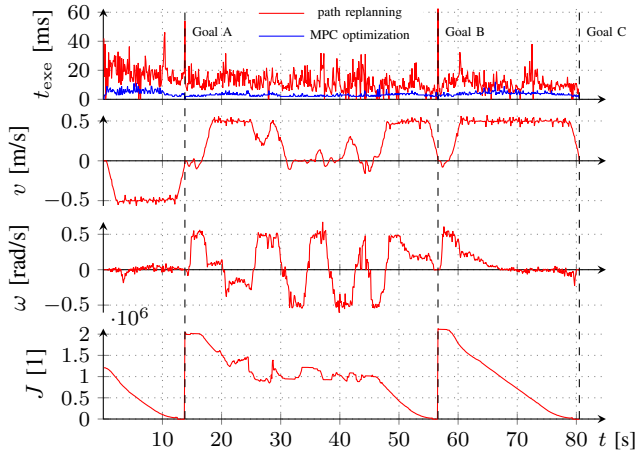


Fig. 9. Algorithm performance, velocity profile, and Lyapunov function in outdoor navigation between goals A, B, and C.

[5] G. Klančar and S. Blažič, "Optimal Constant Acceleration Motion Primitives," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 8502–8511, 2019.

[6] 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.

[7] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[8] M. Đakulović, M. Čikeš, and I. Petrović, "Efficient interpolated path planning of mobile robots based on occupancy grid maps," in *10th IFAC Symposioum on Robotic Control (SYROCO2012)*, 2012, pp. 349–354.

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

[10] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[11] S. Yoon, D. Lee, J. Jung, and D. H. Shim, "Spline-based RRT* using piecewise continuous collision-checking algorithm for car-like vehicles," *Journal of Intelligent & Robotic Systems*, pp. 1–13, 2017.

[12] G. Klančar, M. Seder, S. Blažič, I. Škrjanc, and I. Petrović, "Drivable Path Planning Using Hybrid Search Algorithm Based on E* and Bernstein-Bézier Motion Primitives," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 8, pp. 4868–4882, 2021.

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

[14] D. Kiss and G. Tevesz, "A receding horizon control approach to obstacle avoidance," in *2011 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, 2011, pp. 397–402.

[15] M. Seder, M. Baotić, and I. Petrović, "Receding horizon control for convergent navigation of a differential drive mobile robot," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 653–660, March 2017.