Lifting-based nonlinear model predictive control for high-performance mobile robot tracking

Matsuzaki, Fumiya
Joint Graduate School of Mathematics for Innovation, Kyushu University

Sakaguchi, Akinori
Faculty of Information Science and Electrical Engineering, Kyushu University

Yamamoto, Kaoru
Faculty of Information Science and Electrical Engineering, Kyushu University

https://hdl.handle.net/2324/7442664

出版情報：pp.751-756, 2026-09-14. IEEE
バージョン：

Lifting-based nonlinear model predictive control for high-performance mobile robot tracking

Fumiya Matsuzaki[1], Akinori Sakaguchi[2], and Kaoru Yamamoto[2]

Abstract— **Nonlinear model predictive control (NMPC) is widely used in robotics due to its ability to handle nonlinear dynamics and explicitly enforce state and input constraints by solving an optimal control problem online. However, standard NMPC implementations typically rely on direct time discretization and evaluate costs and constraints only at sampling instants. This paper investigates the real-world effectiveness of a previously introduced lifting-based NMPC formulation that explicitly accounts for intersample responses by minimizing a continuous-time tracking error criterion. The formulation is implemented for trajectory tracking of a differential-drive robot and evaluated against a conventional NMPC approach. Real-world experiments on lemniscate and right-triangular reference trajectories, under variations in target speed and sampling period, show that the lifting-based NMPC consistently achieves lower tracking RMSE and mitigates corner-cutting at vertices. Solve-time measurements further confirm that the approach can be executed online on embedded hardware while maintaining real-time feasibility.**

I. Introduction

Nonlinear model predictive control (NMPC) has become a state-of-the-art technique in robotics because it can handle complex nonlinear dynamics while explicitly enforcing state and input constraints. By solving an optimal control problem at each sampling instant, NMPC provides a framework for high-precision autonomous behavior. These advantages have been demonstrated across a wide range of applications, including vehicle steering control, trajectory tracking, and multi-robot coordination tasks such as flocking and cooperative landing [3], [4], [7], [8], [10], [13]. In most robotic applications, continuous-time plant dynamics are controlled by on-board or networked digital computers that process sensor measurements at discrete intervals. Thus, such systems are inherently sampled-data control systems. Despite this property, standard NMPC implementations often rely on direct time discretizations of the plant dynamics (e.g., Euler's method or the fourth-order Runge–Kutta method).

Conventional sampled-data control design can be broadly classified into two categories: (i) designing a continuous-time controller and subsequently discretizing it, or (ii) discretizing the continuous-time plant first and then designing a discrete-time controller [5]. However, the former does not explicitly account for the sampling period during controller design, whereas the latter typically guarantees performance and stability only at the sampling instant. As a result, intersample ripple, such as undesired intersample oscillations, may arise. Particularly in the context of NMPC, this implies that the satisfaction of state and input constraints is not guaranteed between sampling instants, even if they are satisfied at the sample points.

To address issues with intersample behavior, a lifting technique was introduced as a modern sampled-data control framework [2], [16]. This technique provides a representation that maps continuous-time signals into elements of a function space, allowing intersample responses to be rigorously taken into account. In particular, linear lifting has been developed in conjunction with classical control methods such as linear quadratic regulation and H^2/H^∞ control [1], [14]. A framework for nonlinear lifting has been advocated in recent years [18], and a control method combining it with NMPC has been proposed, utilizing a numerical approximation of the lifted dynamics via the fast-sample fast-hold (FSFH) approximation combined with numerical integration [6]. However, existing studies are limited to simulation studies, and experimental verification on real systems has not yet been demonstrated.

Motivated by the above gap, we investigate the proposed lifting-based NMPC in real-world trajectory-tracking experiments and summarize our contributions as follows:

- We provide the first real-world implementation and experimental validation of lifting-based NMPC on a physical robotic platform.
- We demonstrate that the lifting-based formulation achieves consistently accurate trajectory tracking across a range of operating conditions within the tested ranges, including variations in the sampling period and target speed, and yields consistently lower tracking error than conventional NMPC.
- Although lifting-based NMPC leads to a more involved numerical procedure, we show that the resulting optimal control problem can still be solved online while maintaining real-time feasibility on embedded hardware, as evidenced by solve-time measurements.

The remainder of this paper is organized as follows. Section II provides the mathematical notation used throughout the paper, the fundamental concept of the lifting technique, and the continuous-time unicycle model. In Section III, we describe the hierarchical control architecture through the formulation of the lifting-based NMPC and its numer-

This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP24K07546.

[1]Fumiya Matsuzaki is with the Joint Graduate School of Mathematics for Innovation, Kyushu University, 819-0395, Fukuoka, Japan matsuzaki.fumiya.592@s.kyushu-u.ac.jp

[2]Akinori Sakaguchi and Kaoru Yamamoto are with the Faculty of Information Science and Electrical Engineering, Kyushu University, 819-0395, Fukuoka, Japan sakaguchi@ees.kyushu-u.ac.jp; yamamoto@ees.kyushu-u.ac.jp

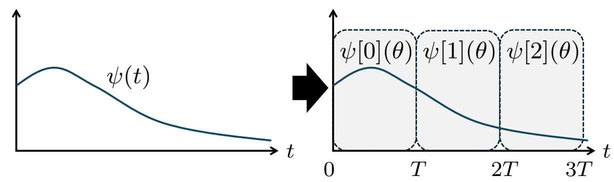


Fig. 1. The graphical image of the lifting technique.

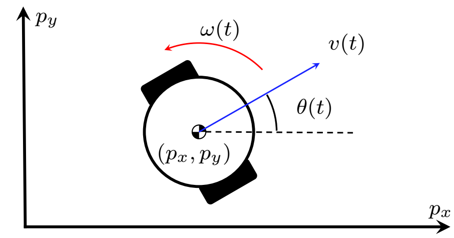


Fig. 2. Differential-drive wheeled robot model.

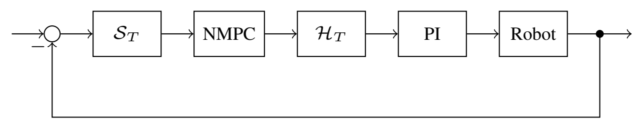


Fig. 3. The structure of control system.

ical implementation using the FSFH approximation scheme. Section IV demonstrates the effectiveness of the proposed approach through the real-environment experiments, and compares the performance with the conventional NMPC. Finally, we conclude the paper and discuss future research directions in Section V.

II. Preliminaries

A. Notation

Throughout this paper, $\mathbb{R}$, $\mathbb{N}$, and $\mathbb{Z}_{\geq 0}$ denote the sets of real numbers, natural numbers, and non-negative integers, respectively. Vectors are represented by boldface symbols. $\mathbf{0}_N \in \mathbb{R}^N$ denotes the vector whose elements are all zeros. The transpose of a matrix or vector M is denoted by $M^\top$. For a vector $\boldsymbol{a} \in \mathbb{R}^N$, $\mathrm{diag}(\boldsymbol{a}) \in \mathbb{R}^{N\times N}$ denotes the diagonal matrix with elements of $\boldsymbol{a}$ on its main diagonal. The notation $M \succeq 0$ (resp. $M \succ 0$) indicates that the matrix M is positive semidefinite (resp. positive definite). For a vector $\boldsymbol{a} \in \mathbb{R}^N$ and a symmetric positive definite matrix $Q \in \mathbb{R}^{N\times N}$, $\|\boldsymbol{a}\|_Q^2 = \boldsymbol{a}^\top Q \boldsymbol{a}$ denotes the weighted squared norm.

B. Lifting – Sampled-Data Control Systems Design

This section reviews the lifting technique, a fundamental theory in modern sampled-data control systems design. Lifting is a method that maps continuous-time signals into an equivalent discrete-time representation, allowing one to rigorously account for intersample behavior [16]. For a given sampling period $T > 0$, the lifting operator $\mathcal{L}$ is defined as:

$$\mathcal{L} : \psi \mapsto \{\psi[k](\cdot)\}_{k=0}^{\infty}, \quad \psi[k](\theta) \triangleq \psi(kT + \theta),$$

where $\theta \in [0, T)$, and ψ belongs to an appropriate function space on $[0, \infty)$, such as L^2 or L^2_{loc}. In other words, the lifting operator converts a continuous-time signal into a discrete-time sequence whose samples are functions defined over the interval $[0, T)$, as illustrated in Fig. 1. This transformation yields a discrete-time, time-invariant system on an infinite-dimensional space, through which the intersample behavior is treated rigorously.

The lifting technique can be applied to nonlinear systems [18]. Consider a continuous-time nonlinear plant expressed as:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \\ \boldsymbol{y}(t) &= h(\boldsymbol{x}(t)), \\ \boldsymbol{x}_0 &= \boldsymbol{x}(0), \end{aligned} \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$, $\boldsymbol{u}(t) \in \mathbb{R}^m$, and $\boldsymbol{y}(t) \in \mathbb{R}^p$ are the state, input, and output vectors at time t, respectively. With the lifting technique, the *lifted system* of system (1) is defined with the lifted input, output, and state as:

$$\begin{aligned} \boldsymbol{x}[k+1](\theta) &= \Phi((k+1)T + \theta, \boldsymbol{x}[k](T), \boldsymbol{u}[k+1](\cdot)), \\ \boldsymbol{y}[k](\theta) &= h(\boldsymbol{x}[k](\theta)), \end{aligned} \tag{2}$$

for $\theta \in [0, T)$, where Φ is the solution of the differential equation (1). Here $\boldsymbol{u}[k+1](\cdot)$ means that $\boldsymbol{x}[k+1](\theta)$ depends on $\boldsymbol{u}[k+1](\tau)$, where $0 \leq \tau < \theta$.

C. Robot Model

We consider a differential-drive wheeled robot as the control target. The robot dynamics are given by

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) \triangleq \begin{bmatrix} v(t)\cos\theta(t) \\ v(t)\sin\theta(t) \\ \omega(t) \end{bmatrix}, \tag{3}$$

where $\boldsymbol{x}(t) \triangleq [p_x(t), p_y(t), \theta(t)]^\top \in \mathbb{R}^3$ denotes the state vector with the 2D position (p_x, p_y) and the heading angle θ in the inertial frame at time $t \geq 0$. The control input $\boldsymbol{u}(t) \triangleq [v(t), \omega(t)]^\top \in \mathbb{R}^2$ represents the control input vector with the linear velocity v and the angular velocity ω, both defined in the body-fixed frame. In this paper, we assume that the robot moves on a horizontal plane without slipping.

III. Methodology

A. Overall Control Structure

In this paper, we employ a hierarchical control architecture consisting of an upper-level NMPC and a lower-level PI tracking controller, as illustrated in Fig. 3. The NMPC operates with sampling period T and, at each state update through a sampler $\mathcal{S}_T$, solves an optimization problem to compute an optimal discrete-time control sequence $\boldsymbol{\nu}^*[k]$ for $k = 0, \ldots, N-1$, where $N \in \mathbb{N}$ denotes the number of prediction steps. This discrete-time command sequence is mapped to a lifted control input signal $\boldsymbol{u}[k](\theta)$ via an appropriate hold function H, i.e., $\boldsymbol{u}[k](\theta) = H(\boldsymbol{\nu}[k])$ under a hold operator $\mathcal{H}_T$. The resulting signal is used as the reference input for the lower-level PI controller, which drives the wheel motors at a higher update rate.

B. Lifting-Based NMPC

In this section, we describe the lifting-based NMPC, which enhances conventional NMPC by incorporating the nonlinear lifting technique [6]. At each sampling instant $t = sT, s \in \mathbb{Z}_{\geq 0}$, we solve the following finite-horizon optimal control problem:

$$\begin{aligned}
&\underset{\substack{\boldsymbol{\nu}[k]\in\mathbb{R}^m\\ k=0,\dots,N-1}}{\text{minimize}} \quad \sum_{k=0}^{N-1}\int_0^T \ell(\tilde{\boldsymbol{x}}[k](\theta),\tilde{\boldsymbol{u}}[k](\theta))d\theta + \varphi(\tilde{\boldsymbol{x}}[N](0))\\
&\text{s.t.}\\
&\boldsymbol{x}[0](\theta) = \Phi(\theta, \boldsymbol{x}(sT), \boldsymbol{u}[0](\cdot)),\\
&\left.\begin{aligned}
&\boldsymbol{x}[k+1](\theta) = \Phi((k+1)T+\theta,\\
&\qquad\qquad \boldsymbol{x}[k](T), \boldsymbol{u}[k+1](\cdot)),\\
&\boldsymbol{u}_{\min} \leq \boldsymbol{u}[k](\theta) \leq \boldsymbol{u}_{\max},
\end{aligned}\right\} \quad k = 0,\dots,N-1,
\end{aligned} \tag{4}$$

where $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is the stage cost, and $\varphi : \mathbb{R}^n \to \mathbb{R}$ is the terminal cost. $\tilde{\boldsymbol{x}}[k](\theta)$ and $\tilde{\boldsymbol{u}}[k](\theta)$ denote the deviations from the reference trajectories and reference inputs within the k-th prediction step, namely

$$\begin{aligned}
\tilde{\boldsymbol{x}}[k](\theta) &\triangleq \boldsymbol{x}[k](\theta) - \boldsymbol{x}_{\mathrm{ref}}[k](\theta),\\
\tilde{\boldsymbol{u}}[k](\theta) &\triangleq \boldsymbol{u}[k](\theta) - \boldsymbol{u}_{\mathrm{ref}}[k](\theta).
\end{aligned}$$

C. Numerical Implementation via FSFH Approximation

Since an analytical solution Φ to the nonlinear differential equations (3) is generally unavailable [9], we employ the FSFH approximation [15], [17] to construct a predictive model of (3). We first introduce the subdivision factor $N' \in \mathbb{N}$ of the sampling interval $[0, T)$ and divide it into N' subintervals $[0, T/N'), [T/N', 2T/N'), \dots, [(N'-1)T/N', T)$. Under this discretization, the state transition over each subdivision is expressed as:

$$\begin{aligned}
&\boldsymbol{x}[k]((i+1)\Delta) - \boldsymbol{x}[k](i\Delta)\\
&= \int_{i\Delta}^{(i+1)\Delta} f(\boldsymbol{x}[k](\tau), \boldsymbol{u}[k](\tau))\mathrm{d}\tau
\end{aligned} \tag{5}$$

for $i = 0, \dots, N'-1$, where $\Delta = T/N'$. To achieve numerical accuracy, this paper employs the fourth-order Runge–Kutta (RK4) method to compute the approximate solution Φ of (3) and Simpson's rule for numerical integration of the cost function within the FSFH framework.

IV. Experiments

This section demonstrates the effectiveness of the lifting-based NMPC through real-world experiments. As a benchmark, we compare the proposed method with a conventional NMPC that neglects intersample behavior. The conventional NMPC solves the following discrete-time optimal control problem at each sampling instant $t = sT, s \in \mathbb{Z}_{\geq 0}$:

$$\begin{aligned}
&\underset{\substack{\boldsymbol{u}[k]\in\mathbb{R}^m\\ k=0,\dots,N-1}}{\text{minimize}} \quad \sum_{k=0}^{N-1} \ell(\tilde{\boldsymbol{x}}[k],\tilde{\boldsymbol{u}}[k]) + \varphi(\tilde{\boldsymbol{x}}[N])\\
&\text{s.t.}\\
&\quad \boldsymbol{x}[0] = \boldsymbol{x}(sT),\\
&\quad \left.\begin{aligned}
&\boldsymbol{x}[k+1] = f_{\mathrm{RK4}}(\boldsymbol{x}[k], \boldsymbol{u}[k]),\\
&\boldsymbol{u}_{\min} \leq \boldsymbol{u}[k] \leq \boldsymbol{u}_{\max},
\end{aligned}\right\} k = 0,\dots,N-1,
\end{aligned} \tag{6}$$

where f_{RK4} denotes the discrete-time dynamics obtained by applying the RK4 method to (3) with step size T. $\tilde{\boldsymbol{x}}[k]$ and $\tilde{\boldsymbol{u}}[k]$ denote the deviations from the reference state and input at the k-th prediction step, defined as $\tilde{\boldsymbol{x}}[k] \triangleq \boldsymbol{x}[k] - \boldsymbol{x}_{\mathrm{ref}}[k]$ and $\tilde{\boldsymbol{u}}[k] \triangleq \boldsymbol{u}[k] - \boldsymbol{u}_{\mathrm{ref}}[k]$, respectively. Unlike the lifting-based approach, the cost function in (6) is evaluated only at the sampling instants.

For both methods, we employ quadratic stage and terminal costs given by

$$\ell(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) = \|\tilde{\boldsymbol{x}}\|_Q^2 + \|\tilde{\boldsymbol{u}}\|_R^2, \quad \varphi(\tilde{\boldsymbol{x}}) = \|\tilde{\boldsymbol{x}}\|_{Q_N}^2, \tag{7}$$

where $Q \succeq 0, R \succ 0$, and $Q_N \succeq 0$ are weighting matrices. While the lifting-based method minimizes the continuous-time integral of the tracking error, the conventional NMPC penalizes the error only at the sampling instants.

A. Experimental Setup

1) Hardware and Software Setup: In our real-world experiments, the control architecture was validated using a custom-built differential-drive wheeled robot. The system was integrated using ROS 2, which manages communication between the high-level NMPC controller and the low-level tracking controller, as well as between the motion-capture system and the robot control stack. Both NMPC methods were executed on a Raspberry Pi 4B, which solves the optimal control problems (4) and (6) online at each sampling instant. For real-time optimization, we used OpEn [11], which provides a high-performance solver based on the Proximal Averaged Newton-type method for Optimal Control (PANOC) [12]. The computed optimal control inputs from the NMPC were forwarded as a reference input signal to a low-level PI controller. This low-level controller was implemented on a Pixracer Pro flight controller running PX4 firmware, which managed the actual motor drives. Communication between the ROS 2 stack and PX4 was established via a Micro XRCE-DDS protocol, where the agent and the client ran on the Raspberry Pi and the Pixracer Pro, respectively. This setup enabled the seamless transmission of NMPC reference commands to the low-level controller. To ensure precise command following, the low-level controller operated at 100 Hz and incorporated both feedback and feedforward components to drive the wheel motors accurately according to the NMPC outputs. The overall experimental architecture is illustrated in Fig. 4.

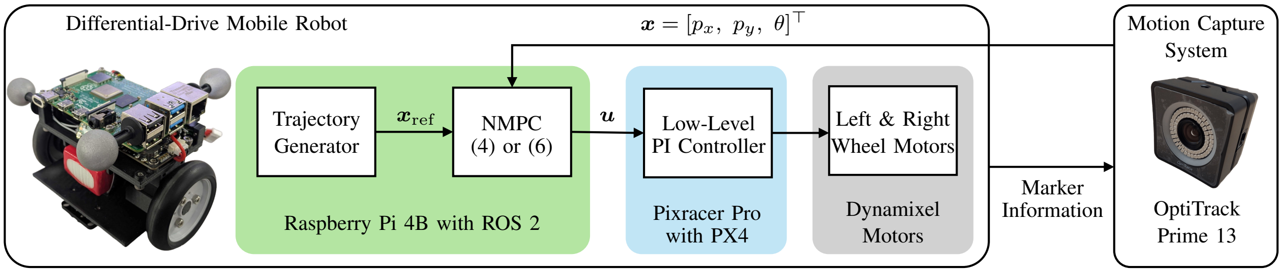


Fig. 4. System architecture of the real-world experiment.

TABLE I
COMMON CONFIGURATION PARAMETERS.

Parameter	Symbol	Value
Sampling period	T	$\{0.10,\ 0.25\}$
Prediction horizon	τ_p	1.0
Subdivision factor	N'	5
Min. control input	$\boldsymbol{u}_{\min}$	$[0.0,\ -8.0]^\top$
Max. control input	$\boldsymbol{u}_{\max}$	$[0.6,\ 8.0]^\top$

TABLE II
METHOD-SPECIFIC WEIGHTS.

Conventional NMPC		Lifting-Based NMPC	
Q	$\operatorname{diag}(60,\ 60,\ 30)$	Q	$\operatorname{diag}(200,\ 200,\ 15)$
R	$\operatorname{diag}(1,\ 1)$	R	$\operatorname{diag}(1,\ 0.8)$
Q_N	$\operatorname{diag}(60,\ 60,\ 30)$	Q_N	$\operatorname{diag}(200,\ 200,\ 15)$

2) *Reference Trajectory Design:* To evaluate both NMPC methods, trajectory-tracking experiments were conducted using reference trajectories generated at two target speeds, $v_{\text{ref}} = 0.3\ \text{m/s}$ and $0.5\ \text{m/s}$, and with two sampling periods, $T = 0.1$ s and $T = 0.25$ s. Although a nonzero input reference is not explicitly included in the NMPC cost function (i.e., $\boldsymbol{u}_{\text{ref}} = \boldsymbol{0}_2$), varying the progression speed of the time-parameterized trajectory enables evaluation of closed-loop responsiveness as well as the associated computational load.

3) *NMPC Setup and Horizon Definition:* In the following, we summarize the controller settings and the experimental parameters used for each method. Note that the prediction horizon was fixed to a constant duration $\tau_p \triangleq NT$, and the number of discretization steps N was adjusted according to the sampling period T. The resulting parameters are reported in Table I.

4) *Weight Tuning Protocol:* Since the two NMPC formulations (4) and (6) exhibit different numerical scaling and sensitivity, both the state and input weight matrices were tuned separately for each method. The resulting parameters are shown in Table II. Tuning was performed using the lemniscate trajectory at a target speed of $v_{\text{ref}} = 0.3$ m/s with a sampling period of $T = 0.1$ s. After tuning, the same weight matrices were kept fixed and used for all subsequent experiments, including those with different trajectory shapes and operating conditions.

TABLE III
COMPARISON OF TRACKING RMSE FOR BOTH METHODS ACROSS DIFFERENT TRAJECTORIES.

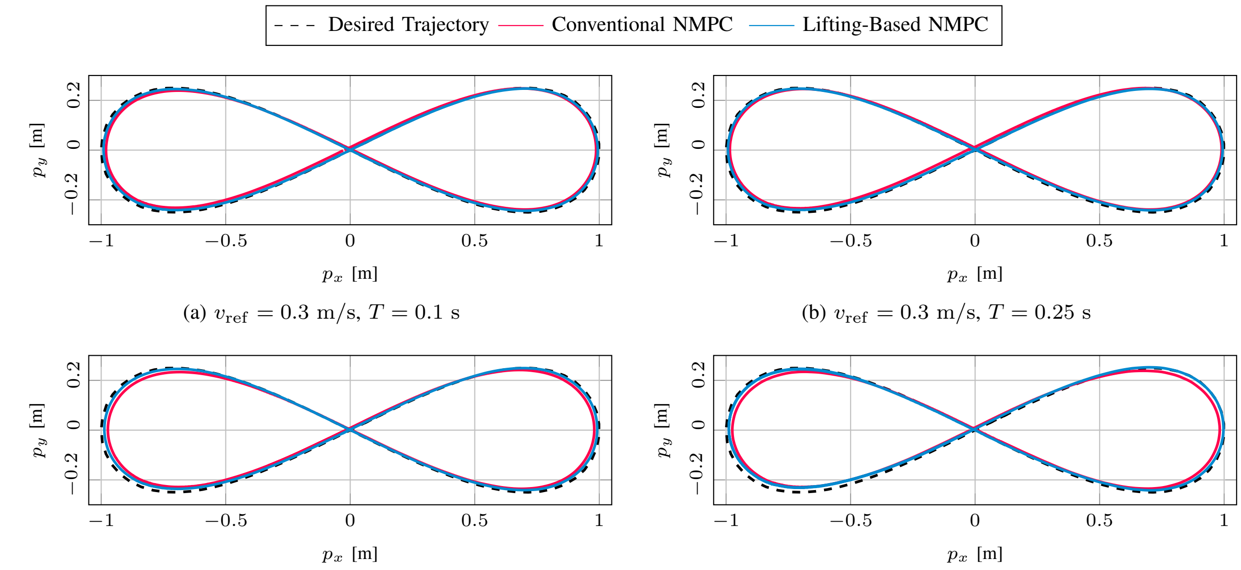
Trajectory	Condition (v_{ref} [m/s], T [s])	RMSE [m] Conventional	RMSE [m] Lifting-Based
Lemniscate	(0.3, 0.10)	0.0119	0.0054
	(0.3, 0.25)	0.0112	0.0056
	(0.5, 0.10)	0.0152	0.0074
	(0.5, 0.25)	0.0156	0.0099
Right-Triangular	(0.3, 0.10)	0.0122	0.0097
	(0.3, 0.25)	0.0149	0.0122
	(0.5, 0.10)	0.0203	0.0162
	(0.5, 0.25)	0.0283	0.0225

B. Hardware Results

1) *Lemniscate Trajectory Tracking:* Fig. 5 illustrates the trajectory-tracking results for the lemniscate reference trajectory under different sampling periods and target speeds. Consistent with the figure and the RMSE results presented in Table III, the lifting-based NMPC achieves lower tracking error than the conventional NMPC in tracking accuracy across all tested conditions. In particular, at the target speed of $0.5\ \text{m/s}$, the conventional NMPC exhibits noticeable deviation from the reference trajectory, cutting inward, whereas the lifting-based NMPC maintains high-accuracy tracking of the desired path.

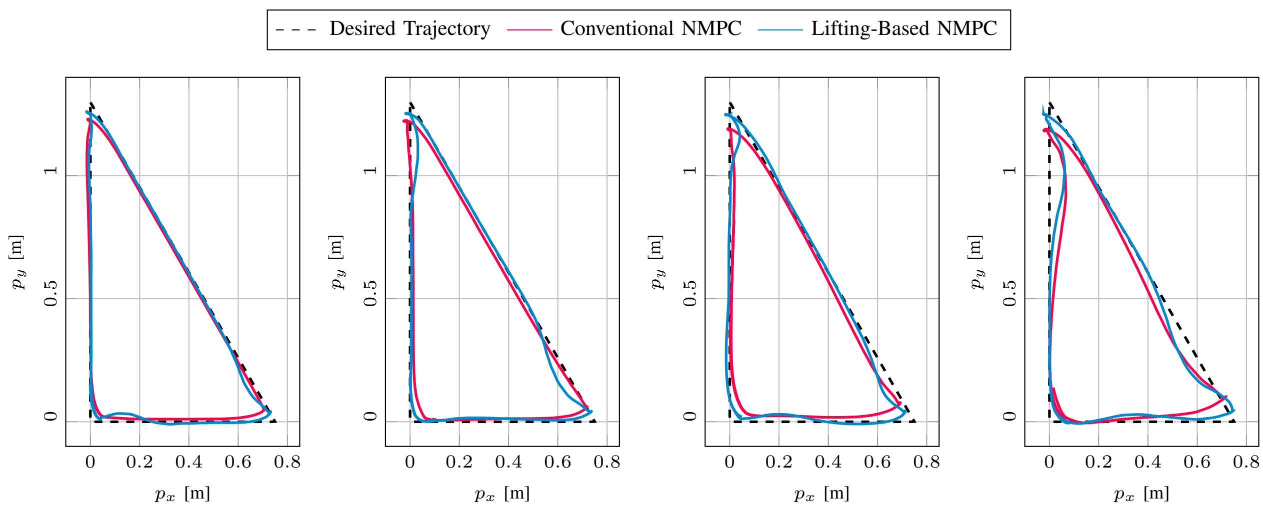
While the tracking accuracy varies with the target speed, the impact of changing the sampling period T appears to be limited. This robustness to larger T is likely related to the geometric characteristics of the lemniscate trajectory, which transitions smoothly between straight and curved segments, allowing the predictive model to approximate the trajectory sufficiently well even with larger sampling intervals. However, even under these favorable conditions, reducing T does not eliminate the inward deviation observed in the conventional NMPC. Overall, these results suggest that the lifting-based NMPC enables high-speed and high-accuracy tracking without requiring excessively short sampling intervals, which is advantageous in computationally constrained settings.

2) *Right-Triangular Trajectory Tracking:* Fig. 6 presents the trajectory-tracking results for the right-triangular reference trajectory under different sampling periods and target speeds, analogous to the lemniscate case. Consistent with Ta-



(a) $v_{\text{ref}} = 0.3$ m/s, $T = 0.1$ s
(b) $v_{\text{ref}} = 0.3$ m/s, $T = 0.25$ s
(c) $v_{\text{ref}} = 0.5$ m/s, $T = 0.1$ s
(d) $v_{\text{ref}} = 0.5$ m/s, $T = 0.25$ s

Fig. 5. Comparison of lemniscate trajectory tracking between conventional NMPC and lifting-based NMPC.



(a) $v_{\text{ref}} = 0.3$ m/s, $T = 0.1$ s (b) $v_{\text{ref}} = 0.3$ m/s, $T = 0.25$ s (c) $v_{\text{ref}} = 0.5$ m/s, $T = 0.1$ s (d) $v_{\text{ref}} = 0.5$ m/s, $T = 0.25$ s

Fig. 6. Comparison of right-triangular trajectory tracking between conventional NMPC and lifting-based NMPC.

ble III, the lifting-based NMPC achieves lower tracking error than the conventional NMPC across all tested conditions. In particular, around the vertex regions, the conventional NMPC exhibits pronounced corner-cutting behavior, deviating significantly from the reference trajectory. This corner-cutting tendency becomes more pronounced at higher target speeds. In contrast, the lifting-based NMPC accurately follows the sharp turns at the vertices, demonstrating its capability to handle abrupt changes in trajectory direction.

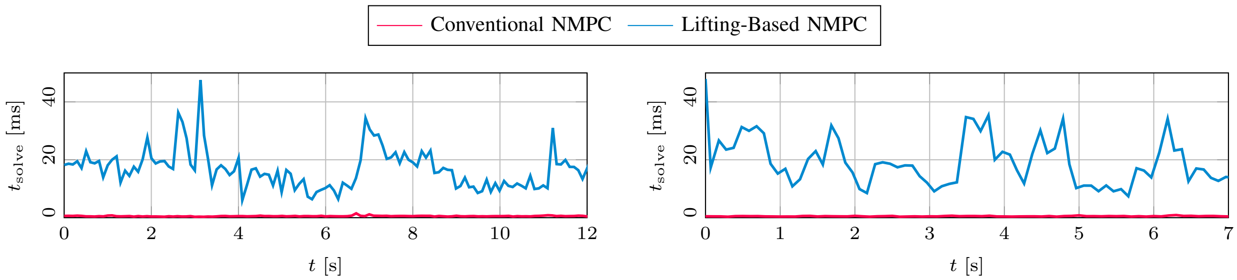
Furthermore, the lifting-based NMPC shows a slight increase in tracking error along the straight segment immediately after a turn, which may be due to tracking the reference trajectory too closely near the vertex. Nevertheless, since the error contribution induced by corner-cutting at the vertices is dominant, the lifting-based NMPC consistently achieves a lower overall RMSE than the conventional method.

C. Computational Results

We analyze solve-time statistics for the most computationally demanding setting, namely the right-triangular trajectory with $T = 0.1$ s and $v_{\text{ref}} \in \{0.3, 0.5\}$ m/s, since its sharp corners yield the most challenging optimization instances.

Fig. 7 shows the time series of solve time per iteration. As can be seen, the observed solve time t_{solve} stays below T throughout the experiment for both methods, i.e., no deadline misses ($t_{\text{solve}} > T$) were observed. We imposed a solver time limit (set to T) to guarantee termination within each sampling period.

In the conventional NMPC, the optimization was solved on the order of milliseconds throughout, with a maximum



(a) $v_{\rm ref} = 0.3$ m/s

(b) $v_{\rm ref} = 0.5$ m/s

Fig. 7. Comparison of solve time between conventional NMPC and lifting-based NMPC with sampling period $T = 0.1$ s using right-triangular trajectory.

observed solve time of $\tilde{t}_{\max} \approx 1.5$ ms. In contrast, the solve time for lifting-based NMPC is longer due to the additional internal subdivision required by the FSFH-based approximation, as well as numerical integration. Nevertheless, the maximum time remained $\tilde{t}_{\max} \approx 48$ ms, which is within the sampling period T. These results suggest that, although the lifting-based NMPC is computationally more demanding than the conventional approach, it remains suitable for online, real-time implementation.

V. Conclusion

In this work, we extended the lifting-based NMPC framework to the trajectory-tracking control of a differential-drive wheeled robot. The lifting-based method was implemented on actual robot hardware, and its performance was validated through trajectory-tracking experiments. The experimental results demonstrated that the lifting-based NMPC yielded consistently lower tracking error than the conventional NMPC for both lemniscate and triangular trajectories under various operating conditions. Notably, the lifting-based NMPC effectively mitigated the corner-cutting phenomenon observed with the conventional NMPC during sharp turns, leading to improvements in tracking performance. Furthermore, the lifting-based NMPC was shown to be computationally feasible for real-time implementation on embedded hardware. Future work includes theoretical analysis of the recursive feasibility and stability of the lifting-based NMPC, application of the proposed method to robotic systems with stronger nonlinearities (e.g., quadrotors), and extension of the approach to large-scale systems such as formation control and swarm robotics.

References

[1] B. Bamieh and J. Pearson, "A general framework for linear periodic systems with applications to H^∞ sampled-data control," *IEEE Transactions on Automatic Control*, vol. 37, no. 4, pp. 418–435, 1992.

[2] B. Bamieh, J. Pearson, B. A. Francis, and A. Tannenbaum, "A lifting technique for linear periodic systems with applications to sampled-data control," *Systems & Control Letters*, vol. 17, no. 2, pp. 79–88, 1991.

[3] V. Cataffo, G. Silano, L. Iannelli, V. Puig, and L. Glielmo, "A nonlinear model predictive control strategy for autonomous racing of scale vehicles," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2022, pp. 100–105.

[4] J. Chen, L. Xu, H. Ebel, and P. Eberhard, "An online optimization-based trajectory planning approach for cooperative landing tasks," in *38th IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2025, pp. 12 108–12 114.

[5] T. Chen and B. A. Francis, *Optimal Sampled-Data Control Systems*. Springer London, 1995.

[6] N. Gerdpratoom, F. Matsuzaki, Y. Yamamoto, and K. Yamamoto, "Enhanced sampled-data model predictive control via nonlinear lifting," *International Journal of Robust and Nonlinear Control*, vol. 35, no. 18, pp. 7621–7632, 2025.

[7] N. Gerdpratoom and K. Yamamoto, "Decentralized nonlinear model predictive control-based flock navigation with real-time obstacle avoidance in unknown obstructed environments," *Frontiers in Robotics and AI*, vol. 12, 2025.

[8] A. Nag, S. Huang, A. Themelis, and K. Yamamoto, "Flock navigation with dynamic hierarchy and subjective weights using nonlinear MPC," in *IEEE Conference on Control Technology and Applications*, 2022, pp. 1135–1140.

[9] D. Nesić and A. R. Teel, "Sampled-data control of nonlinear systems: An overview of recent results," in *Perspectives in robust control*, S. R. Moheimani, Ed. London: Springer London, 2001, pp. 221–239.

[10] J. Schlagenhauf, P. Hofmeier, T. Bronnenmeyer, R. Paelinck, and M. Diehl, "Cascaded nonlinear MPC for realtime quadrotor position tracking," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 7026–7032, 2020.

[11] P. Sopasakis, E. Fresk, and P. Patrinos, "OpEn: Code generation for embedded nonconvex optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6548–6554, 2020.

[12] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *56th IEEE Annual Conference on Decision and Control*, 2017, pp. 1939–1944.

[13] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear MPC and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, pp. 3357–3373, 2022.

[14] H. Toivonen, "Sampled-data control of continuous-time systems with an H_∞ optimality criterion," *Automatica*, vol. 28, no. 1, pp. 45–54, 1992.

[15] K. Yamamoto, M. Nagahara, and Y. Yamamoto, "Signal reconstruction with generalized sampling," in *56th IEEE Annual Conference on Decision and Control*, 2017, pp. 6253–6258.

[16] Y. Yamamoto, "A function space approach to sampled data control systems and tracking problems," *IEEE Transactions on Automatic Control*, vol. 39, no. 4, pp. 703–713, 1994.

[17] Y. Yamamoto, A. G. Madievski, and B. D. Anderson, "Approximation of frequency response for sampled-data control systems," *Automatica*, vol. 35, no. 4, pp. 729–734, 1999.

[18] Y. Yamamoto and K. Yamamoto, "Nonlinear sampled-data systems—a lifting framework," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 6406–6410, 2023.