Tracking of Unicycle Robots Using Event-Based MPC With Adaptive Prediction Horizon

Zhongqi Sun, Yuanqing Xia, Senior Member, IEEE, Li Dai, and Pascual Campoy

Abstract—in this article, we propose two event-based model predictive control (MPC) schemes with adaptive prediction horizon for tracking of unicycle robots with additive disturbances. The schemes are able to reduce the computational burden from two aspects: reducing the frequency of solving the optimization control problem (OCP) to relieve the computational load and decreasing the prediction horizon to decline the computational complexity. Event-triggering and self-triggering mechanisms are developed to activate the OCP solver aperiodically, and a prediction horizon update strategy is presented to decrease the dimension of the OCP in each step. The proposed schemes are tested on a networked platform to show their efficiency.

Index Terms—Adaptive prediction horizon, event-triggered control, model predictive control (MPC), self-triggered control, unicycle robots.

I. INTRODUCTION

Unicycle robots are a class of typical nonholonomic systems, and tracking control is a fundamental problem for cooperative control, formation control, and path planning. So far, many approaches have been developed for this problem, see for example [1]–[3]. However, these techniques ignore the mechanical constraint such as input saturation and nonholonomic constraint. Model predictive control (MPC), also known as receding horizon control, is an efficient approach for explicitly handling constraints and optimizing the performance [4]–[6]. It requires to solve a complex optimization control problem (OCP) in each step to provide a prediction control sequence, and only the first portion of this sequence is implemented to the plant.

Therefore, heavy computational load and low utilization of the results are the two shortcomings of MPC, which may prevent its application to fast control systems. This article intends to develop implementable MPC schemes to realize the tracking control of unicycle robots while overcoming the two disadvantages of traditional MPC.

To solve the computational problem in MPC, we first recall the development of networked control systems (NCSs), in which the controllers and the sensors are no longer implemented on dedicated platforms, but using shared communication networks [7]. Limited communication resources and constrained energy pose new challenges in control design. To reduce the number of control updates, a recent approach is to sample only when needed. This is the basic idea of event-based control [8]. This motivates us to use the idea of event-based control in MPC to reduce the frequency of solving the OCP: to compute only when needed.

Event-triggering and self-triggering mechanisms are two main approaches in event-based control. Their difference lies in that the event-triggered control relies on continuous or periodic states measurement, while self-triggered control computes the next triggering time at the previous one based on the estimation of the future states. Therefore, self-triggered control has the advantage of requiring less states information, while at the same time, is susceptible to uncertainties when compared to the event-triggered control scheme [9]. Event-triggered and self-triggered MPC are reported in [9]–[18] aiming at reducing the number of solving the OCP or handling the problems in NCSs.

The event-triggered and self-triggered MPC schemes discussed previously are able to relieve computational load to some extent, but are still not suitable for application. Because they only reduce the frequency of solving the OCP, and the computational complexity in each triggering instant remains high. In this article, we will develop event-triggered and self-triggered MPC schemes with adaptive prediction horizon strategy to relieve the heavy computation burden for tracking of unicycle robots. The schemes will reduce the computational burden from two perspectives. On one hand, the controller is updated in an event-driven fashion: to update only when a specific condition is violated. In such a way, it can reduce the average frequency of solving the OCP. On the other hand, the prediction horizon decreases adaptively as the triggering error approaches a terminal region. This may reduce the dimension of the OCP; thereby, reducing the computational complexity at each sampling time. In summary, the main contributions of this article include: 1) Two event-based MPC with adaptive horizon schemes are developed based on event- and self-triggering frameworks. 2) Input-to-state
stability for both approaches proposed is established. 3) The proposed schemes are tested on an experimental platform. The experimental results show that the schemes are efficient to reduce the computational load. From the results, we further discuss the advantages and disadvantages of both schemes through comparison.

The remainder of this article is organized as follows. We formulate the control problem and develop a computable cost in Section II. Two event-based MPC schemes with adaptive prediction horizon are developed in Section III. In Section IV, we test the proposed schemes in a networked platform to verify the efficiency. Finally, Section V concludes this article.

**Notation:** \( \mathbb{R} \) and \( \mathbb{N} \) are the real and nonnegative integers, respectively. For some \( r_1 \in \mathbb{R}, n_2 \in \mathbb{N} \), and \( n_2 > n_1 \), \( \mathbb{R}_{\geq n_1}, \mathbb{N}_{\geq n_2}, \) and \( \mathbb{N}_{[n_1, n_2]} \) denote the sets \( \{ r \in \mathbb{R} | r > r_1 \}, \{ n \in \mathbb{N} | n \geq n_1 \}, \) \( \{ n \in \mathbb{N} | n \leq n_2 \} \), and \( \{ n \in \mathbb{N} | n_1 \leq n \leq n_2 \} \), respectively. For a symmetrical matrix \( P, P \geq 0 \) means that \( P \) is positive definite. \( X(P) \) and \( \lambda(P) \) denote the maximum and minimum eigenvalues of matrix \( P \). For a vector \( x, \| x \|_P = \sqrt{x^T P x} \), and \( \| x \|_p = \sqrt{x^T P x} \) with \( P > 0 \) represent the Euclidean norm and \( P \)-norm, respectively. \( \sup_{x \in A} \{ \} \) and \( \inf_{x \in A} \{ \} \) represent the upper and lower bound of the elements in \( A \), respectively.

## II. Problem Formulation

### A. Tracking System Description

Consider a leader–follower tracking problem based on virtual structure. The leader is described by the following kinematics:

\[
\dot{\xi}_r(t) = f_r(\xi_r(t), u_r(t)) = \begin{bmatrix} \cos \theta_r(t) & 0 \\ \sin \theta_r(t) & 0 \end{bmatrix} u_r(t)
\]  

where \( \xi_r(t) = [p_{r_1}(t), \theta_r(t)]^T \) is the state with position \( p_{r_1}(t) = x_r(t), \) orientation \( \theta_r(t), \) and \( u_r(t) = [v_r(t), \omega_r(t)]^T \) is the control input with linear velocity \( v_r(t) \) and angular velocity \( \omega_r(t). \)

Considering the nonholonomic constraint, we control the head position of the follower. Its kinematics is given by

\[
\dot{\xi}_f(t) = f_f(\xi_f(t), u_f(t)) = \begin{bmatrix} \cos \theta_f(t) & -\rho \sin \theta_f(t) \\ \sin \theta_f(t) & \rho \cos \theta_f(t) \end{bmatrix} u_f(t)
\]  

where \( \xi_f(t) = [p_{f_1}(t), \theta_f(t)]^T, \) \( p_{f_1}(t) = [x_f(t), y_f(t)]^T, \) \( u_f(t) = [v_f(t), \omega_f(t)]^T, \) and \( \rho \) is the wheel base, as shown in Fig. 1(a). The control input is assumed to be constrained by \( |v_f(t)| \leq a \) and \( |\omega_f(t)| \leq b, \) where \( a > 0 \) and \( b > 0 \) are known constants.

The leader–follower setup is shown in Fig. 1(b), in which \( O \) is the global frame, \( O \) and \( I \) are the Frenet–Serret frames fixed on the leader and the follower, respectively. In this virtual structure framework, the tracking position is defined as \( p_d = [x_d, y_d]^T \) with respect to frame \( O. \) Define the tracking error as \( p_e = [x_e, y_e]^T \) with respect to frame \( I, \) which is given by

\[
p_e(t) = R(-\theta_f(t))(p_r(t) - p_f(t)) + R(\theta_e(t))p_d
\]  

where \( \theta_e(t) = \theta_r(t) - \theta_f(t), \) and \( R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \) is the rotation matrix. Taking the derivative of (3), its dynamics is then

\[
\dot{p}_e(t) = f_d(p_e(t), u_e(t)) = \begin{bmatrix} 0 & \omega_f \\ -\omega_f & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \end{bmatrix} + u_e(t)
\]  

where

\[
u_e(t) = \begin{bmatrix} -v_f + (v_r - x_d \omega_r) \cos \theta_r - x_d \omega_r \sin \theta_r \\ -\rho \omega_f + (v_r - x_d \omega_r) \cos \theta_r + y_d \omega_r \cos \theta_r \end{bmatrix}
\]

### B. Discretization and Computational Modeling

Minimizing a cost for continuous system usually requires a quite long optimization time. A discretized system has advantages in computational perspective. Consequently, we discretize (2) using forward Euler approach as

\[
\xi_f(k+1) = \xi_f(k) + \delta f_h(\xi_f(k), u_f(k)) + d(k)
\]  

where \( d(k) \) is external disturbance. Similarly

\[
p_e(k+1) = f_d(p_e(k), u_f(k))
\]  

\[
= \begin{bmatrix} 1 & \delta \omega_f(k) \\ 0 & 1 \end{bmatrix} p_e(k) + \delta u_e(k)
\]

where \( \delta \) is the sampling period. Without loss of generality, we assume that the tracking error is bounded by \( \| p_e(k) \| \leq \ell. \) The cost to be optimized online is then formulated as

\[
J_d(p_e(k), u_e(k), N) = \sum_{i=0}^{N-1} L(p_e(k+i|k), u_e(k+i|k)) + g(p_e(k+N|k))
\]  

where stage cost \( L(p_e(k+i|k), u_e(k+i|k)) = \| p_e(k+i|k) \|_2^2 + \| u_e(k+i|k) \|_2^2 \), and the terminal cost \( g(p_e(k+N|k)) = \| p_e(k+N|k) \|_2^2. \) To guarantee stability of the system, the weighting matrices \( Q, P, \) and \( R \) are designed to satisfy the following assumption.

**Assumption 1:** There exists a robust terminal set \( \Omega_e \) and a corresponding controller \( \kappa(p_e), \) such that, \( \forall p_e(k) \in \Omega_e, \)

\[
g(p_e(k+1)) - g(p_e(k)) \leq -L(p_e(k), \kappa(x(k))).
\]  

Before proceeding, we introduce two lemmas for convenience of later analysis.
**Lemma 1:** The nonlinear function \( f_h(\xi, u) \) is locally Lipschitz continuous in \( \xi \) with Lipschitz constant \( L_P = \sqrt{2(a^2 + \rho^2 b^2)} \).

**Lemma 2 (See [19]):** Let \( S = \{ N_{m_0} \times N_{n_0} \} \). Suppose \( \mu : S \rightarrow \mathbb{R}_{> 0} \), \( b : S \rightarrow \mathbb{R}_{> 0} \), and \( \phi : \mathbb{R}_{> 0} \rightarrow \mathbb{R}_{> 0} \) is a continuous nondecreasing function. For \( c \in \mathbb{R}_{\geq 0} \), \( \alpha \in \mathbb{R}_{> 0} \) and \((m, n) \in S\), if
\[
\mu^\alpha(m, n) \leq c + \sum_{s = m_0}^{m-1} \sum_{t = n_0}^{n-1} b(s, t)\phi(\mu(s, t))
\]
then
\[
\mu(m, n) \leq \{ \Psi_{\alpha}^{-1}(\Psi_{\alpha}(c) + B(m, n)) \}^\pi
\]
for all \( m \leq m_1 \) and \( n \leq n_1 \), where
\[
\Psi_{\alpha}(\gamma) = \int_1^\gamma \frac{d\phi(s^{1/\alpha})}{\phi(\gamma^{1/\alpha})}, \quad B(m, n) = \sum_{s = m_0}^{m-1} \sum_{t = n_0}^{n-1} b(s, t)
\]
\( \Psi_{\alpha}^{-1} \) is the inverse of \( \Psi_{\alpha} \), and \((m_1, n_1) \in S\) is chosen such that \( \Psi_{\alpha}(c) + B(m, n) \) is in the domain of \( \Psi_{\alpha}^{-1} \) for all \( m \leq m_1 \) and \( n \leq n_1 \).

## III. Event-Based MPC with Adaptive Horizon

In this section, we will develop an event-triggered MPC and a self-triggered MPC to stabilize the system. The prediction horizons of both schemes are decreased adaptively as the tracking error converging to the origin. At the same time, the sampling period is aperiodic and the controller is updated only when some specific conditions are violated.

### A. Optimization Problem

In a traditional MPC framework, the OCP is solved in every step and only the first control action is used to control the system, which implies the waste of the computational resources. Moreover, the prediction horizon is an invariable constant. This means that the OCP has the same dimension in each step even if the system states are close to the terminal region or the origin. Therefore, we utilize event-triggering mechanism in MPC to reduce the rate of solving the OCP. Meanwhile, we shrink the prediction horizon to reduce the dimension of the OCP and the computational complexity. To this aim, define \( \{ k_j \}, j \in \mathbb{N}_{\geq 0} \), as the triggering time sequence, in which the successive update times have the following relationship:
\[
k_{j+1} = k_j + m_{k_j}, \quad k_0 = 0
\]
where \( m_{k_j} \in \mathbb{N}_{\geq 1} \) is the interexecution time. In event-triggered control, \( m_{k_j} \) is determined based on the current states measurement, while in self-triggered control, it is calculated at the previous triggering instant.

In addition, define \( N_{k_j} \in \mathbb{N}_{\geq 1} \) as the prediction horizon at the triggering instant \( k_j \), which is adaptively updated as follows:
\[
N_{k_{j+1}} = N_{k_j} - n_{k_j}, \quad N_0 = N_p
\]
where \( n_{k_j} \in \mathbb{N}_{\geq 0} \) is the shrinking length of the prediction horizon. Similarly, \( n_{k_j} \) is updated at the current triggering instant in event-triggered control and is determined at the previous triggering time in self-triggered control.

Based on the framework introduced previously, the OCP to be solved online is then described as follows.

**OCP 1:** At triggering instant \( k_j \), find the optimal control sequence \( u_f^j(k_j) = \{ u_f^j(k_j | k_j), u_f^j(k_j + 1 | k_j), \ldots, u_f^j(k_j + N_{k_j} - 1 | k_j) \} \) as well as the corresponding trajectory \( \xi_f^j(k_j) = \{ \xi_f^j(k_j + 1 | k_j), \xi_f^j(k_j + 2 | k_j), \ldots, \xi_f^j(k_j + N_{k_j} | k_j) \} \) by solving the following problem:
\[
\min_{u_f(k_j)} J_d(p_e(k_j), u_f(k_j), N_{k_j})
\]
\[
\text{s.t.} \quad p_e(k_j | k_j) = p_e(k_j)
\]
\[
u_f(k_j + i | k_j) \in U
\]
\[
p_e(k_j + i + 1 | k_j) = f_d(p_e(k_j + i | k_j), u_f(k_j + i | k_j))
\]
\[
p_e(k_j + N_{k_j} | k_j) \in \Omega_e
\]
where \( i \in \mathbb{N}_{0, N_{k_j} - 1} \), \( \Omega_e = \{ \hat{p}_e : \| \hat{p}_e \|_R^2 \leq \varepsilon^2 \} \) with \( \varepsilon > 0 \) is the terminal region.

**Remark 1:** The constraints in OCP 1 are standard formulations in MPC. At each triggering time, the solver is initialized by the measurement of the actual state, i.e., constraint (15), which provides the feedback. Based on the initial state, the future behavior is predicted subject to the input constraint (16) and the system constraint (17). To guarantee stability, the state at the end of the horizon should be guaranteed to enter a terminal set, which is imposed in (18). The design of this terminal set can be found in [20].

**Remark 2:** For a nominal system, recursive feasibility of the OCP is easy to establish. However, for a disturbed system, a large disturbance may cause the feasibility lost. Therefore, we assume that the disturbance \( d(k) \) in (5) is upper bounded by \( \| d(k) \| \leq \eta \) with \( \eta \in \mathbb{R}_{> 0} \), and this bound is derived to guarantee recursive feasibility of the OCP 1. The detailed analysis about this upper bound is similar to the idea employed in [14], which is omitted here for the sake of simplicity.

### B. Event-Triggering Mechanism

The event-triggering condition is developed based on the measurement of current state and its optimal prediction obtained at the previous triggering time. When the norm of their error violates a predefined threshold, it stands to reason that a new control sequence should be calculated. Before proceeding, we first study the error between the actual trajectory and the optimal one using an \( m \) step open-loop control stating from the same point, i.e.,
\[
\| \xi_f(k_j + m) - \xi_f^*(k_j + m|k_j) \|
\]
\[
= \left\| \xi_f(k_j) + \sum_{i=0}^{m-1} \delta f_h(\xi_f(k_j + i), u_f^*(k_j + i)) + d(k_j + i) \right\| - \xi_f^*(k_j) - \sum_{i=0}^{m-1} \delta f_h(\xi_f(k_j + i), u_f^*(k_j + i)) \right\|
\]
(19)
Using the condition $\xi_f(k_j) = \xi_f^j(k_j)$ and Lipschitz condition yields
\[
\|\xi_f(k_j + m) - \xi_f^j(k_j + m|k_j)\| - \lambda_{1}\delta\sum_{i=0}^{m-1}\|\xi_f(k_j + i) - \xi_f^j(k_j + i)\| + m\eta.
\] (20)

Applying the Gronwall–Bellman–Ou–Iang-type inequality given by Lemma 2 corresponds to
\[
\|\xi_f(k_j + m) - \xi_f^j(k_j + m|k_j)\| \leq m\eta e^{\lambda_{1}\delta(m-1)}.
\] (21)

Based on the abovementioned analysis, we design the threshold of the triggering condition as $\sigma\eta e^{\lambda_{1}\delta(\sigma-1)}$, where $\sigma \in \mathbb{N}_{\geq 1}$ is a tuning parameter representing the minimum interexecution time. The triggering condition is then designed as follows:
\[
\|\xi_f(k_j + m_{k_j}) - \xi_f^j(k_j + m_{k_j}|k_j)\| \geq \sigma\eta e^{\lambda_{1}\delta(\sigma-1)}
\] (22)
and the actual interexecution time is given by
\[
m_{k_j} = \sup_m\{\|\xi_f(k_j + m) - \xi_f^j(k_j + m|k_j)\| < \sigma\eta e^{\lambda_{1}\delta(\sigma-1)}\}.
\] (23)

Moreover, we let the system be automatically triggered at the initial time $k_0$ and at $k_j + N_{k_j}$, because no control action is available in both of the two cases. As a result, the interexecution time is upper and lower bounded by $\sigma \leq m_{k_j} \leq N_{k_j}$.

**Remark 3.** Increasing the minimal interexecution time $\sigma$ can reduce the triggering frequency, however, at the sacrifice of the tracking performance, and thus how to choose $\sigma$ should be a tradeoff.

A long prediction can ensure the stability, but render the OCP complex. Shrinking the prediction horizon may have the advantage of reducing the dimension of the OCP. In standard MPC, the prediction horizon is a constant to ensure the tracking error at the end of the prediction horizon enters the terminal region. However, as the tracking error approaching to the terminal region, a shorter length of prediction horizon may be sufficient to guarantee the satisfaction of the terminal constraint. This is the basic idea of the shrinking strategy of the prediction horizon designed in the following. At each triggering time instant, the size of previous prediction horizon can provide evidence for designing the shrinking strategy. The shortest horizon length from the previous prediction is given by
\[
\hat{N}_{k_j} = \inf\{i : p^c_{-i}(k_j + i|k_j) \in \Omega_{e}, i \in [0, N_{k_j-1}]\}.
\] (24)

However, over shortening the prediction horizon may result in infeasible of the OCP and, thus, destabilize the system. Therefore, we have to consider stability condition of the closed-loop system. To ensure stability of the closed-loop system, it should be required that $k_{j+1} + N_{k_{j+1}} > k_j + N_{k_j}$. As a result, an upper bound of the shrinking length is given by
\[
n_{k_j} \leq m_{k_j} - 1.
\] (25)

To summarize, the prediction horizon is updated in terms of (13) and is decreased by
\[
n_{k_j} = \min\{m_{k_j} - 1, N_{k_j} - \hat{N}_{k_j}\}.
\] (26)

**Algorithm 1:** Event-Triggered MPC With Adaptive Prediction Horizon Scheme.

1: Solve the OCP (1) to find out the optimal control sequence $u^*_f(k_j)$ and the corresponding trajectory $\xi^*_f(k_j)$.
2: Initialize the time index $i = 0$;
3: if $(\|\xi_f(k_j + i) - \xi_f^j(k_j + i|k_j)\| < \sigma\eta e^{\lambda_{1}\delta(\sigma-1)}$ or $i < N_{k_j})$
   then
4: Apply the control action $u^*_f(k_j + i|k_j)$ to the follower;
5: Increase the time index $i = i + 1$, measure the actual state $\xi_f(k_j + i)$, and goto step 3;
6: else
7: Determine the inter-execution time $m_{k_j} = i$;
8: Find out $\hat{N}_{k_j}$ such that $\xi_f^j(k_j + \hat{N}_{k_j}|k_j) \in \Omega_e$ and $\xi_f^j(k_j + \hat{N}_{k_j} - 1|k_j) \notin \Omega_e$;
9: Determine the prediction horizon $N_{k_{j+1}}$ in accordance with (26);
10: Update the triggering time $k_{j+1} \rightarrow k_j$ and goto step 1.
11: end if

The development of event-triggering condition and the prediction horizon update strategy are codesigned rather than a simple combination. From (26), one notes that the decreased length of the horizon $n_{k_j}$ is related to the interexecution time $m_{k_j}$ as well as the prediction in the previous triggering instant. The condition (24) presents a shrunk horizon $\hat{N}_{k_j}$ that ensures feasibility for the OCP at current triggering time. While (25) limits the shrinking size of the horizon for establishing closed-loop stability of the system. Fig. 2 demonstrates the relationships between $m_{k_j}, N_{k_j}, \hat{N}_{k_j}$, and $N_{k_{j+1}}$. The prediction horizon at $k_{j+1}$ should satisfy
\[
k_j + N_{k_j} < k_{j+1} + N_{k_{j+1}} \leq k_{j+1} + N_{k_j}
\] (27)
i.e., the end of the prediction should fall into the interval with blue shadow in Fig. 2. At the same time, the tracking error is guaranteed to enter the terminal region at $k_{j+1} + N_{k_{j+1}}$.

The event-triggered MPC with adaptive prediction horizon scheme is then summarized in Algorithm 1.

The following theorem states the main results of event-triggered MPC with adaptive prediction horizon scheme.
**Theorem 1**: Suppose that the follower is controlled by the optimizing control \( u^*_f(k_j + i|k_j) \) at time \( k_j + i \), the controller update time is determined by (22), and the prediction horizon is decreased by (26). Then, system (6) is input-to-state stable.

The proof of Theorem 1 is provided in Appendix B.

**C. Self-Triggering Mechanism**

The event-triggering mechanism requires measuring the states and calculating the tracking error in each step. We present a self-triggering mechanism to wake up the OCP solver in this section. The next triggering time is determined at the previous one. Meanwhile, a suboptimal converging performance is guaranteed. For this goal, take the optimal cost as a Lyapunov function and study the deviation of the Lyapunov function between two sequential sampling times. For a nominal system, i.e., neglecting the disturbances in (5), the periodic-time-triggered MPC with constant prediction horizon \( N \) can guarantee the following convergence performance:

\[
L(p_e(k|k), u_e(k|k)) \leq J_d(p^*_e(k), u^*_e(k), N) - J_d(p^*_e(k+1), u^*_e(k+1), N). \tag{28}
\]

The disturbance, self-triggering mechanism and the variable prediction horizon may degrade the performance. Therefore, the self-triggering mechanism and the prediction horizon update strategy are designed to guarantee a suboptimal performance of the form

\[
b_k \sum_{i=0}^{m-1} L(p_e(k_j + i|k_j), u_e(k_j + i|k_j)) \leq J_d(p^*_e(k_j), u^*_e(k_j), N_{k_j}) - J_d(p^*_e(k_{j+1}), u^*_e(k_{j+1}), N_{k_{j+1}}) \tag{29}
\]

where \( 0 < \beta < 1 \) represents the performance loss level.

To guarantee performance (29), the interexecution time is chosen as

\[
m_{k_j} = \sup_m \left\{ m : h(m, N_{k_j}, \eta) \leq (1 - \beta) \times \sum_{i=0}^{m-1} L(p_e(k_j + i|k_j), u_f(k_j + i|k_j)) \right\} \tag{30}
\]

where \( h(m, N_{k_j}, \eta) = \sum_{i=0}^{N_{k_j}-1} [\eta^2 \tilde{q}^2 e^{2L_P(i-1)} + 2m\eta \tilde{q}^2 e^{L_P(i-1)}(r + \varepsilon)] + m \in \mathbb{N}_{[1,N_{k_j}]} \). The next triggering time is then determined by

\[
k_{j+1} = k_j + m_{k_j}. \tag{31}
\]

For the prediction horizon update, the strategy is the same as that in the event-triggered case. Thus, the prediction horizon is also updated by (13). The only difference from the event-triggered case is that the next prediction horizon is determined at the previous triggering time. The self-triggered MPC scheme is then summarized in Algorithm 2.

**Remark 4**: A balance between the triggering frequency and the optimal performance loss can be achieved by tuning \( \beta \). A relatively large \( \beta \) can reduce the frequency of solving the OCP 1 and slow down the horizon update rate. Whereas a relatively small \( \beta \) can ensure a better closed-loop performance with a higher rate but slower changing of the horizon update.

For the self-triggered case, the main result is concluded as follows:

**Theorem 2**: Suppose that the follower is controlled by the optimizing control \( u^*_f(k_j + i|k_j) \) at time \( k_j + i \), the controller update time is determined by (31), and the prediction horizon is updated the same as that in the event-triggered case. Then, system (6) is input-to-state stable, and the tracking error is guaranteed to enter \( \Omega_e \) in finite time.

The proof of Theorem 2 can be found in Appendix C.

**IV. EXPERIMENTS**

In this section, we show the effectiveness of the proposed event-triggered MPC with adaptive prediction horizon algorithms using an Epuck robot [21], which is a typical unicycle modeled robot. Its wheel speed is bounded by 0.13 m/s and wheel base is given by \( \rho = 0.0267 \) m.

**A. Experimental System and Control Architecture**

The experimental platform is shown in Fig. 3, which consists of a motion capture system, a wireless communication system and a couple of Epuck robots. The motion capture system is equipped with eight capture cameras to detect the infrared reflection balls fixed on the robots, and further to provide the position and orientation of the robots. The robots can communicate with each other as well as with a PC through a wireless Wi-Fi network.

The control architecture is shown in Fig. 4. The proposed MPC schemes run in a PC equipped with a dual-core 3.20 GHz
Intel i5 CPU, 7.88 GB RAM, and the 64-b Windows 10 operating system. The optimization problem is solved using interior point approach with solver interior point optimizer (see [22]). The motion capture system transmits the state information of the robots to the PC over the wireless network. The controller in the PC computes the control prediction according to the feedback information and sends the current control signal to the robots. The Epuck robot is equipped with a Wi-Fi module to receive the current control signal. The information is analyzed by an ARM 7 processor and executed by a dsPIC controller.

B. Experiment Results

In the experiment, we set the reference trajectory as a circular path with linear velocity $v_r = 0.015$ m/s and angular velocity $\omega_r = 0.04$ rad/s. We set the upper bound of the linear and angular speed of the follower to be $a = 0.065$ m/s and $b = 2.8$ rad/s, respectively. Its desired separation with the leader is set to be $p_d = [-0.1, -0.1]^{T}$ with respect to frame $^rO$. We set the sampling period as $\delta = 0.2$ s and the initial prediction horizon as $N_0 = 50$. The positive definite matrices in the cost function are chosen as $P = \text{diag}(0.4, 0.4)$ and $Q = \text{diag}(0.2, 0.2)$, respectively.

We first test the event-triggered MPC scheme with adaptive horizon on the platform. In the design procedure, we assume that the external disturbances are bounded by $\eta = 0.01$ and set the minimal interexecution steps as $\sigma = 2$. Fig. 5 shows the tracking trajectory. Fig. 6 shows the actual control inputs and the tracking errors. It can be seen that the tracking errors converge to a neighborhood of the origin.

Fig. 7 shows the errors between the actual trajectory and the optimal one, which are used to activate the OCP solver. Meanwhile, we also depict the triggering time and prediction horizon at each triggering instant. It can be observed that the frequency of solving the OCP is aperiodic and is reduced compared to traditional periodic time-triggered MPC. The solver is waked up 279 times by event-triggered MPC while it requires 600 times solving the OCP in traditional MPC. In addition, the prediction horizon decreases as the states of the follower approaching its desired trajectory, which implies that the dimension of the OCP in each step is reducing. This may reduce the computational complexity of solving the OCP.

Next, we test the self-triggered MPC strategy with adaptive prediction horizon on this experimental system. We assume that the disturbance bound is the same as in the event-triggering case. Besides, we set the parameter $\beta = 0.2$. It can be seen from the tracking trajectory in Fig. 8 that the scheme can reach a satisfying performance. The control inputs and the tracking errors are plotted in Fig. 9, which shows that the tracking errors converge to a neighborhood of the origin.

Since the self-triggered MPC scheme is designed based on Lyapunov function (i.e., the cost function), we plot the cost as well as the triggering time and the prediction horizon in Fig. 10. The cost is decreasing by step that is caused by the self-triggering mechanism. At the same time, it shows that the scheme reaches a suboptimal performance compared to periodic time-triggered MPC as shown in Fig. 13. We can also note that the sampling is aperiodic and is reduced compared to traditional MPC, which indicate that the ratio of solving the OCP is reduced. The OCP is solved 368 times by self-triggering strategy. Moreover, the prediction horizon is decreasing as the time goes by, which
implies the reduction of the computational complexity of the OCP in each step.

To show the efficiency of the approaches proposed in this article, we compare our results with that by a standard MPC, i.e., with periodic time-triggering strategy and fixed prediction horizon. The control objective and the parameter settings are the same as aforementioned. Figs. 11 and 12 show the tracking trajectory and control inputs by the standard MPC. It is noted that the performance is slightly better than the event-based MPC proposed in this article. However, the performances by event- and self-triggered MPC in this article are satisfying, which requires much fewer computations. This can be verified by Fig. 13, which shows the optimal cost and the controller’s update times. To further show the efficiency of the shrinking horizon strategy, the computation times by the schemes proposed and by standard MPC are collected in every update instants. To present an intuitive and fair comparison, we plot their distributions in Fig. 14. It is apparently that, from the statistics view, the event-based MPC with adaptive prediction horizon schemes are able to reduce the optimization time due to the decreased complexity of the OCP.

To further illustrate the effectiveness of the approaches proposed, we test our strategies in a lane change maneuver scenario. In this case, both the linear velocity and the angular velocity are varying. The linear velocity is accelerated from 0.03 m/s to 0.06 m/s and is decelerated to 0.04 m/s afterward. The angular velocity varies between $0.2\pi$ and $-0.2\pi$ during the lane change. We use Spline to generate the desired reference trajectory. The desired separation with the reference trajectory is set to be $p_{d} = [0\, m, 0\, m]^T$ with respect to frame $O$ in this scenario and the initial prediction horizon is chosen as $N = 20$. The other parameters are the same as that in the circular trajectory tracking experiments.

Figs. 15 and 16 show the trajectories, the triggering times and the prediction horizons using event- and self-triggering strategies, respectively. Figs. 17 and 18 show the control inputs of event- and self-triggering strategies, respectively. It can be observed that the proposed approaches reach a similar property as that in the circular tracking task. This indicates that the proposed strategies in this article are also able to handle the tracking tasks with higher dynamical characteristics.
C. Analysis and Discussion

From the experimental results, we conclude that both event-triggered and self-triggered MPC schemes with adaptive prediction horizon are able to save the computational resource from the following two aspects: 1) reducing the average frequency of solving the OCP to decrease the computational load; and 2) lowering the dimension of the OCP to relieve the computational complexity. Since the difference of the design approach and triggering mechanism, the proposed two schemes have their own features.

First, it can be seen from the tracking trajectories in Figs. 5 and 8 and tracking errors in Figs. 6 and 9 that the tracking
performance by self-triggered MPC is better than that by event-triggered case. This is mainly caused by their different triggering mechanism. In event-triggered case, the solver is activated when the deviation between the actual trajectory and the optimal one reaches a threshold, while in self-triggered case, the triggering mechanism is designed based on the cost function, i.e., the OCP is resolved when a suboptimal performance is violated.

Second, compared with the event-triggered MPC, the self-triggered MPC is more conservativeness from engineering point of view. Because in self-triggered strategy, the next controller update time is estimated based on the worst case of the disturbances, while in event-triggered control the next triggering time is determined based on the real-time measurement of the actual states. The conservatism can be shown by the triggering times: taking the circular tracking task as an example, the OCP is solved 89% more by self-triggering mechanism than that by event-triggering condition. This also explains the better performance achieved by self-triggered MPC. Meanwhile the self-triggered mechanism does not require continuous states measurement.

Third, by comparing the triggering time distribution in Figs. 7 and 10 (or Figs. 15 and 16), it can be found that in self-triggering condition the controller is triggered more sparse at the beginning and the triggering becomes dense as the cost decreasing. This phenomenon is induced by its triggering mechanism. When the cost is relatively large, more performance loss is allowed. As the cost approaches to zero, less and less permission for the performance loss is allowed to maintain the suboptimal performance.

Finally, we discuss a gap between the actual experiment and the theoretical result. From the prediction horizon update strategy (13), we know that the prediction horizon may decrease to \( N = 2 \), i.e., only one step is required to predict. Nevertheless, in the experiment, we found that when the prediction horizon is reduced to \( N = 2 \), sometimes feasibility will be lost leading to the system unstable. This may be caused by several reasons. First, the upper bound of the external disturbances is beyond calculation. A too large bound estimation results in a too conservatism scheme design, and if this bound is too small, feasibility may not be guaranteed, especially for an only one step prediction. Second, the experimental platform is a typical NCS, and the typical problems in NCS such as packet loss and time delay may also invalid the scheme on the condition that only one step is predicted. Third, the motion capture system may induce unfeasible of the scheme due to frame loss and update delay, etc. To overcome the external uncertainties, we set a lower bound for the prediction as \( N = 5 \) in the experiment, which can be seen from Figs. 7, 10, 15, and 16. This may enhance the robustness of the algorithm.

V. CONCLUSION

Two robust event-based MPC schemes with adaptive prediction horizon have been developed to save the computational resource for tracking of unicycle robots. First, an event-triggered MPC based on the deviation of the actual trajectory and its prediction is developed and a prediction horizon update strategy is presented. Second, a self-triggering mechanism based on Lyapunov theory is introduced into MPC with the same prediction horizon update strategy to relieve the computational load. Its difference from the event-triggered case is that the next update time as well as the next prediction horizon are computed at the previous triggering time. Finally, the proposed schemes are tested on a networked experimental platform in the laboratory, which shows that the average frequency of solving the OCP is reduced and the prediction horizon is decreasing as the tracking error approaches to a neighborhood of the origin.

In this article, the disturbances should be bounded. How to further improve robustness of the proposed approaches will be our future research. Another interesting topic is to introduce learning method into MPC for uncertain systems.

APPENDIX A

PROOF OF LEMMA 1

Proof: Considering the function values of \( f_h(\xi, u) \) at \( \xi_1 \) and \( \xi_2 \) with the same \( u \), we have
\[
\|f_h(\xi_1, u) - f_h(\xi_2, u)\|^2
= v^2(\cos \theta_1 - \cos \theta_2)^2 + \rho^2 \omega^2(\sin \theta_2 - \sin \theta_1)^2
+ v^2(\sin \theta_1 - \sin \theta_2)^2 + \rho^2 \omega^2(\cos \theta_2 - \cos \theta_1)^2
\leq 2(v^2 + \rho^2 \omega^2)(\theta_1 - \theta_2)^2
\leq 2(a^2 + \rho^2 \omega^2)(\theta_1 - \theta_2)^2
\]
where the mean value theorem is used. From the results mentioned above, we conclude that \( \| f_h(\xi_1, u) - f_h(\xi_2, u) \| \leq \| f_h(\xi_1, u) - f_h(\xi_2, u) \| \leq L_P \| \xi_1 - \xi_2 \| \) with \( L_P = \sqrt{2(a^2 + \rho^2 b^2)} \).

**APPENDIX B**

**PROOF OF THEOREM 1**

**Proof:** Take the optimal cost as a Lyapunov function and consider the difference of the cost between \( k \) and \( k+1 \)

\[
V(k+1) - V(k) = J_d(p_e(k+1)), u_e(k+1), N(k+1)) - J_d(p_e(k)), u_e(k), N(k))
\]

\[
= \sum_{i=0}^{N_k} (\| p_e(k+1+i) \|^2 + \| u_e(k+1+i) \|^2)
\]

\[
- \sum_{i=0}^{N_k} (\| p_e(k+i) \|^2 + \| u_e(k+i) \|^2)
\]

\[
+ \| p_e(k+1+N_k+1) \|^2 - \| p_e(k+1+N_k) \|^2.
\]

Then, (32) can be rewritten as

\[
V(k+1) - V(k) = -\sum_{i=0}^{m_k} (\| p_e(k+i) \|^2 + \| u_e(k+i) \|^2)
\]

\[
+ \sum_{i=m_k}^{N_k} (\| p_e(k+i) \|^2 + \| u_e(k+i) \|^2)
\]

\[
+ \| p_e(k+1+N_k+1) \|^2 - \| p_e(k+1+N_k) \|^2.
\]

Using the feasible control input construction technique (see [14]), triangle inequality and the Gronwall–Bellman–Ou–Iang-type inequality, we have the following result:

\[
V(k+1) - V(k) \leq -\sum_{i=0}^{m_k} (\| p_e(k+i) \|^2 + \| u_e(k+i) \|^2)
\]

\[
+ h(m_k, N_k, \eta)
\]

\[
\leq -\| p_e(k) \|^2 + h(\sigma, N_P, \eta)
\]

\[
(34)
\]

where \( h(m_k, N_k, \eta) = \sum_{i=m_k}^{N_k} |i^2 \eta^2 \epsilon^2 e^{2L_P(i-1)} + 2m_k \eta \epsilon^2 \eta^2 e^{L_P(i)} + \sigma \eta^2 \epsilon^2 e^{L_P(N_k)} + (r + \epsilon) \) and \( h(\sigma, N_P, \eta) \) is obviously a \( K_\infty \) function with respect to \( \eta \). This implies that the tracking error will converge to a neighbourhood of the origin, and the range of this neighbourhood is related to the minimum interexecution time \( \sigma \) and the bound of the disturbance \( \eta \).

**APPENDIX C**

**PROOF OF THEOREM 2**

**Proof:** Choose the Lyapunov function candidate as \( V(k_j) = J_d(p_e(k_j), u_e(k_j), N(k_j)) \). From the result (34), it is not difficult to derive the following result:

\[
V(k_j+1) - V(k_j) \leq -\sum_{i=0}^{m_k} L(p_e(k+j), u_e(k+j)) + h(m_k, N_k, \eta)
\]

from which the input-to-state stability of system (4) can be drawn, and the argument is the same as that in event-triggered case.

Next, we show that the tracking error can enter \( \Omega_e \) in finite time. Since the self-triggering condition (31), it follows that

\[
V(k_j+1) - V(k_j) \leq -\beta \sum_{i=0}^{m_k} L(p_e(k+j), u_e(k+j)).
\]

If \( p_e \notin \Omega_e \), inequality (35) satisfies

\[
V(k_j+1) - V(k_j) \leq -\beta \frac{\lambda(Q)}{\lambda(R)} e^2
\]

\[
V(k_j) - V(k_{j-1}) \leq -\beta \frac{\lambda(Q)}{\lambda(R)} e^2
\]

\[
\cdots
\]

\[
V(k_1) - V(k_0) \leq -\beta \frac{\lambda(Q)}{\lambda(R)} e^2.
\]

Summing up the inequalities from (36) to (38) yields

\[
V(k_j+1) \leq V(k_0) - \beta \int_{k_0}^{k_j} \frac{\lambda(Q)}{\lambda(R)} e^2.
\]

This implies \( V(k_j+1) < 0 \) as \( j \to \infty \), which contradicts the fact that Lyapunov function or the cost function is positive definite. Therefore, the tracking error can enter \( \Omega_e \) in finite time.

**REFERENCES**


Zhongqi Sun was born in Hebei Province, China, in 1986. He received the B.S. degree in computer and automation, in 2010, from Hebei Polytechnic University, Hebei, and the Ph.D. degree in control science and engineering, in 2018, from the Beijing Institute of Technology, Beijing, China.

From September 2018 to August 2019, he was a Postdoctoral Fellow with the Faculty of Science and Engineering, University of Groningen, Netherlands. He is currently an Assistant Professor of control science and engineering with the School of Automation, Beijing Institute of Technology. His research interests include intelligent systems, model predictive control, machine learning, and robotic systems.

Yuanqing Xia was born in Anhui Province, China, in 1971. He received the graduate degree in mathematical education from the Department of Mathematics, Chuzu University, China, in 1991, the M.S. degree in fundamental mathematics from Anhui University, China, in 1998, and the Ph.D. degree in control theory and control engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001.

From 1991 to 1995, he was with Tongcheng Middle-School, Anhui, where he was a Teacher. During January 2002–November 2003, he was a Postdoctoral Research Associate with the Institute of Systems Science, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, where he worked on navigation, guidance, and control. From November 2003 to February 2004, he was with the National University of Singapore as a Research Fellow, where he worked on variable structure control. From February 2004 to February 2006, he was with the University of Glamorgan, Pontypridd, U.K., as a Research Fellow, where he worked on networked control systems. From February 2007 to June 2008, he was a Guest Professor with Innsbruck Medical University, Innsbruck, Austria, where he worked on biomedical signal processing. Since 2004, he has been with the Department of Automatic Control, Beijing Institute of Technology, Beijing, first as an Associate Professor, and as a Professor since 2008. In 2012, he was appointed as Xu Teli Distinguished Professor with the Beijing Institute of Technology and obtained a National Science Foundation for Distinguished Young Scholars of China. He has authored or coauthored eight monographs with Springer and Wiley, and more than 100 papers in journals. He is the author of research interests in the fields of networked control systems, robust control and signal processing, active disturbance rejection control, and flight control.

Dr. Xia was the recipient of Second Award of the Beijing Municipal Science and Technology (No. 1) in 2010, Second National Award for Science and Technology (No. 2) in 2011, and Second Natural Science Award of The Ministry of Education (No. 1) in 2012. He is a Deputy Editor for the Journal of the Beijing Institute of Technology, Associate Editor for Acta Automatica Sinica, Control Theory and Applications, the International Journal of Innovative Computing, Information and Control, and International Journal of Automation and Computing.

Li Dai was born in Beijing, China, in 1988. She received the B.S. degree in information and computing science, in 2010, and the Ph.D. degree in control science and engineering, in 2016, from the Beijing Institute of Technology, Beijing, China.

She is currently an Assistant Professor of control science and engineering with the School of Automation, Beijing Institute of Technology. Her research interests include model predictive control, distributed control, data-driven control, and networked control systems.

Pascual Campoy received the master’s degree in automatics engineering and the Ph.D. degree in automatics and robotics from the Universidad Politecnica Madrid (UPM), Madrid, Spain, in 1983 and 1998, respectively. He is currently a Full Professor in automatics engineering and robotics from the Universidad Politecnica Madrid (UPM), Madrid, Spain, where he is also a Lecturer in control, machine learning, and computer vision. He was a Visiting Professor with the DCSC Department, TU Delft, The Netherlands, from 2014 to 2019, and previously a Visiting Professor with Queensland University of Technology, Brisbane City, QLD, Australia, in 2011, and Tong Ji University, Shanghai, China, in 2013. He is leading the Research Group on “Computer Vision and Aerial Robotics,” UPM, within the Centre of Automatics and Robotics, whose activities are aimed at increasing the autonomy of unmanned aerial vehicles (UAV) by exploiting the powerful sensory of vision, using cutting-edge technologies in image processing, control, and artificial intelligence. He has been the Head Director of more than 40 R&D projects, including R&D European projects, national R&D projects, and more than 25 technological transfer projects directly contracted with the industry. He is the author of around 200 international scientific publications and nine patents, three of them are registered internationally.

Dr. Campoy is awarded in the top international UAV competitions: IMAV12, IMAV13, IARC14, IMAV16, and IMAV17, and is the General Chair for IMAV 2019.