Leveraging Experience for Computationally Efficient Adaptive Nonlinear Model Predictive Control

Vishnu R. Desaraju and Nathan Michael

Abstract— **This work presents Experience-driven Predictive Control (EPC) as a fast technique for solving nonlinear model predictive control (NMPC) problems with uncertain system dynamics. EPC leverages an affine dynamics model that is updated online via Locally Weighted Projection Regression (LWPR) to capture nonlinearities, uncertainty, and changes in the system dynamics. This model enables the NMPC problem to be re-cast as a quadratic program (QP). The QP can then be solved via multi-parametric techniques to generate a mapping from state, reference, and dynamics model to a locally optimal, affine feedback control law. These mappings, in conjunction with the basis functions learned via LWPR, define a notion of experience for the controller as they capture the full input-output relationship for previous actions the controller has taken. The resulting experience database allows EPC to avoid solving redundant optimization problems, and as it is constructed online, enables the system to operate more efficiently over time. We demonstrate the performance of EPC through a set of hardware-in-the-loop simulation studies of a quadrotor micro air vehicle that is subjected to unmodeled exogenous perturbations.**

I. Introduction

As robots are deployed in complex and unknown real-world environments, the ability to track trajectories accurately becomes essential for safety. However, accurate tracking can be particularly difficult to achieve if the robot's dynamics change online, e.g., due to environmental effects or hardware degradation. Furthermore, operation in these types of environments may preclude reliable, high-rate communication with a base station, and as a result, the robot must be able to operate safely and reliable with typically limited onboard computational resources. Therefore, in this work we develop a computationally-efficient feedback control strategy that leverages past experiences to enable accurate and reliable operation in the presence of unmodeled system dynamics. The proposed approach employs infrequent online optimization to construct a database of reusable affine feedback controllers that are parameterized by the system dynamics and locally recover the performance of a Nonlinear Model Predictive Controller. Furthermore, we combine this database with an online learned model of the system dynamics to enable adaptation to model perturbations.

High-rate adaptive control is readily achieved via feedback control techniques such as model-reference adaptive control [1] and $\mathcal{L}_1$ adaptive control [2]. However, this simplicity may be at the expense of safety, as such methods do not provide constraint satisfaction guarantees and are purely reactive techniques that seek to eliminate the effects of unmodeled dynamics, even when they may be beneficial. In contrast, model predictive control (MPC) techniques seek to balance the reactive nature of traditional feedback controllers and the anticipative nature of infinite-horizon optimal control techniques. Thus, MPC yields improved trajectory tracking via finite-horizon optimization while reducing computational complexity relative to infinite-horizon formulations.

However, performance of these predictive approaches is largely dependent on the accuracy of the prediction model. When applied to a linear system, or a system that does not deviate significantly from a nominal operating point, the linear MPC problem can be formulated and solved efficiently as either a constrained linear or quadratic program [3]. However, if the operating range deviates greatly from a nominal linearization point, the formulation must account for the nonlinear dynamics to ensure that the optimization is performed with respect to an accurate prediction of the system evolution. Moreover, even a fixed nonlinear model may be insufficient to accurately predict the system's motion due to modeling errors and unmodeled dynamics. The use of a nonlinear dynamics model also significantly increases the computational complexity of the resulting nonlinear MPC (NMPC) problem, which must be formulated as a constrained nonlinear program.

Therefore, there are two key challenges that must be addressed in order to apply NMPC to these challenging, high-rate control problems: (1) maintaining an accurate model of uncertain, time-varying dynamics, and (2) reducing complexity to increase computational efficiency.

A. Model Accuracy

The issue of model accuracy for predictive control has been addressed through various adaptation and learning-based approaches. Most existing adaptive MPC approaches assume a structured system model with uncertain parameters that can be estimated online. These approaches then combine a standard MPC formulation with an online parameter estimator, e.g., a Luenberger observer or Kalman filter, to achieve more accurate, deliberative actions [4]–[6].

However, treating all model uncertainty as estimable parameters can limit the overall model accuracy, particularly when the system is subject to complex, exogenous perturbations, such as aerodynamic effects on an aerial vehicle. Learning-based function approximation techniques can be applied to address this issue. The resulting semi-structured approaches augment a structured system model with a non-parametric, online-learned component, e.g., via a Gaussian

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {rajeswar, nmichael}@cmu.edu
We gratefully acknowledge the support of ARL grant W911NF-08-2-0004.

process [7]. The resulting model is then queried within the NMPC formulation while continuing to adapt to model changes. While techniques such as Gaussian process regression scale poorly with the amount of training data, another kernel-based approach, Locally Weighted Projection Regression (LWPR), summarizes training data using linear basis functions [8]. The resulting incremental updates enable fast model learning that is suitable for finite-horizon control [9].

B. Computational Efficiency

Computational efficiency can be evaluated in terms of increased solution speed and decreased redundant computation. For linear MPC formulations, there are a variety of techniques aimed at increasing solution speed. Several of these approaches leverage efficient convex optimization techniques [10, 11] and exploit matrix structure in the LP or QP formulations [12] to compute solutions quickly. Alternatively, explicit MPC approaches precompute the optimal linear MPC solutions for a polytopic decomposition of the state space, reducing the complexity of online computation [13, 14]. Other approaches, such as partial enumeration (PE) [15], balance the strengths of the online and offline approaches and yield fast solution times on large problems.

While some fast, online NMPC solution techniques have been developed, they rely on iterative, approximate solution techniques built around fast convex optimization solvers [16]–[18]. Consequently, they inherently cannot achieve the solution speeds attained by linear MPC formulations. Explicit NMPC [19] moves the optimization offline to achieve high-speed online control, but it is known to scale poorly as the resulting lookup table grows exponentially with the horizon length and number of constraints. As a result, NMPC has not been amenable to high-rate, real-time operation, particularly on systems with severe computational constraints. The nonlinear partial enumeration (NPE) algorithm [20] combines linear and nonlinear formulations to achieve high-rate predictive control with a nonlinear model, while also improving performance over time to better approximate the NMPC solution. However, its dependence on nonlinear optimization for performance improvement limits scalability and the rate at which performance improves.

While some MPC algorithms seek to reduce the amount of redundant computation performed by reusing past solutions [10], they still must solve an optimization problem at every control iteration. PE-based techniques achieve greater efficiency through the online creation of a controller database that dramatically reduces the number of optimization problems that must be solved. However, as these methods assume the dynamics model is fixed and accurate, the controllers produced are invalidated if the dynamics change.

The construction of a database from past actions in order to facilitate choosing future actions is also the foundation of transfer and lifelong learning algorithms. These learning-based approaches consider executing tasks that, by analogy to the PE approaches, can be viewed as a particular state-reference sequence. Transfer learning seeks to use experience gained from past tasks to bootstrap learning a new task [21], similar to efficient MPC strategies [10]. Lifelong learning shares similarities with the PE approaches in that it makes this experience transfer bidirectional to learn policies that maximize performance over all past and present tasks [22]. However, the PE approaches maintain a finite set of controllers that are updated through infrequent computation and do not permit interpolation. Whereas lifelong learning algorithms, such as ELLA [22] or OMTL [23], maintain a set of bases derived from past experience that aid in reconstructing task models whenever new data is received.

Therefore, we propose an Experience-driven Predictive Control (EPC) methodology that combines aspects of NPE with online model learning via LWPR. As in NPE, EPC leverages an online-updated database of past experiences in order to achieve high-rate, locally-optimal feedback control with constraint satisfaction. However, we also parameterize the learned feedback control laws by the system dynamics, enabling online adaptation to model perturbations as the system accumulates experience. This manuscript refines an earlier workshop presentation [24] and includes an assessment of real-time, hardware-in-the-loop performance.

II. Approach

In this section, we present the Experience-driven Predictive Control (EPC) algorithm for fast, adaptive, nonlinear model predictive control. In the context of predictive control, we first define experience to be the relationship between previous states, references, and system dynamics models and the optimal control law applied at that time. Past dynamics models capture the effects of uncertainty on observed system evolution, while previous states capture the system's behavior under optimal control policies for a given dynamics model. Therefore, EPC constructs and leverages a two-part representation of past experiences to improve the accuracy of its finite-horizon lookahead. The first is the set of linear basis functions maintained by the Locally Weighted Projection Regression (LWPR) algorithm [8] that capture observed variations in the system dynamics. The second is a mapping from states and references to locally optimal controllers that is updated online and is parameterized by the current estimate of the vehicle dynamics.

A. Online Model Adaptation via LWPR

Predictive control techniques for nonlinear systems employ either a nonlinear dynamics model that incurs the complexity of solving nonlinear programs or a more computationally efficient local approximation of the nonlinear dynamics. Therefore, given the nonlinear dynamics $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, nominal state $\mathbf{x}^*$, and nominal control $\mathbf{u}^*$, we define $\bar{\mathbf{x}} = \mathbf{x} - \mathbf{x}^*$ and $\bar{\mathbf{u}} = \mathbf{u} - \mathbf{u}^*$ and derive an affine approximation of the dynamics via a first-order Taylor series expansion, $\bar{\mathbf{x}}^{\text{nom}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c}$. We extend this model with an online-learned component via LWPR that estimates perturbations to the nominal model, including nonlinearities, modeling errors, and unmodeled exogenous forces.

LWPR models a nonlinear function (from an input $\mathbf{z}$ to an output $\mathbf{y}$) by a Gaussian-weighted combination of local linear functions [8]. These basis functions encapsulate all past dynamics information, in contrast to storing all past training data as in a Gaussian process. New linear functions are introduced as required when the existing set of bases are insufficient to represent new data with the desired accuracy. It also has a forgetting factor to control rate of adaptation to model changes by adjusting the effects of prediction error on the weight for each basis. As a result, LWPR is robust to uninformative or redundant data, retains information capturing all past experience, and adapts its estimate to changing dynamics. $\mathbf{T}^{\mathrm{T}}$ LWPR updates its estimate incrementally via partial least squares, with $\mathcal{O}(|\mathbf{z}|)$ complexity, making it well-suited to real-time operation.

Taking $\mathbf{z} = \begin{bmatrix}\mathbf{x}_k^{\mathrm{T}} & \mathbf{u}_k^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$ and $\mathbf{y} = \bar{\mathbf{x}}_{k+1} - \bar{\mathbf{x}}_{k+1}^{\mathrm{nom}}$, the prediction output $\hat{\mathbf{y}}$ gives the estimated perturbation to the nominal dynamics model at a query point $\mathbf{z}$ (where the nominal model is given by the Taylor approximation about $\mathbf{z}$). The total predictive dynamics model is then given by

$$\bar{\mathbf{x}}_{k+1} = \bar{\mathbf{x}}_{k+1}^{\mathrm{nom}} + \hat{\mathbf{y}} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \mathbf{c} + \hat{\mathbf{y}} \tag{1}$$

As LWPR learns the perturbation model online, it may initially return high-variance estimates when the system enters a new region of the input space (i.e., values of $\mathbf{z}$ for which the system has minimal experience). Therefore, to limit the effects of the resulting transients in the estimate, we introduce a simple gate based on the model uncertainty maintained by LWPR. If model uncertainty is high at a given query point, we instead use a zero-order hold on the previous estimate. As the system continues to gain experience in its operating domain, this gate will cease to be applied.

Finally, following the insight from $\mathcal{L}_1$ adaptive control [2], we introduce a low-pass filter on the disturbance estimate before it is incorporated into the predictive model (1). This formulation enables LWPR to learn the perturbation model quickly while limiting changes to system dynamics to be within the bandwidth of the system.

B. Receding-Horizon Control Formulation

The use of an affine model (1) that automatically adapts to capture the effects of nonlinearities and unmodeled dynamics permits a simplified optimal control formulation for EPC relative to techniques such as nonlinear partial enumeration (NPE) that require solving a nonlinear program due to the general nonlinear dynamics model. Taking the current state as the nominal state, $\mathbf{x}^* = \mathbf{x}_0$, and given N reference states $\mathbf{r}_1, \ldots, \mathbf{r}_N$, let $\bar{\mathbf{r}} = \mathbf{r} - \mathbf{x}^*$. We formulate the receding-horizon control problem as a quadratic program:

$$\begin{aligned}
\underset{\bar{\mathbf{u}}_k}{\operatorname{argmin}} \sum_{k=0}^{N-1} & \frac{1}{2}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1})^{\mathrm{T}}\mathbf{Q}(\bar{\mathbf{x}}_{k+1} - \bar{\mathbf{r}}_{k+1}) \\
& + \frac{1}{2}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{y}}})^{\mathrm{T}}\mathbf{R}(\bar{\mathbf{u}}_k - \bar{\mathbf{u}}_{\hat{\mathbf{y}}}) \\
\text{s.t.} \quad & \bar{\mathbf{x}}_{k+1} = \mathbf{A}\bar{\mathbf{x}}_k + \mathbf{B}\bar{\mathbf{u}}_k + \tilde{\mathbf{c}} \\
& \mathbf{G}_{\mathbf{x}}\bar{\mathbf{x}}_{k+1} \leq \mathbf{g}_{\mathbf{x}} \\
& \mathbf{G}_{\mathbf{u}}\bar{\mathbf{u}}_k \leq \mathbf{g}_{\mathbf{u}} \\
& \forall\ k = 0, \ldots, N-1
\end{aligned} \tag{2}$$

where $\tilde{\mathbf{c}} = \mathbf{c} + \hat{\mathbf{y}}$. If a control input, $\bar{\mathbf{u}}_{\hat{\mathbf{y}}}$, can be derived from the model adaptation term (e.g., if $\hat{\mathbf{y}}$ is an acceleration disturbance, $\bar{\mathbf{u}}_{\hat{\mathbf{y}}}$ is the corresponding force) we subtract it in the cost function to avoid penalizing disturbance compensation.

To simplify notation, define $\boldsymbol{x} = \begin{bmatrix}\bar{\mathbf{x}}_1^{\mathrm{T}}, \ldots, \bar{\mathbf{x}}_N^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{r} = \begin{bmatrix}\bar{\mathbf{r}}_1^{\mathrm{T}}, \ldots, \bar{\mathbf{r}}_N^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{u} = \begin{bmatrix}\bar{\mathbf{u}}_0^{\mathrm{T}}, \ldots, \bar{\mathbf{u}}_{N-1}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, $\boldsymbol{u}_{\hat{\mathbf{y}}} = \begin{bmatrix}\bar{\mathbf{u}}_{\hat{\mathbf{y}}}^{\mathrm{T}}, \ldots, \bar{\mathbf{u}}_{\hat{\mathbf{y}}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$

$$\boldsymbol{\mathcal{B}} = \begin{bmatrix} \mathbf{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{AB} & \mathbf{B} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \ldots & \mathbf{B} \end{bmatrix}, \ \boldsymbol{c} = \begin{bmatrix} \tilde{\mathbf{c}} \\ (\mathbf{A}+\mathbf{I})\,\tilde{\mathbf{c}} \\ \vdots \\ \sum_{i=0}^{N-1}\mathbf{A}^i\tilde{\mathbf{c}} \end{bmatrix},$$

$\boldsymbol{\mathcal{Q}} = \operatorname{diag}(\mathbf{Q}, \ldots, \mathbf{Q})$, $\boldsymbol{\mathcal{R}} = \operatorname{diag}(\mathbf{R}, \ldots, \mathbf{R})$, $\boldsymbol{\mathcal{G}}_{\mathbf{x}} = \operatorname{diag}(\mathbf{G}_{\mathbf{x}}, \ldots, \mathbf{G}_{\mathbf{x}})$, $\boldsymbol{\mathcal{G}}_{\mathbf{u}} = \operatorname{diag}(\mathbf{G}_{\mathbf{u}}, \ldots, \mathbf{G}_{\mathbf{u}})$, $\boldsymbol{g}_{\mathbf{x}} = \begin{bmatrix}\mathbf{g}_{\mathbf{x}}^{\mathrm{T}}, \ldots, \mathbf{g}_{\mathbf{x}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$, and $\boldsymbol{g}_{\mathbf{u}} = \begin{bmatrix}\mathbf{g}_{\mathbf{u}}^{\mathrm{T}}, \ldots, \mathbf{g}_{\mathbf{u}}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}}$. Also, noting that $\bar{\mathbf{x}}_0 = \mathbf{0}$, we can rewrite (2) as

$$\begin{aligned}
\underset{\boldsymbol{u}}{\operatorname{argmin}} \ & \frac{1}{2}(\boldsymbol{x} - \boldsymbol{r})^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{x} - \boldsymbol{r}) + \frac{1}{2}(\boldsymbol{u} - \boldsymbol{u}_{\hat{\mathbf{y}}})^{\mathrm{T}}\boldsymbol{\mathcal{R}}(\boldsymbol{u} - \boldsymbol{u}_{\hat{\mathbf{y}}}) \\
\text{s.t.} \quad & \boldsymbol{x} = \boldsymbol{\mathcal{B}}\boldsymbol{u} + \boldsymbol{c} \\
& \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{x} \leq \boldsymbol{g}_{\mathbf{x}} \\
& \boldsymbol{\mathcal{G}}_{\mathbf{u}}\boldsymbol{u} \leq \boldsymbol{g}_{\mathbf{u}}
\end{aligned}$$

We can construct an equivalent QP entirely in terms of $\boldsymbol{u}$ by substituting the dynamics constraints and dropping constant terms in the cost function

$$\begin{aligned}
\underset{\boldsymbol{u}}{\operatorname{argmin}} \ & \frac{1}{2}\boldsymbol{u}^{\mathrm{T}}\boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h}^{\mathrm{T}}\boldsymbol{u} \\
\text{s.t.} \quad & \boldsymbol{\Gamma}\boldsymbol{u} \leq \boldsymbol{\gamma}
\end{aligned} \tag{3}$$

where $\boldsymbol{\mathcal{H}} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}\boldsymbol{\mathcal{B}} + \boldsymbol{\mathcal{R}}$, $\boldsymbol{h} = \boldsymbol{\mathcal{B}}^{\mathrm{T}}\boldsymbol{\mathcal{Q}}(\boldsymbol{c} - \boldsymbol{r}) - \boldsymbol{\mathcal{R}}\boldsymbol{u}_{\hat{\mathbf{y}}}$,

$$\boldsymbol{\Gamma} = \begin{bmatrix}\boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{\mathcal{B}} \\ \boldsymbol{\mathcal{G}}_{\mathbf{u}}\end{bmatrix}, \text{ and } \boldsymbol{\gamma} = \begin{bmatrix}\boldsymbol{g}_{\mathbf{x}} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}\end{bmatrix}$$

Defining $\boldsymbol{\lambda}$ as the vector of Lagrange multipliers and $\boldsymbol{\Lambda} = \operatorname{diag}(\boldsymbol{\lambda})$, the first two Karush-Kuhn-Tucker (KKT) optimality conditions (stationarity and complementary slackness) for the QP can then be written as

$$\begin{aligned}
\boldsymbol{\mathcal{H}}\boldsymbol{u} + \boldsymbol{h} + \boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{\lambda} &= \mathbf{0} \\
\boldsymbol{\Lambda}(\boldsymbol{\Gamma}\boldsymbol{u} - \boldsymbol{\gamma}) &= \mathbf{0}
\end{aligned} \tag{4}$$

If we only consider the active constraints (i.e., with $\lambda > 0$) for a given solution, we can reconstruct $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ by solving a linear system derived from (4), where the subscript a indicates rows corresponding to the active constraints

$$\begin{bmatrix}\boldsymbol{\mathcal{H}} & \boldsymbol{\Gamma}_a^{\mathrm{T}} \\ \boldsymbol{\Gamma}_a & \mathbf{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{u} \\ \boldsymbol{\lambda}_a\end{bmatrix} = \begin{bmatrix}-\boldsymbol{h} \\ \boldsymbol{\gamma}_a\end{bmatrix}$$

Assuming the active constraints are linearly independent (Bemporad, et al. [13] suggest alternatives if this assumption fails), the resulting QP control law, $\boldsymbol{u}$, is affine in the predicted state error, $\boldsymbol{r}$, and parameterized by the system dynamics

$$\boldsymbol{u} = \boldsymbol{\mathcal{E}}_5\boldsymbol{r} - \left(\boldsymbol{\mathcal{E}}_5\boldsymbol{c} - \boldsymbol{\mathcal{E}}_4\boldsymbol{\mathcal{R}}\boldsymbol{u}_{\hat{\mathbf{y}}} + \boldsymbol{\mathcal{E}}_3\begin{bmatrix}\boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}}\boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-}\end{bmatrix}_a\right) \tag{5}$$

where $\boldsymbol{\mathcal{E}}_1 = \boldsymbol{\Gamma}_a \boldsymbol{\mathcal{H}}^{-1}$, $\boldsymbol{\mathcal{E}}_2 = -(\boldsymbol{\mathcal{E}}_1 \boldsymbol{\Gamma}_a^{\mathrm{T}})^{-1}$, $\boldsymbol{\mathcal{E}}_3 = \boldsymbol{\mathcal{E}}_1^{\mathrm{T}} \boldsymbol{\mathcal{E}}_2$, $\boldsymbol{\mathcal{E}}_4 = \boldsymbol{\mathcal{H}}^{-1} + \boldsymbol{\mathcal{E}}_3 \boldsymbol{\mathcal{E}}_1$, and $\boldsymbol{\mathcal{E}}_5 = \boldsymbol{\mathcal{E}}_4 \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$. Moreover, since the coefficients in (5) are all functions of $\mathbf{A}$, $\mathbf{B}$, and $\tilde{\mathbf{c}}$, the overall control law $\boldsymbol{\kappa}(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N)$ can be written in terms of a parameterized feedback gain matrix $\mathbf{K}$ and feedforward vector $\mathbf{k}_{\mathrm{ff}}$

$$\boldsymbol{\kappa}(\mathbf{x}_0, \mathbf{r}_1, \ldots, \mathbf{r}_N) = \mathbf{K}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}})\boldsymbol{r} + \mathbf{k}_{\mathrm{ff}}(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}) \quad (6)$$

This parameterization also extends to the KKT condition checks to determine whether a previously computed controller is locally optimal. The active Lagrange multipliers $\boldsymbol{\lambda}_a$ follow a similar form to the control law

$$\boldsymbol{\lambda}_a = -\boldsymbol{\mathcal{E}}_6 \boldsymbol{r} + \left(\boldsymbol{\mathcal{E}}_6 \boldsymbol{c} - \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{R}} \boldsymbol{u}_{\hat{\mathbf{y}}} + \boldsymbol{\mathcal{E}}_2 \begin{bmatrix} \boldsymbol{g}_{\mathbf{x}}^{+} - \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ -\boldsymbol{g}_{\mathbf{x}}^{-} + \boldsymbol{\mathcal{G}}_{\mathbf{x}} \boldsymbol{c} \\ \boldsymbol{g}_{\mathbf{u}}^{+} \\ -\boldsymbol{g}_{\mathbf{u}}^{-} \end{bmatrix}_a \right) \quad (7)$$

where $\boldsymbol{\mathcal{E}}_6 = \boldsymbol{\mathcal{E}}_3^{\mathrm{T}} \boldsymbol{\mathcal{B}}^{\mathrm{T}} \boldsymbol{\mathcal{Q}}$.

Therefore, instead of storing the affine controller gains and Lagrange multipliers required to evaluate the KKT conditions, it is sufficient to store only the set of active constraints. This enables a memory-efficient implementation for constrained systems. The controller and KKT matrices can then be reconstructed online using (5), (7), and the current $\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}}$. Consequently, this parameterized formulation enables us to adapt and apply any previously computed controller, when appropriate according to the KKT conditions, even as the system dynamics evolve. The complete algorithm is detailed below.

C. EPC Algorithm

As described in Alg. 1, EPC constructs a database defined as a mapping $\mathcal{M}$ from experiences to controllers. At the beginning of each control iteration, EPC queries the current state and reference, as well as the current affine model from LWPR, $(\mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}})$. It then queries the parameterized mapping (line 6), and if the KKT conditions are met for an element, applies the corresponding controller. If no controller from prior experience is applicable (line 14), it solves the QP (3) to add a new parameterized element to the mapping, updating the stored experiences with the current scenario. In parallel, EPC applies commands from a short-horizon intermediate QP with slack on state constraints (to ensure problem feasibility), in order to maintain a desired control update rate (line 15). As new controllers are added to the database, less valuable controllers (indicated by a lower `importance` score) are removed (line 19) to bound the number of elements that may be queried in one control iteration, thereby ensuring computational tractability.

In addition to introducing adaptation to unmodeled dynamics, the parameterization by experience and the introduction of an online updated linear dynamics model eliminates the most computationally expensive component of NPE - the nonlinear program. Although the nonlinear program does not limit the control rate in NPE, it does limit how quickly new controllers can be computed, consequently limiting the practical horizon length and increasing the dependence on the intermediate controller. With its quadratic program formulation, EPC has the advantage of faster solution times in the parallel thread that can be leveraged to reduce the dependence on the intermediate controller or increase the prediction horizon. Additionally, the nonlinear program solutions in NPE serve as fixed feedforward terms in the resulting affine control laws, precluding a completely adaptive control strategy. With EPC, the local controllers are fully parameterized, allowing controllers computed using past experience to be adapted to the present scenario.

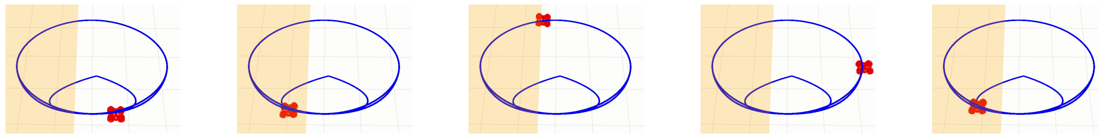
Algorithm 1 Experience-driven Predictive Control

1: $\mathcal{M} \leftarrow \emptyset$ or $\mathcal{M}_{\text{prior}}$
2: **while** control is enabled **do**
3: $\quad \boldsymbol{x} \leftarrow$ current system state
4: $\quad \boldsymbol{r} \leftarrow$ current reference sequence
5: $\quad \mathbf{A}, \mathbf{B}, \tilde{\mathbf{c}} \leftarrow$ current dynamics model from LWPR
6: $\quad$ **for** each element $m_i \in \mathcal{M}$ **do**
7: $\quad\quad$ Compute $\boldsymbol{u}, \boldsymbol{\lambda}_a$ via (5),(7)
8: $\quad\quad$ **if** $\boldsymbol{x}, \boldsymbol{r}$ satisfy parameterized KKT criteria **then**
9: $\quad\quad\quad$ `importance`$_i \leftarrow$ current time, sort $\mathcal{M}$
10: $\quad\quad\quad$ `solution_found` $\leftarrow$ true
11: $\quad\quad\quad$ Apply affine control law (6) from m_i
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad$ **if** `solution_found` is false **then**
15: $\quad\quad$ Apply interm. control via (3) with slack variables
16: $\quad\quad$ Update QP formulation with current model
17: $\quad\quad$ Generate new controller via QP (3) (in parallel)
18: $\quad\quad$ **if** $|\mathcal{M}| >$ maximum table size **then**
19: $\quad\quad\quad$ Remove element with min. `importance`
20: $\quad\quad$ **end if**
21: $\quad\quad$ Add $m_{\text{new}} = (\mathbf{K}, \mathbf{k}_{\mathrm{ff}},$ `importance`$)$ to $\mathcal{M}$
22: $\quad$ **end if**
23: **end while**

III. Results

To validate the performance of the EPC algorithm, we seek to demonstrate the following results: **R1:** stable control performance with constraint satisfaction, **R2:** real-time computation of control commands, **R3:** adaptation performance, and **R4:** applicability of experiences to novel scenarios. Thus, we conduct a series of hardware-in-the-loop simulations of a quadrotor micro air vehicle tracking trajectories that cross a region where strong, exogenous disturbances (e.g., wind) act on the vehicle.

Experimental Design and Implementation Details: The simulator and controller are built around ROS [25], and the controller uses the qpOASES library [26] to solve the quadratic programs. To assess performance on a constrained computational platform, the simulation trials are run on an ODROID-XU4 (2 GHz ARM processor with 2 GB RAM) that satisfies the size, weight, and power limitations of a small, 750 g quadrotor. We employ a standard hierarchical



(a) (b) (c) (d) (e)

Fig. 1: Snapshots of the quadrotor executing the elliptical trajectory that traverses the disturbance region (highlighted).

control setup [27], applying EPC separately to the translational and rotational dynamics.

The quadrotor is commanded to fly ten laps at 0.7 m/s around an elliptical trajectory (Fig. 1) that intersects a region in which a constant disturbance torque is applied about the x and y axes. Given that the disturbance acts on the rotational dynamics, we focus on the EPC used for attitude control in the following results. The attitude dynamics are modeled by six states (body frame Euler angles and rates) with one torque input about each axis [27], and we select a horizon (N) of 15 control iterations to ensure that the predicted state evolution captures the effects of the control inputs. As attitude controllers are commonly run at rates exceeding 200 Hz to ensure stability of these fast dynamics [20], we note that a viable attitude controller must consistently return a control command within 5 ms.

Constraint Satisfaction: To demonstrate safety under limited control authority, we enforce constraints on the torque control inputs that are more restrictive than the nominal commands that would be applied to track the trajectory. As a result, these constraints are activated repeatedly as the vehicle tracks the trajectory. In order to satisfy these constraints, EPC learns 22 different parameterized feedback control laws, as shown in Fig. 2. Moreover, the intermediate controller (denoted controller 0) is only applied in the early laps, indicating that the majority of the controllers are learned quickly and then reused in subsequent laps. This intelligent controller switching also yields reliable constraint satisfaction (**R1**), as shown in Fig. 3.

Real-time Computation: Over the course of this trial, the mean time required to query the controller database is 0.77 ms with a standard deviation of 0.75 ms. In contrast, the mean time to solve the equivalent QP is 4.7 ms with a standard deviation of 3.2 ms, which violates the consistent 5 ms command requirement. This confirms that EPC is a computationally efficient approach for adaptive nonlinear model predictive control suitable for high-rate applications, such as attitude control of a quadrotor, even on computationally constrained platforms (**R2**).

Adaptation Performance: In addition to constraint satisfaction, EPC substantially improves trajectory tracking accuracy in the presence of sudden changes to the system dynamics, as shown in Fig. 4. As expected, tracking performance improves over time with the accumulation of experience. In addition to extending the controller database, this experience refines the LWPR model. Consequently, the model yields increasingly accurate estimates of the exogenous torques, as shown in Fig. 5.

Figure 6 illustrates the performance of EPC relative to

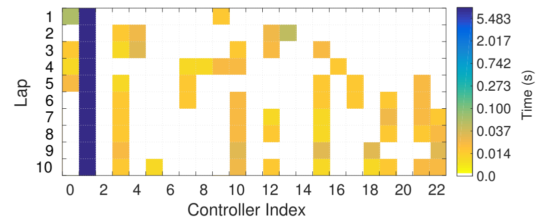


Fig. 2: Learned controllers are reused in subsequent laps, ultimately eliminating the dependence on the intermediate controller (column 0). Colors denote the total usage time (in seconds) for each controller.

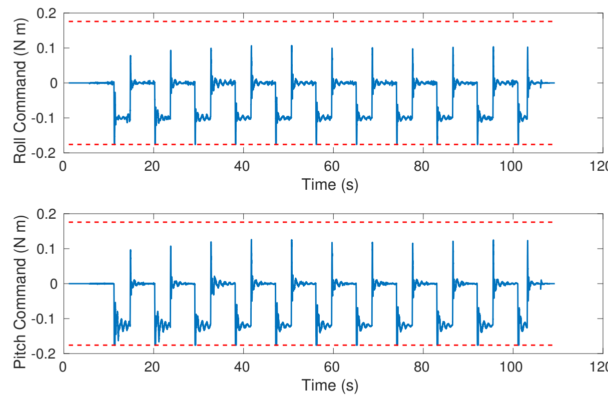


Fig. 3: EPC successfully satisfies roll and pitch control input constraints (dashed red lines) via controller switching.

two baseline approaches: $\mathcal{L}_1$ adaptive control (L1AC) [2] and an adaptive MPC formulation based on a state predictor (Luenberger observer). The gains for the L1AC are selected to match the nominal gains computed by EPC. The low-pass filter bandwidth is equivalent for both controllers to ensure a fair comparison of the adaptation laws. As the core EPC formulation is equivalent to a quadratic program-based MPC, we consider EPC with the Luenberger observer as the second baseline. Additionally, while EPC embeds the disturbance estimate in the prediction model to enable constraint satisfaction, L1AC adds it as a compensation term to the resulting command. It therefore lacks any safe means of constraint satisfaction, precluding a comparison of constrained control performance. We therefore loosen the EPC control input constraints to aid comparison.

As Fig. 6 shows, EPC (after obtaining sufficient experience) reduces peak tracking error by an average of 26.8% relative to $\mathcal{L}_1$ adaptive control. EPC (with LWPR) also reduces peak tracking error by an average of 17.2% relative to the variant with a Luenberger observer, confirming that

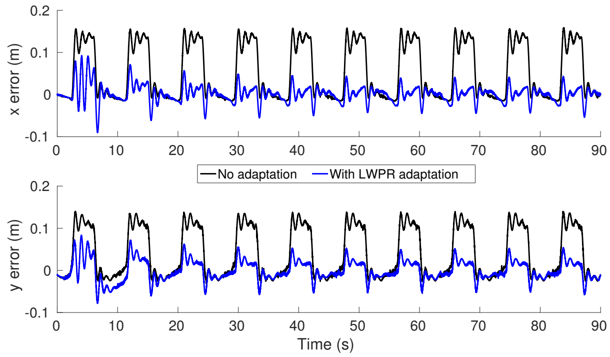


Fig. 4: Comparison of EPC tracking performance with and without LWPR-based adaptation.

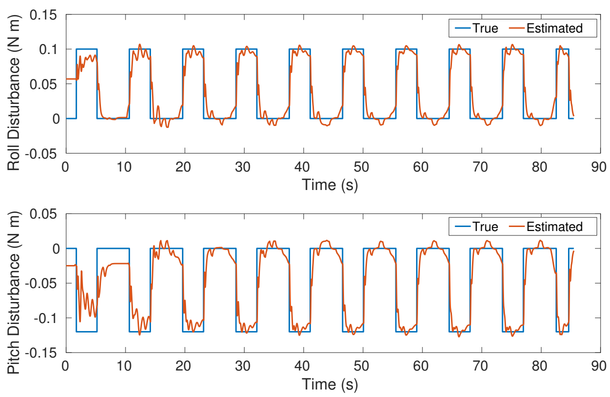


Fig. 5: LWPR accurately estimates the torque disturbances about the x- and y-axes as it tracks the elliptical trajectory.

the improvement relative to L1AC is not simply due to integrating the estimate into the prediction model. Moreover, these results show that the combination of a predictive controller driven by an online learned, reusable model yields significantly improved tracking performance (**R3**).

Application to Novel Scenarios: Finally, to evaluate the generalizability of experience, we consider a more complex scenario. Over the course of this 1000 s trial, the quadrotor is commanded to track a series of smooth but random trajectories through the same environment as before. Figures 7 and 8 show these trajectories, which achieve maximum commanded velocities of 1.7 m/s and accelerations of 5.1 m/s^2. The vehicle dynamics are also perturbed by a stochastic process emulating turbulent air flow, introducing noise into the LWPR training data.

Due to the randomization, the quadrotor enters and exits the disturbance region following a variety of trajectories. The resulting disturbance estimate (Fig. 9) shows transient behavior during the initial traversals of the disturbance region (e.g. during the first 200 s of the trial), with disturbance estimate rise times greater than 1.5 s. However, these transients do not reappear, even as the vehicle traverses the region in previously unseen ways while executing this diverse set of trajectories. Moreover, the disturbance estimate has a consistent rise time of approximately 0.5 s for the remainder of the trial (**R3**). This indicates that the experience gained through

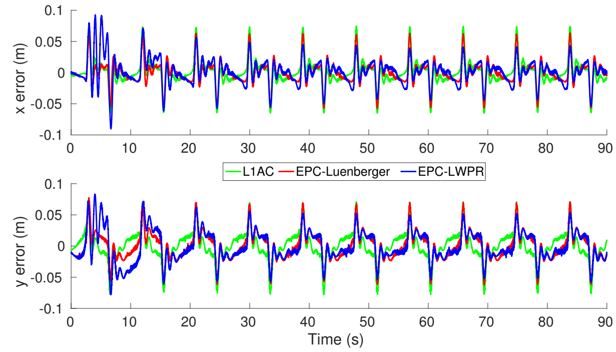


Fig. 6: EPC with LWPR yields improved position tracking error compared to $\mathcal{L}_1$ adaptive control (L1AC) and EPC with a simple state predictor (EPC-Luenberger).

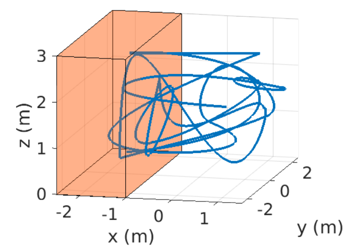


Fig. 7: Representative trajectories entering and exiting the disturbance region (highlighted), taken from a 100 s window of the randomized trial.

the initial traversals is applicable to the numerous novel scenarios encountered in the future and yields a consistent improvement in disturbance estimation performance (**R4**).

The controller also performs as expected (**R1**). Even for this long trial with diverse trajectories, EPC only computes 52 controllers to maintain constraint satisfaction (see Fig. 10) and achieves comparable query times to the previous trial. These results again illustrate the computational efficiency of this Experience-driven Predictive Control approach and its suitability for use on flight-viable constrained computational platforms (**R2**).

IV. Conclusions and Future Work

In this work, we present the Experience-driven Predictive Control (EPC) algorithm for fast, adaptive, nonlinear model predictive control. EPC constructs a database of reusable feedback controllers that are parameterized by the system dynamics. When combined with an online-learned model of the system dynamics using Locally-Weighted Projection Regression (LWPR), this enables online adaptation to perturbations to the dynamics model. As the system gains experience through operation, both the controller database and the dynamics model are improved to yield increased tracking accuracy, even in the presence of sudden changes in the dynamics model. This also implies that if the system is initialized with prior experience (e.g., from past operation), it could further reduce the transient effects of learning.

The hardware-in-the-loop simulation trials presented in this work provide a preliminary assessment of the EPC algo-

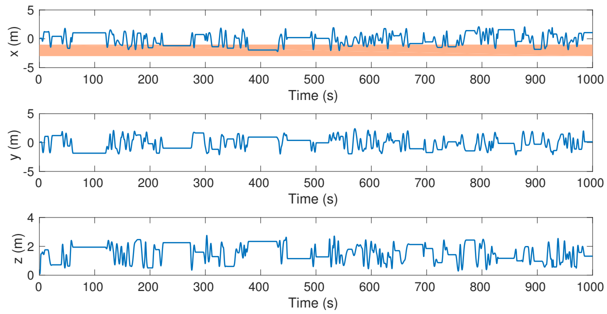


Fig. 8: Reference trajectory components for the randomized trial with the disturbance region highlighted along the x-axis

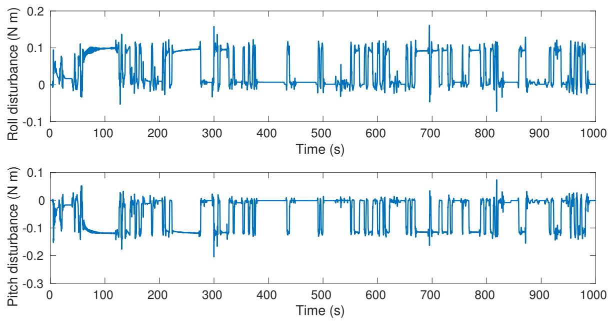


Fig. 9: Roll and pitch disturbance estimates for the randomized trial show an initial transient but have consistent performance for the remainder of the trial

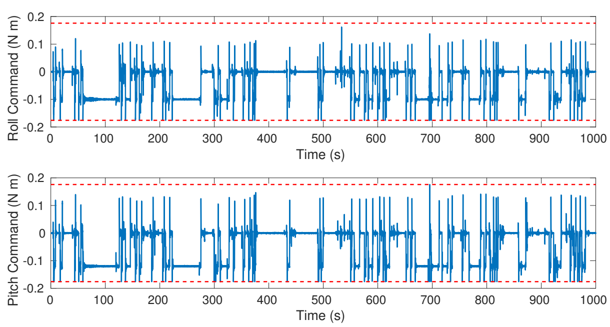


Fig. 10: EPC satisfies control input constraints for the entire duration of the randomized trial while tracking a diverse set of trajectories

rithm and demonstrate its viability for real-time operation on computationally constrained platforms. We have equipped a small-scale quadrotor with this computational platform and intend to pursue experimental validation of the approach in the presence of real-world unmodeled dynamics.

References

[1] K. S. Narendra and A. M. Annaswamy, *Stable Adaptive Systems*. Prentice Hall, 1989.

[2] J. Wang, F. Holzapfel, E. Xargay, and N. Hovakimyan, "Non-Cascaded Dynamic Inversion Design for Quadrotor Position Control with L1 Augmentation," in *Proc. of the CEAS Specialist Conf. on Guidance, Navigation & Control*, Delft, Netherlands, Apr. 2013.

[3] J. Löfberg, "Minimax approaches to robust model predictive control," Ph.D. dissertation, Linköping University, 2003.

[4] H. Fukushima, T. Kim, and T. Sugie, "Adaptive model predictive control for a class of constrained linear systems based on the comparison model," *Automatica*, vol. 43, no. 2, pp. 301–308, 2007.

[5] P. Bouffard, A. Aswani, and C. Tomlin, "Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, St. Paul, MN, May 2012, pp. 279–284.

[6] Y. K. Ho, H. K. Yeoh, and F. S. Mjalli, "Generalized Predictive Control Algorithm with Real-Time Simultaneous Modeling and Tuning," *Indust. & Eng. Chem. Research*, vol. 53, no. 22, pp. 9411–9426, 2014.

[7] C. Ostafew, A. Schoellig, and T. Barfoot, "Learning-Based Nonlinear Model Predictive Control to Improve Vision-Based Mobile Robot Path-Tracking in Challenging Outdoor Environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, May 2014, pp. 4029–4036.

[8] S. Vijayakumar, A. DSouza, and S. Schaal, "Incremental Online Learning in High Dimensions," *Neural Comp.*, vol. 17, no. 12, pp. 2602–2634, 2005.

[9] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Adaptive optimal feedback control with learned internal dynamics models," *From Motor Learn. to Inter. Learn. in Rob.*, vol. 264, pp. 65–84, 2010.

[10] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *Intl. J. of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, May 2008.

[11] S. Richter, C. N. Jones, and M. Morari, "Computational Complexity Certification for Real-Time MPC With Input Constraints Based on the Fast Gradient Method," *IEEE Trans. Autom. Control*, vol. 57, no. 6, pp. 1391–1403, 2012.

[12] Y. Wang and S. Boyd, "Fast Model Predictive Control Using Online Optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[13] A. Bemporad, V. D. M. Morari, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, pp. 3–20, 2002.

[14] A. Domahidi, M. N. Zeilinger, M. Morari, and C. N. Jones, "Learning a Feasible and Stabilizing Explicit Model Predictive Control Law by Robust Optimization," in *Proc. of the IEEE Conf. on Decision and Control*, Dec. 2011, pp. 513–519.

[15] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852–860, 2007.

[16] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H. G. Bock, E.-D. Gilles, and J. P. Schlöder, "An Efficient Algorithm for Nonlinear Model Predictive Control of Large-Scale Systems Part I: Description of the Method," *Automatisierungstechnik*, vol. 50, no. 12, pp. 557–567, 2012.

[17] B. Houska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, 2011.

[18] F. Debrouwere, M. Vukov, R. Quirynen, M. Diehl, and J. Swevers, "Experimental validation of combined nonlinear optimal control and estimation of an overhead crane," in *Proc. of the Intl. Fed. of Autom. Control*, Cape Town, South Africa, Aug. 2014, pp. 9617–9622.

[19] A. Grancharova and T. Johansen, *Explicit Nonlinear Model Predictive Control*. Springer Berlin Heidelberg, 2012, vol. 429.

[20] V. Desaraju and N. Michael, "Fast Nonlinear Model Predictive Control via Partial Enumeration," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016, pp. 1243–1248.

[21] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *J. of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[22] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Intl. Conf. on Machine Learning*, 2013, pp. 507–515.

[23] A. Saha, P. Rai, H. Daumé, and S. Venkatasubramanian, "Online learning of multiple tasks and their relationships," in *Intl. Conf. on Artif. Intell. and Stat.*, 2011, pp. 643–651.

[24] V. R. Desaraju and N. Michael, "Experience-driven Predictive Control," in *Robot Learn. and Plan. Workshop at RSS*, Ann Arbor, MI, June 2016.

[25] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, 2009.

[26] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[27] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "Experimental evaluation of multirobot aerial control algorithms," *IEEE Robotics & Automation Magazine*, pp. 56–65, Sept. 2010.