Learning-Based Fast Nonlinear Model Predictive Control for Custom-Made 3D Printed Ground and Aerial Robots

Mohit Mehndiratta, Erkan Kayacan, Siddharth Patel, Erdal Kayacan, and Girish Chowdhary

1 Introduction

In almost all robotic applications, there are always time-varying system dynamics and/or environmental variations throughout the operation. For instance, off-road agricultural robots, including fruit picking robots, driverless tractors, and sheep shearing robots, must be operated on varying soil conditions. Furthermore, there are always topological challenges, such as bumps and hollows in a field. All these challenges bring additional uncertainties to the system which can be modeled as longitudinal and lateral slip variations [17]. Since the performance of a model-based controller is guaranteed for an accurate mathematical model of the system, any plant-model mismatch results in suboptimal performance. Therefore, for a guaranteed performance from a model-based controller, the aforementioned variations must be learnt over time, and the controller must adapt itself to the changing conditions autonomously. Another example is mass variations in package delivery problems of aerial robots. When the total mass of a multi-rotor unmanned aerial vehicle (UAV) is considered, the payload changes may result in massive variations in its

M. Mehndiratta · S. Patel
Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798
e-mail: mohit005@e.ntu.edu.sg; PATE0006@e.ntu.edu.sg

E. Kayacan (✉)
Massachusetts Institute of Technology, Cambridge, MA 02139 USA
e-mail: erkank@mit.edu

E. Kayacan
Aarhus University, Department of Engineering, DK-8000 Aarhus C, Denmark
e-mail: erdal@eng.au.dk

G. Chowdhary
University of Illinois at Urbana–Champaign, Champaign, IL, USA
girishc@illinois.edu

S. V. Raković, W. S. Levine (eds.), *Handbook of Model Predictive Control*,
Control Engineering, https://doi.org/10.1007/978-3-319-77489-3_24

dynamic model which will also result in suboptimal performance for a model-based controller. Motivated by the challenges listed above, our goal is to use an online learning-based nonlinear model predictive control (NMPC) for systems with uncertain and/or time-varying dynamic models.

As a solution to modeling mismatch problem between the plant to be controlled and its corresponding mathematical model, adaptation of either controller parameters or deployed mathematical model parameters is not a novel idea. In adaptive control, controller adapts itself systematically to compensate lack of modeling due to uncertain and/or time-varying parameters. This feature, apparently, exterminates the effect of parameter uncertainties on the closed-loop system's performance [7]. A well-utilized strategy in this area is the adaptive-optimal control, which comprises of the use of an adaptive controller for stability during the learning phase, followed by the switch to the main model-based optimal controller that eventually optimizes the performance. An online switching metric is developed that initiates the switching to model predictive control (MPC) after gaining enough confidence in the parameter estimates, as realized in [5, 6]. On the other hand, an alternative learning approach could be to combine the control with some optimization-based estimation scheme including predictive filtering and moving horizon estimation (MHE) [1], which is also the case in this work. These estimators are model-based estimators, which can incorporate parameter variations along with the state estimation, to learn the uncertain system parameters online. This learning-based NMPC has been utilized for numerous robotic applications including constrained path tracking of a mobile robot in [26], control of a 3 degree of freedom helicopter in [23], control of lateral dynamics of a fixed-wing UAV in [30], control of a quadrotor in [4], teleoperation of an underwater vehicle in [11], and robust obstacle avoidance in [9, 21].

In addition to MHE, extended Kalman filter (EKF) can also be utilized for online learning. However, EKF is based on the linearization of the nonlinear system at the current estimate and is only suitable for unconstrained problems. In other words, EKF might give irrational estimation results, e.g. less than zero or larger than one for slip parameters [13, 15, 16, 18]. On the contrary, MHE strategy exploits the past measurements available over a window and solves an optimization problem to estimate the system's states and unknown parameters [22]. Additionally, MHE is a powerful nonlinear estimator that is not only suitable for non-Gaussian disturbance but is also competent in handling constraints explicitly [28]. This implies, MHE will never give irrational estimation results for the aforementioned slip parameters [29].

In this work, the efficacy of the learning-based NMPC is elaborated for the trajectory tracking of two custom-made 3D printed robotic platforms: an off-road agricultural ground vehicle and an aerial robot for package delivery problem. In the first application, NMPC is utilized for controlling a field robot in an off-road terrain. Since the ground conditions, including surface quality (loose soil, grass) and terrain topography (uphill and downhill), may change over the time, modeling errors are induced [14]. As an artifice, nonlinear MHE (NMHE) is employed to learn the changing operational conditions, so that a better performing NMPC can be realized. Secondly, an in-flight payload dropping application of a tilt-rotor tricopter UAV is addressed. With each drop of payload, the total UAV mass varies and this results in

a plant-model mismatch. Therefore, in order to eliminate this mismatch and hence, achieve a superior tracking accuracy from NMPC, NMHE is utilized to learn the UAV mass online. For both the applications, fast NMPC and NMHE solution methods are incorporated and the test-results are obtained from real-time experiments.

The remaining part of this study is organized as follows: Section 2 illustrates the receding horizon control and estimation methods in terms of NMPC and NMHE problem formulations. In Section 3, the leaning-based NMPC-NMHE framework is demonstrated for the tracking problems of two robotic systems. Finally, the drawn conclusions are presented in Section 4.

2 Receding Horizon Control and Estimation Methods

In this section, we briefly discuss the optimal control problems (OCPs) of NMPC and NMHE. For both the OCPs, the considered nonlinear system is modelled as:

$$\dot{\mathtt{x}}(t) = \mathtt{f}\big(\mathtt{x}(t), \mathtt{u}(t), \mathtt{p}\big), \tag{1}$$

where $\mathtt{x}(t) \in \mathbb{R}^{n_\mathtt{x}}$, $\mathtt{u}(t) \in \mathbb{R}^{n_\mathtt{u}}$ and $\mathtt{p}(t) \in \mathbb{R}^{n_\mathtt{p}}$ are the state, input, and system parameter vectors, respectively, at time t; $\mathtt{f}(\cdot,\cdot,\cdot) : \mathbb{R}^{n_\mathtt{x}+n_\mathtt{u}+n_\mathtt{p}} \longrightarrow \mathbb{R}^{n_\mathtt{x}}$ is the continuously differentiable state update function and $\mathtt{f}(0,0,\mathtt{p}) = 0\ \forall t$. The derivative of $\mathtt{x}$ with respect to t is denoted by $\dot{\mathtt{x}} \in \mathbb{R}^{n_\mathtt{x}}$.

Similarly, a nonlinear measurement model denoted as $\mathtt{y}(t)$ can be described with the following equation:

$$\mathtt{y}(t) = \mathtt{h}\big(\mathtt{x}(t), \mathtt{u}(t), \mathtt{p}\big), \tag{2}$$

where $\mathtt{h}(\cdot,\cdot,\cdot) : \mathbb{R}^{n_\mathtt{x}+n_\mathtt{u}+n_\mathtt{p}} \longrightarrow \mathbb{R}^{n_\mathtt{x}}$ is the measurement function which describes the relation between the variables of the system model and the measured outputs of the real-time system.

2.1 Nonlinear Model Predictive Control

NMPC is an advanced, dynamic optimization-based strategy for feedback control that solely relies on the accuracy of the mathematical model for its optimum performance. In NMPC strategy, a parametric OCP is formulated in the form of a least square function, in order to penalize deviations of predicted system's trajectory (including states and control inputs) from the specified reference. The parametric nature of the OCP is due to its dependence on the current state (measured or estimated). In addition, to keep the computational burden realizable for a real-time application (especially for fast robotic systems), the optimization problem is solved over a finite window, commonly known as *prediction horizon* (N_c). It may be worth noting that NMPC typically leads to non-convex optimization problems, in contrast to linear MPC in which nearly all formulations use convex cost and constraint functions [24].

In NMPC, the dynamic optimization problem is recursively solved for the optimal control inputs, over the given prediction horizon ($t_j \leq t \leq t_{j+N_c}$) at each sampling instant. We formulate the following least-square type cost function in discrete time, which is commonly utilized for tracking applications:

$$\min_{\mathtt{x}_k, \mathtt{u}_k} \quad \frac{1}{2}\left\{ \sum_{k=j}^{j+N_c-1} \left(\left\| \mathtt{x}_k - \mathtt{x}_k^{\mathrm{ref}} \right\|_{W_\mathtt{x}}^2 + \left\| \mathtt{u}_k - \mathtt{u}_k^{\mathrm{ref}} \right\|_{W_\mathtt{u}}^2 \right) + \left\| \mathtt{x}_{N_c} - \mathtt{x}_{N_c}^{\mathrm{ref}} \right\|_{W_{N_c}}^2 \right\} \tag{3a}$$

$$\text{s.t.} \quad \mathtt{x}_j = \hat{\mathtt{x}}_j, \tag{3b}$$

$$\mathtt{x}_{k+1} = \mathtt{f_d}(\mathtt{x}_k, \mathtt{u}_k, \mathtt{p}), \qquad k = j, \cdots, j+N_c-1, \tag{3c}$$

$$\mathtt{x}_{k,\min} \leq \mathtt{x}_k \leq \mathtt{x}_{k,\max}, \qquad k = j, \cdots, j+N_c, \tag{3d}$$

$$\mathtt{u}_{k,\min} \leq \mathtt{u}_k \leq \mathtt{u}_{k,\max}, \qquad k = j, \cdots, j+N_c-1, \tag{3e}$$

where $\mathtt{x}_k \in \mathbb{R}^{n_\mathtt{x}}$ is the differential state, $\mathtt{u}_k \in \mathbb{R}^{n_\mathtt{u}}$ is the control input and $\hat{\mathtt{x}}_j \in \mathbb{R}^{n_\mathtt{x}}$ is the current state estimate; time-varying state and control references are denoted by $\mathtt{x}_k^{\mathrm{ref}}$ and $\mathtt{u}_k^{\mathrm{ref}}$, respectively; the terminal state reference is denoted by $\mathtt{x}_{N_c}^{\mathrm{ref}}$; the discrete time dynamical model is represented by $\mathtt{f_d}(\cdot,\cdot,\cdot)$; $W_\mathtt{x} \in \mathbb{R}^{n_\mathtt{x} \times n_\mathtt{x}}$, $W_\mathtt{u} \in \mathbb{R}^{n_\mathtt{u} \times n_\mathtt{u}}$ and $W_{N_c} \in \mathbb{R}^{n_\mathtt{x} \times n_\mathtt{x}}$ are the corresponding weight matrices, which are assumed constant for simplicity, however, their time-varying formulation can also be included in a similar manner. Furthermore, $\mathtt{x}_{k,\min} \leq \mathtt{x}_{k,\max} \in \mathbb{R}^{n_\mathtt{x}}$ and $\mathtt{u}_{k,\min} \leq \mathtt{u}_{k,\max} \in \mathbb{R}^{n_\mathtt{u}}$ specify the lower and upper bounds on the states and control inputs, respectively.

Once the solution to the OCP (3) at t_j is available, the first computed control input ($\mathtt{u}_j$) is applied to the system for a short time period, that typically coincides with the sampling time [19]. This sampling time has to be kept short enough with respect to the system's dynamics, while sufficiently long at the same time to facilitate timely computation of the optimized solution. Subsequently, a new optimization problem is solved for the prediction window $[t_{j+1}, t_{j+N_c+1}]$, which itself is moving forward with time. Due to this shifting property of the prediction window, the NMPC is also known as *receding horizon* control technique.

The last expression in (3a) represents the final cost incurred due to the finite prediction horizon and is generally referred to as the *terminal penalty* term. This term is often included in the problem formulation for stability reasons [19]. In addition, some other stability results include a problem formulation with sufficiently long horizon [10], an additional prediction horizon and a locally stabilizing control law [25].

2.2 Nonlinear Moving Horizon Estimation

Typically, MHE is considered as a *dual problem* of MPC as they exploit the similar optimization problem structure; despite the fact that MPC predicts the future of the system, while MHE utilizes the past measurements over an *estimation horizon* for state estimation [20, 31]. Moreover, the two main differences of optimization problem formulation of MHE from MPC are: (i) there is no initial state constraint

like in (3b), and (ii) the optimization variables are the states and unknown system parameters, excluding the control inputs as they are already given to the system in the past.

In a similar manner to NMPC, the NMHE scheme is also formulated using a least square function to penalize the deviation of estimated outputs $\big(\mathtt{h}(\cdot,\cdot,\cdot)\big)$ from measurements ($\mathtt{z}$). The performance of NMHE also relies on the availability of an accurate system model, while a mismatch in the form of process noise between the system model and the real plant may deteriorate the optimal estimation solution, which eventually may lead to an unstable closed-loop. To address this issue, a suitable component (*arrival cost*) is included in the final optimization problem formulation of NMHE, as done in [20]. The NMHE formulation includes an estimation horizon containing M measurements $(\mathtt{z}_{\mathtt{S}},\cdots,\mathtt{z}_j,)$ taken at time $t_{\mathtt{S}}<\cdots<t_j$, where the length of the horizon is given by $T_E=t_j-t_{\mathtt{S}}$, and $j-M+1\overset{\text{def}}{=}\mathtt{S}$ is taken for notational convenience. Finally, the discrete time dynamic optimization problem to estimate the constrained states ($\hat{\mathtt{x}}$) as well as the unknown parameter ($\hat{\mathtt{p}}$) at time t_j using the process model $\mathtt{f}(\cdot,\cdot,\cdot)$, measurement model $\mathtt{h}(\cdot,\cdot,\cdot)$ and available measurements within the horizon, is of the form [20]:

$$\min_{\hat{\mathtt{x}}_k,\hat{\mathtt{p}}} \quad \left\{ \left\| \begin{matrix} \hat{\mathtt{x}}_{\mathtt{S}}-\bar{\mathtt{x}}_{\mathtt{S}} \\ \hat{\mathtt{p}}-\bar{\mathtt{p}}_L \end{matrix} \right\|^2_{P_{\mathtt{S}}} + \sum_{k=\mathtt{S}}^{j} \|\mathtt{z}_k-\mathtt{h}(\hat{\mathtt{x}}_k,\mathtt{u}_k,\mathtt{p})\|^2_V + \sum_{k=\mathtt{S}}^{j-1} \|w_k\|^2_W \right\} \tag{4a}$$

$$\text{s.t.} \quad \hat{\mathtt{x}}_{k+1}=\mathtt{f}_{\mathtt{d}}(\hat{\mathtt{x}}_k,\mathtt{u}_k,\mathtt{p})+w_k, \qquad k=\mathtt{S},\cdots,j-1, \tag{4b}$$

$$\hat{\mathtt{x}}_{k,\min}\le\hat{\mathtt{x}}_k\le\hat{\mathtt{x}}_{k,\max}, \qquad k=\mathtt{S},\cdots,j, \tag{4c}$$

$$\hat{\mathtt{p}}_{\min}\le\hat{\mathtt{p}}\le\hat{\mathtt{p}}_{\max}, \tag{4d}$$

where w_k represents the added process noise; $\hat{\mathtt{x}}_{k,\min}\le\hat{\mathtt{x}}_{k,\max}$ and $\hat{\mathtt{p}}_{\min}\le\hat{\mathtt{p}}_{\max}$ specify the lower and upper bounds on the estimated state and parameter vectors, respectively; $\bar{\mathtt{x}}_{\mathtt{S}}$ and $\bar{\mathtt{p}}_{\mathtt{S}}$ denote the estimated state and parameter values (arrival cost data) at the start of estimation horizon, i.e., at $t_{\mathtt{S}}$. The weight matrices $P_{\mathtt{S}}$, V, and W are interpreted as the inverse of the covariance matrices and are evaluated as:

$$P_{\mathtt{S}}=Q_0^{-\frac{1}{2}}=\begin{bmatrix} Q_0^{\mathtt{x}} & 0 \\ 0 & Q_0^{\mathtt{p}} \end{bmatrix}^{-\frac{1}{2}}, \quad V=R^{-\frac{1}{2}}, \quad W=Q^{-\frac{1}{2}}=\begin{bmatrix} Q^{\mathtt{x}} & 0 \\ 0 & Q^{\mathtt{p}} \end{bmatrix}^{-\frac{1}{2}}, \tag{5}$$

where Q_0 is the initial covariance matrix (incorporating state and parameter, both), R is the measurement noise covariance matrix and Q is the process noise covariance matrix. With the above choice of weight matrices, it is assured that the NMHE scheme results in a maximum-likelihood estimate for the very likely trajectories [31].

The first term in (4a) is generally referred to as the arrival cost. It is incorporated into the objective function in order to accommodate the effect of past measurements (before the beginning of estimation horizon), in the current state and parameter estimates. This can be interpreted as analogous to terminal penalty term of NMPC which summarizes the response of the system after the prediction horizon. EKF is often utilized to update the arrival cost for practical implementation, as also done in [20].

Another parameter that affects the performance of NMHE is the choice of estimation window length M, which in general is problem-specific. It basically represents a trade-off between computational liability and estimation accuracy that simultaneously grow with M. In the case of small but fast robotic systems, like ground robots and UAVs, we cannot indefinitely increase M as limited computation power is available on-board. Moreover, it is not necessarily true that the estimation accuracy always increases with M, as the plant-model mismatch degrades the significance of model prediction which adversely affects the estimation performance [19]. That is, the selection of a too high value of M for the system in which the unknown parameter (to be estimated) is radically changing, plant-model anomalies may arise that eventually may result in deteriorated overall estimation quality.

3 Real-Time Applications

In this section, two real-time robotic applications will be presented to show how we have addressed the two main problems encountered in NMPC application, which are lack of modeling and online solution of the nonlinear optimization problem. The applications include the trajectory tracking problems of the ground and aerial robotic systems with time-varying dynamic model parameters which are estimated using NMHE. Owing to the similarities between the optimization problems of NMPC and NMHE defined in (3) and (4), respectively, we solve them utilizing the direct multiple shooting method and real-time iteration approach, which is incorporated in ACADO toolkit [2]. In ACADO toolkit, firstly the optimization problem, in terms of system equations and constraints, is defined in a C++ environment and then, the self-contained C codes are obtained using its code generation package [2]. Finally, these generated C codes can be utilized to run on C/C++ or MATLAB/Simulink based software platforms.

3.1 Ultra-Compact Field Robot

Firstly, we illustrate NMPC for the trajectory tracking problem of a 3D printed field robot, operating in an off-road terrain. Since the soil conditions and terrain topography may vary over the operation, modeling uncertainties would arise. In order to tackle these operational uncertainties and hence, achieve optimum control performance, NMHE is utilized to estimate two slip (or traction) parameters, namely, (α, κ), in addition to performing the state estimation task.

3.1.1 System Description

The 3D printed field robot as shown in Figure 1 has been built utilizing practical, hands-on experience with various sensors and actuators. A real-time kinematic (RTK) differential global navigation satellite system (GNSS), i.e., a Septentrio Altus

(a) Field robot in corn plots (b) CAD drawing of the field robot

Fig. 1: Ulta-compact 3D printed field robot.

APS-NR2 GNNS receiver (Septentrio Satellite Navigation NV, Belgium), is used to obtain highly accurate positional information, which has a specified position accuracy of 0.03 m at a 5-Hz measurements rate. The Trimble network supplies real-time kinematic correction signals via 4G internet. A gyroscope (PmodGYRO with an ST L3G4200D, Digilent Inc., USA) is mounted on the body of the robot to measure the yaw rate of the 3D printed field robot at a rate of 5-Hz with a resolution of 1°. Four powerful 12V brushed DC motors with 131 : 1 metal gearboxes (Pololu Corporation, USA) are used as actuators, and four integrated quadrature encoders for brushed DC motors (Pololu Corporation, USA) are used to measure the speed of the wheels of the field robot with an accuracy of 0.05 m/s.

The real-time NMHE and NMPC are implemented and executed on an on-board computer, i.e., Raspberry Pi 3, which is equipped with Quad Core 1.2 GHz Broadcom BCM2837 64bit CPU and 1 GB of RAM. The inputs of NMHE are the position, speed and yaw rate, while the outputs are full state and parameter vectors that are fed to the NMPC. In addition to the full state and parameter information, NMPC receives the reference trajectory throughout the prediction horizon and then generates a control signal, i.e., the desired yaw rate, and sends it to the low-level controller, i.e., Kangaroo x2 motion controller (Dimension Engineering, USA). Apart from the desired yaw rate, the low-level controller receives the measured speed information from encoders and generates voltage values which are sent to the motor driver (Sabertooth dual 12A motor driver, Dimension Engineering, USA) to control the speeds of the DC motors. The low-level controller is executed at a rate of 50-Hz, which is 10 times more than the high-level controller.

3.1.2 System Model

In this section, we represent the nonlinear system and measurement models of the field robot according to (1) and (2), respectively. Instead of using the traditional

kinematic model of a mobile robot, an adaptive nonlinear kinematic model, which is an extension of the traditional model, is derived as the system model of the field robot in this study. Two traction parameters (α, κ) are added to minimize deviations between the real-time system and system model. These parameters, i.e., α and κ, represent the effective speed and steering of the field robot, respectively. It is noted that they must be between zero and one, and it is inherently arduous to measure them. The field robot's model can be formulated with the following equations:

$$\dot{x} = \alpha v \cos\psi, \tag{6a}$$

$$\dot{y} = \alpha v \sin\psi, \tag{6b}$$

$$\dot{\psi} = \kappa r, \tag{6c}$$

where x and y denote the position of the field robot, ψ denotes the yaw angle, v denotes the speed and r denotes the yaw rate. The state, parameter, input and measurement vectors are, respectively, denoted as follows:

$$\mathrm{x} = \begin{bmatrix} x & y & \psi \end{bmatrix}^T, \tag{7}$$

$$\mathrm{p} = \begin{bmatrix} v & \alpha & \kappa \end{bmatrix}^T, \tag{8}$$

$$\mathrm{u} = r, \tag{9}$$

$$\mathrm{z} = \begin{bmatrix} x & y & v & r \end{bmatrix}^T. \tag{10}$$

3.1.3 Control Scheme

The control objective is to design NMPC in order to track a predefined trajectory. The optimized solution as the desired set point is forwarded to the low-level controller, which is a proportional-integral-derivative (PID) controller. The response of this low-level PID controller is finally given to the motors of the field robot.

3.1.4 Implementation of NMHE

The inputs of NMHE are the position, speed and yaw rate of the field robot as defined in (10). The outputs of NMHE, the position, yaw angle, speed and traction parameters, are the full state and parameter vectors (7)-(8). The NMPC requires full state and parameter as input to generate the desired yaw rate applied to the field robot; therefore, the full estimated state and parameter values by NMHE are fed to NMPC.

The NMHE formulation is solved at each sampling instant with the following constraints on the traction parameters:

$$0 \leq \alpha \leq 1, \tag{11a}$$

$$0 \leq \kappa \leq 1. \tag{11b}$$

The standard deviations of the measurements are set to $\sigma_x = \sigma_y = 0.03\,\text{m}$, $\sigma_v = 0.05\,\text{m/s}$, $\sigma_r = 0.0175\,\text{rad/s}$, based on the experimental analysis. Therefore, the following weighting matrices V, P_S and W are used in NMHE design:

$$\begin{aligned} V &= \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_v^2, \sigma_r^2)^{-1/2}, \\ &= \text{diag}(0.03^2, 0.03^2, 0.5^2, 0.0175^2)^{-1/2}, \end{aligned} \tag{12a}$$

$$\begin{aligned} P_\text{S} = W &= \text{diag}(x^2, y^2, \psi^2, v^2, \alpha^2, \kappa^2)^{-1/2}, \\ &= \text{diag}(10.0^2, 10.0^2, 0.1^2, 1.0^2, 0.25^{2/2}, 0.25^2)^{-1/2}. \end{aligned} \tag{12b}$$

3.1.5 Implementation of NMPC

The NMPC formulation is solved at every sampling instant with the following constraints on the input:

$$-0.1(\text{rad/s}) \leq r \leq 0.1(\text{rad/s}). \tag{13}$$

The state and input references for the field robot are changed online and defined as follows:

$$\mathrm{x}_r = [x_r, y_r, \psi_r]^T \quad \text{and} \quad \mathrm{u}_r = r_r, \tag{14}$$

where x_r and y_r are the position references, r_r is the yaw rate reference, and the yaw angle reference is calculated from the position references as:

$$\psi_r = \text{atan2}\,(\dot{y}_r, \dot{x}_r) + \lambda\,\pi, \tag{15}$$

where λ describes the desired direction of the field robot ($\lambda = 0$ for forward and $\lambda = 1$ for backward). If the yaw rate reference, calculated from the reference trajectory, is used as the input reference, steady state error might occur in case of a mismatch between the system model and the real system. Therefore, the measured yaw rate is used as the input reference to penalize the input rate in the objective function.

The weighting matrices W_x, W_u and W_{N_c} are selected as follows:

$$W_\mathrm{x} = \text{diag}(1,1,1), \quad W_\mathrm{u} = 10 \quad \text{and} \quad W_{N_c} = 10 \times W_\mathrm{x}. \tag{16}$$

The weighting matrix for the input W_u is set larger than the weighting matrix for the states W_x, in order to ensure a well-damped closed-loop system behaviour. In addition, the weighting matrix for the terminal penalty W_{N_c} is set 10 times larger than the weighting matrix for the states W_x. This implies that the last deviations between the predicted states and their references in the prediction horizon are minimized in the objective function 10 times more than the previous points in the prediction horizon. The reason for doing that is the error at the end of the prediction horizon plays a critical role in terms of the stability of the control algorithm.

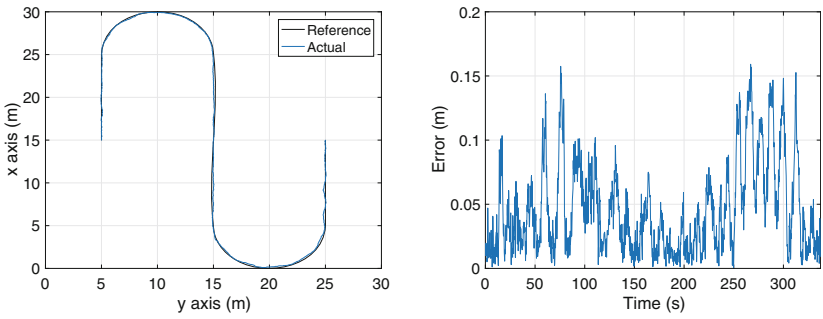
If the prediction horizon is large, the computation burden for NMPC increases unreasonably, such that solving a non-convex optimization problem online will be

infeasible. Moreover, if the prediction horizon is selected to be too small, NMPC cannot stabilize the system. Therefore, the prediction horizon of the NMPC has to be large enough in reference to the velocity of the vehicle, in order to obtain a stable control performance. Since the field robot is a quite slow system, it is not required to select a very large value for the prediction horizon. Thus, it is set to 3 seconds.

3.1.6 Results

Throughout the real-time experiments, a reference trajectory consisting of straight and curved lines is tracked by the 3D printed robot, which is controlled employing the NMPC-NMHE framework. Thus, the performance of the framework can be investigated for different path geometries. The system has a constant speed, and yaw rate is the input to the system. The closest point on the reference trajectory to the 3D printed robot is calculated and then, the next 15 points are fed to the NMPC as reference trajectory due to the fact that the length of the prediction horizon (N_c) is set to 15.

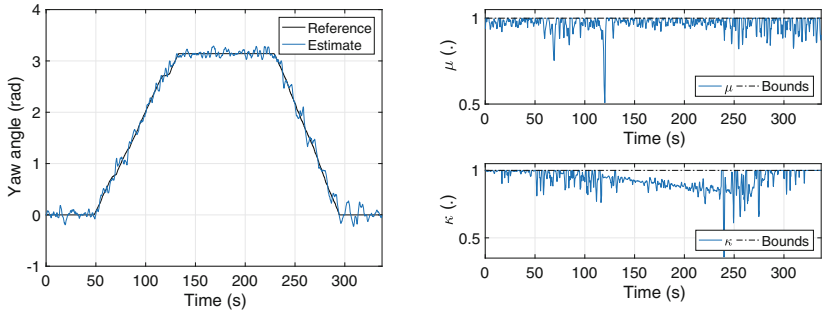
The control performance of the 3D printed robot is shown in Figure 2. As can be seen in Figure 2a, the robot is capable of staying on-track throughout the experiment and tracking the target trajectory accurately. The variation of Euclidean error with time is shown in Figure 2b and its mean value is approximately 0.0459 m, which is within the tolerance for an agricultural application.



(a) Reference and actual trajectories (b) Euclidean error

Fig. 2: Trajectory tracking control performance.

The performance of NMHE in estimating the yaw angle and traction parameters is shown in Figure 3. NMPC needs full state information to generate a control signal. The position in x- and y-axes is measured; however, the yaw angle cannot be measured in practice. Therefore, NMHE estimates the yaw angle, which plays a very important role in the trajectory tracking performance. As seen in Figure 3a, the yaw angle has been controlled very accurately. Moreover, the traction parameters are immeasurable and the constraints on these parameters are defined in (11). It is

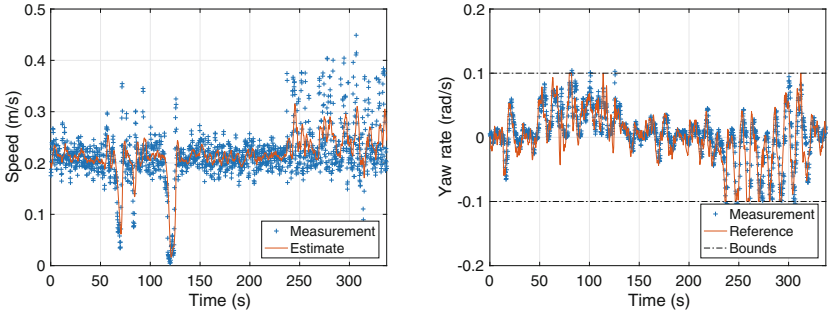


(a) Reference and estimated yaw angle (b) Traction parameters

Fig. 3: Estimation performance of NMHE.

important to estimate the traction parameters, because soil conditions can change over the time. Therefore, the online estimation of the parameters is required to learn soil conditions and thus, adapt NMPC to the changing working conditions. As can be seen in Figure 3b, the estimated values are within the bounds. Moreover, it is observed that the traction parameter estimates stabilize at certain values, so that a stable trajectory tracking performance is ensured.

The measured and estimated speed of the 3D printed robot is shown in Figure 4a. NMHE is capable of filtering noisy measurements. Additionally, the control signal,



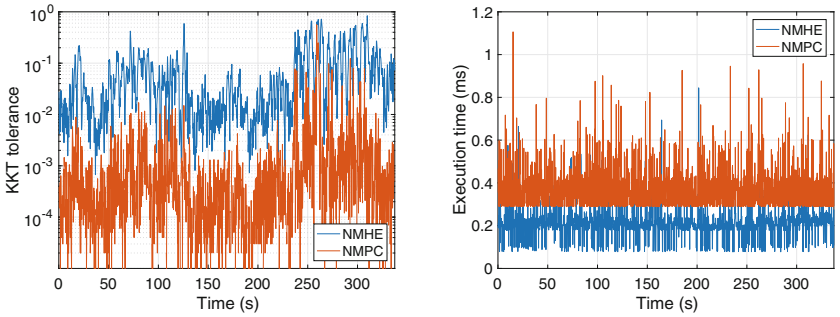
(a) Measured and estimated speed (b) Reference and actual yaw rate

Fig. 4: Speed and yaw rate.

i.e., yaw rate reference, generated by the NMPC is shown in Figure 4b. It is observed that the NMPC is capable of dealing with the input constraints and the low-level controller shows a good control performance.

It is necessary to check the optimality of the NMPC-NMHE framework, because a single quadratic programming iteration at each sampling time instant may result in a suboptimal solution. Therefore, the Karush-Kuhn-Tucker (KKT) tolerances for

NMHE and NMPC are shown in Figure 5a. The KKT tolerances are very small, but they are not equal to zero. The reason is that a quadratic program is solved precisely only for the linear systems, such that the KKT tolerance becomes zero. Moreover, the low and non-drifting KKT tolerances emphasize that the optimization problems in the NMHE and NMPC are well defined and properly scaled. The execution times for the NMHE and NMPC are shown in Figure 5b. Their mean values are 0.2101 ms and 0.3813 ms, respectively, which implies that the overall computation time for the NMPC-NMHE framework is around 0.5914 ms.



(a) KKT Tolerances (b) Execution time

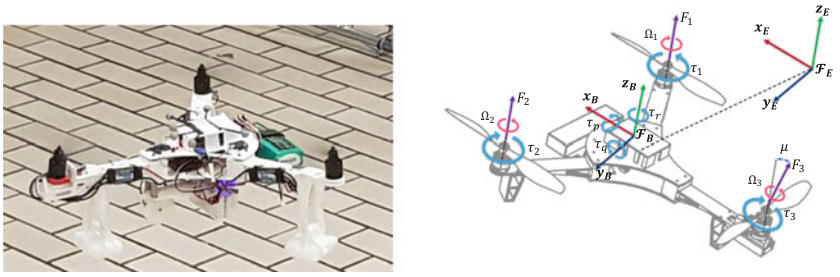
Fig. 5: KKT tolerances and execution times of NMHE and NMPC.

3.2 Tilt-Rotor Tricopter UAV

In this application, we tackle a real-life package delivery problem, where a UAV takes off with the full payload, tracks a predefined trajectory in 3D, drops each package to the time-based designated location, and finally, returns to its starting location with no payload. In this application, the UAV mass is 1.608 kg without any payload. The dropping mechanism is designed to drop four payloads in the sequence 55 g, 75 g, 77 g, and 86 g, respectively, which makes the total takeoff mass to be 1.901 kg. Considering the total payload of 293 g, it is almost 18% of the total mass of the UAV. This means a massive change in the model parameters which has to be handled during the control of the system. In this application, we learn the variations in the mass online and feed the estimated mass value to the model which is used by the NMPC.

3.2.1 System Description

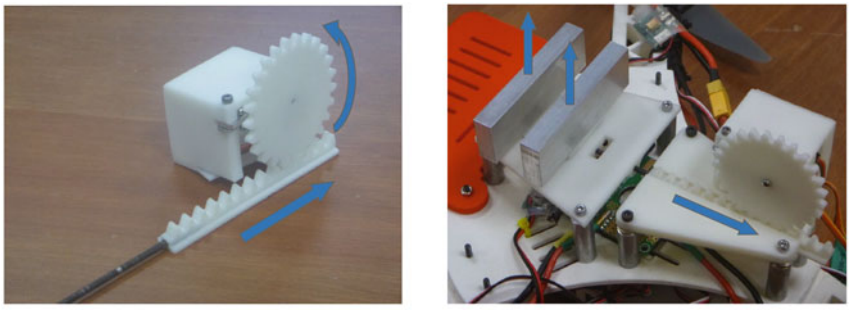
The aerial robot used in this application is a 3D printed tilt-rotor tricopter, as shown in Figure 6a. It is a custom-made system, which is developed based on the other Talon tricopter frames available in the market. The frame is customized, such that it provides flexibility to accommodate all the electronics as needed. The Pixhawk flight controller is used as the low-level stabilization controller. In addition, the tricopter also houses the on-board computer, i.e., Raspberry Pi 3, which serves two vital functions. One is wireless communication with the ground-station computer, and the other is controlling the servomotor for the payload drop mechanism.



(a) Actual setup (b) Coordinate frame and sign conventions

Fig. 6: 3D printed tilt-rotor tricopter UAV.

The mechanism used to hold and drop the payload throughout the flight has two plates, which are supported at the base of the UAV. Amongst them, one houses the servomotor, while the other holds the payload blocks to be dropped. A circular gear mounted on the servomotor drives a linear gear, shown in Figure 7a, that results in a linear motion. This linear motion pulls the rod attached to the linear gear and thus,



(a) Servo-gear mounting (b) Assembly with weights

Fig. 7: Payload drop mechanism.

drops the blocks one-by-one in the process. The dropping mechanism is shown in Figure 7b with an inverted view.

3.2.2 System Model

The tilt-rotor tricopter is considered as a rigid-body having two stationary rotors and one non-stationary (or tilting) rotor, as shown in Figure 6b. In our configuration, the two stationary rotors – RR (right rotor) rotating clockwise and LR (left rotor) rotating counter-clockwise – are placed in the front of the body (CG - centre of gravity), while the tilting rotor – BR (back rotor) rotating counter-clockwise – is mounted at the rear part of the body.

3.2.3 Kinematic Equations

The translational and rotational motion, describing position and orientation of the UAV, are obtained using the transformation from body-fixed frame ($\mathcal{F}_B$) to Earth-fixed frame ($\mathcal{F}_E$). They are written as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_{EB} \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = T_{EB} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{17}$$

where x, y, z and ϕ, θ, ψ are the translational position and rotational attitude, respectively, which are defined in $\mathcal{F}_E$; u, v, w and p, q, r are the translational and rotational velocities that are defined in $\mathcal{F}_B$; R_{EB} is the translation transformation matrix between frames $\mathcal{F}_E$ and $\mathcal{F}_B$, while T_{EB} maps the rotational velocity component from $\mathcal{F}_B$ to $\mathcal{F}_E$. The matrices R_{EB} and T_{EB} are given as ($c : cos$, $s : sin$, $t : tan$):

$$R_{EB} = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - s\psi c\phi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\psi c\phi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix}, \tag{18a}$$

$$T_{EB} = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & \frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix}. \tag{18b}$$

3.2.4 Rigid-Body Equations

The rigid-body dynamic equations of the tilt-rotor tricopter are derived based on the Newton-Euler formulation in the body coordinate system, similar to [3]. Within these equations, the tricopter is assumed to be a point mass, wherein all the forces and moments act at the CG. The corresponding force and moment equations can be written as:

Force Equations

$$\dot{u} = rv - qw + g\sin(\theta) + \frac{1}{m}F_x, \tag{19a}$$

$$\dot{v} = pw - ru - g\sin(\phi)\cos(\theta) + \frac{1}{m}F_y, \tag{19b}$$

$$\dot{w} = qu - pv - g\cos(\phi)\cos(\theta) + \frac{1}{m}F_z, \tag{19c}$$

Moment Equations

$$\dot{p} = \left(\frac{1}{I_{xx}I_{zz} - I_{xz}^2}\right)\Big[\{-pq(I_{xz}) + qr(I_{yy} - I_{zz})\}I_{zz} - \{qr(I_{xz}) + pq(I_{xx} - I_{yy})\}I_{xz} + \tau_x(I_{zz}) - \tau_z(I_{xz})\Big], \tag{20a}$$

$$\dot{q} = pr\left(\frac{I_{zz} - I_{xx}}{I_{yy}}\right) - (r^2 - p^2)\left(\frac{I_{xz}}{I_{yy}}\right) + \tau_y\left(\frac{1}{I_{yy}}\right), \tag{20b}$$

$$\dot{r} = \left(\frac{1}{I_{xx}I_{zz} - I_{xz}^2}\right)\Big[\{qr(I_{xz}) + pq(I_{xx} - I_{yy})\}I_{xx} - \{-pq(I_{xz}) + qr(I_{yy} - I_{zz})\}I_{xx} + \tau_z(I_{xx}) - \tau_x(I_{xz})\Big], \tag{20c}$$

where F_x, F_y, F_z, are the total external forces and τ_x, τ_y, τ_z, are the total external moments acting on the tricopter body in frame $\mathcal{F}_B$. In addition, I_{xx}, I_{yy}, I_{zz} and I_{xz} represent the moments of inertia of the whole tricopter along axes $\mathcal{F}_{B_x}$, $\mathcal{F}_{B_y}$, $\mathcal{F}_{B_z}$ and $\mathcal{F}_{B_{xz}}$, respectively. One may note that unlike a quadrotor UAV, the tilt-rotor tricopter only has a single plane of symmetry, i.e., along $\mathcal{F}_{B_{xz}}$ plane. Therefore, the effect of asymmetric moment I_{xz} is explicitly considered in (20), in contrast to what is done in [3].

3.2.5 External Forces and Moments

The external forces and moments generated by the rotors rotating at a certain angular velocity Ω are modelled as:

$$F_i = K_F\Omega_i^2 \quad \text{and} \quad \tau_i = K_\tau\Omega_i^2, \tag{21}$$

where F_i and τ_i are the external force and drag-moment generated, respectively. Also, K_F and K_τ are positive intrinsic parameters of the rotor and are commonly known as the *force* and *drag-moment* coefficients, respectively. According to the tilt-rotor tricopter configuration shown in Figure 6b, the expression for total external force acting on the tricopter body in $\mathcal{F}_B$ frame is written as:

$$F_{ext} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} 0 \\ -F_3\sin(\mu) \\ F_1 + F_2 + F_3\cos(\mu) \end{bmatrix}, \tag{22}$$

where μ is the tilting angle of the back rotor. On the other hand, the total external moment acting on the tricopter platform is the summation of moment due to propeller's rotation τ_{prop}, and the moment due to change in orientation of propeller's rotation plane τ_{gyro}. The latter is commonly known as gyroscopic moment and can be written as:

$$\tau_{gyro} = \sum_{n=1}^{3} J_P(\mathrm{x}_{rate} \times r_n)\Omega_n, \tag{23}$$

where J_P is propeller's moment of inertia and r_n is the unit reaction vector along the rotational axis of n^{th} rotor and x_{rate} is the angular velocity vector. Finally, the expression for total external moment is:

$$\tau_{ext} = \tau_{prop} + \tau_{gyro}, \tag{24}$$

where

$$\tau_{prop} = \begin{bmatrix} (F_2 - F_1)l_2 \\ (F_3 cos(\mu))l_1 - (F_1 + F_2)l_3 + \tau_3 sin(\mu) \\ \tau_1 - \tau_2 - \tau_3 cos(\mu) + (F_3 sin(\mu))l_1 \end{bmatrix}, \tag{25}$$

$$\tau_{gyro} = \begin{bmatrix} J_P\{q(\Omega_1 - \Omega_2) - \Omega_3(\cos(\mu)q + \sin(\mu)r)\} \\ J_P\{p(\Omega_2 - \Omega_1) + \cos(\mu)\Omega_3\} \\ J_P\{-p\sin(\mu)\Omega_3\} \end{bmatrix}. \tag{26}$$

Furthermore, the constant intrinsic parameters for the considered tilt-rotor tricopter UAV are listed in Table 1. These parameters are either obtained by experiments or by any of the system identification method. In this application, we physically measured the mass (m) of the UAV (without the payload mass) and the moment arm lengths (l_1, l_2, l_3). However, for the evaluation of the moment of inertias ($I_{(..)}$) and thrust (K_f) as well as drag-moment (K_τ) coefficients, simple experiments are performed; details of which can be referred from [8, 12].

Table 1: Tilt-rotor tricopter intrinsic parameters

Parameter	Description	Value
m	Mass of tricopter UAV	1.608 kg
l_1	Moment arm	0.284 m
l_2	Moment arm	0.212 m
l_3	Moment arm	0.092 m
I_{xx}	Moment of Inertia about $\mathcal{F}_{B_x}$	0.016053 kg-m^2
I_{yy}	Moment of Inertia about $\mathcal{F}_{B_y}$	0.028158 kg-m^2
I_{zz}	Moment of Inertia about $\mathcal{F}_{B_z}$	0.032752 kg-m^2
I_{xz}	Moment of Inertia about $\mathcal{F}_{B_{xz}}$	0.029763 kg-m^2
K_f	Aerodynamic force coefficient	3.76×10^{-5} N-s^2
K_τ	Aerodynamic drag-moment coefficient	2.56×10^{-6} Nm-s^2

3.2.6 Control Scheme

In contrast to what is done in [27], NMPC in this implementation is designed to be responsible for tracking a given position trajectory. Based upon the current feedback of the other states, optimized solutions for the control inputs in terms of the total thrust and attitude angles are computed. These optimized solutions are then passed to the low-level controller as their desired setpoints. Moreover, the low-level attitude controller is selected as a PID controller (implemented in Pixhawk), which is designed individually for each axis.

3.2.7 Implementation of NMPC

The state, parameter, control and measurement vectors for the high-level NMPC are considered to be composed of:

$$\mathrm{x}_{\mathrm{NMPC}} = [x, y, z, u, v, w]^T, \tag{27}$$

$$\mathrm{p}_{\mathrm{NMPC}} = m, \tag{28}$$

$$\mathrm{u}_{\mathrm{NMPC}} = [F_z, \phi, \theta, \psi]^T, \tag{29}$$

$$\mathrm{z}_{\mathrm{NMPC}} = [x, y, z, u, v, w]^T. \tag{30}$$

Additionally, the final nonlinear programming (NLP) formulation for high-level NMPC also requires the parametrization of the nonlinear model (in translation) with respect to the three rotational rates namely, p, q and r. Therefore, to obtain the solution of the formulated NLP, the three rotational rates are fed to the NMPC along with the other states at each sampling instant. Furthermore, the following state and control nominal values are selected for the parametrization of the state and control trajectories:

$$\mathrm{x}^{\mathrm{ref}} = \mathrm{x}_{N_c}^{\mathrm{ref}} = [x_r, y_r, z_r, 0, 0, 0]^T, \quad \text{and} \quad \mathrm{u}^{\mathrm{ref}} = [\mathrm{mg}, -0.0414, 0, 0]^T, \tag{31}$$

where m and g are the UAV mass and gravitational constant, respectively.

Some constraints are introduced in the definition of NMPC due to the restrictions put up by the real setup. Typically, these are the input constraints that are imposed in order to achieve a stable behaviour from the low-level controller:

$$0.5mg\ (\mathrm{N}) \leq F_z \leq 1.5mg\ (\mathrm{N}), \tag{32a}$$

$$-15\ (^\circ) \leq \phi \leq 15\ (^\circ), \tag{32b}$$

$$-15\ (^\circ) \leq \theta \leq 15\ (^\circ). \tag{32c}$$

Also, the following weight matrices are selected by trial-and-error:

$$W_{\mathrm{x}} = \mathrm{diag}(25, 26, 32, 1.0, 1.0, 1.1), \tag{33a}$$

$$W_{\mathrm{u}} = \mathrm{diag}(0.024, 22, 25, 80), \tag{33b}$$

$$W_{N_c} = \mathrm{diag}(40, 40, 40, 1, 1, 1). \tag{33c}$$

Furthermore, the prediction window $N_c = 30$ is selected to facilitate the real-time applicability of the control framework. One may note that for defining the constraints and obtaining NMPC weights, the UAV mass is selected to be the maximum takeoff mass, i.e., $m = 1.901$ kg.

3.2.8 Implementation of NMHE

In this application, the main task of NMHE is to estimate the UAV mass (m) online, which is made time-varying by a sequential drops of payload. The overall state, parameter, control and measurement vectors for NMHE design are considered to be composed of:

$$\mathrm{x}_{\mathrm{NMHE}} = [u, v, w]^T, \tag{34}$$

$$\mathrm{p}_{\mathrm{NMHE}} = m, \tag{35}$$

$$\mathrm{u}_{\mathrm{NMHE}} = [F_z, \phi, \theta]^T, \tag{36}$$

$$\mathrm{z}_{\mathrm{NMHE}} = [u, v, w, F_z, \phi, \theta]^T. \tag{37}$$

One may note that the state vector for NMHE in (34) is different than the state vector for NMPC in (27). This is because m only appears in the force equations of (19). Moreover, the three rotational rates are included in the measurements along with states and inputs in order to solve the underlying NLP, as also done in NMPC.

For the selected tricopter model, the weight matrices P_{S}, V and W are chosen to be:

$$P_{\mathrm{S}} = \mathrm{diag}(5.4772^2, 5.4772^2, 5.4772^2, 8.9443^2)^{-1/2}, \tag{38a}$$

$$V = \mathrm{diag}(0.0447^2, 0.0447^2, 0.0447^2, 0.2236^2, 0.1^2, 0.1^2)^{-1/2}, \tag{38b}$$

$$\mathrm{W} = \mathrm{diag}(0.01^2, 0.0316^2, 0.0316^2, 0.0316^2)^{-1/2}. \tag{38c}$$

The above values of the weight matrices are decided based upon experience, incorporating the definitions in (5). Additionally, in order to achieve a constrained estimation of the UAV mass, the initial knowledge about the maximum takeoff mass and the minimum assembly mass is exploited and hence, the following constraints are imposed:

$$1.5 \text{ (kg)} \leq m \leq 2.0 \text{ (kg)}. \tag{39}$$

Furthermore, the estimation window length M is selected to be equal to 70, which is more than the prediction horizon length of 30 for NMPC. This is purposely kept in order to realize a slower learning from NMHE.

3.2.9 Results

In this section, we present the results of the implementation of NMPC for the high-level position tracking of a tilt-rotor tricopter UAV. In addition to tracking, we also analyse its robustness for the time-varying dynamics of the system by conducting experiments for two scenarios: *NMPC without learning*, and *NMPC with learning* (also referred to as NMPC-NMHE framework).

The real-time implementation of the entire process is summarized in Figure 8. The NMPC and NMHE, which are running at 50-Hz and 30-Hz, respectively, are designed using 's-functions' in Simulink, which are generated via the ACADO toolkit. The OptiTrack motion capture system, consisting of eight cameras running at 240 frames/sec, is used to get the feedback during experiments. The controller commands along with the feedback (position and orientation) are sent to the UAV via Raspberry Pi over a wireless network. The low-level controller is running on the Pixhawk module, which takes the NMPC commands from Raspberry Pi 3 and further computes the actuator commands that are finally given to the tricopter motors. The communication between the UAV and ground-station is achieved through `ROS` running on a hardware consisting of Intel® Core™i7-4710MQ CPU@2.50GHz processor with 15.6 GB of memory on a 64-bit Ubuntu Linux system.

The initial state of the UAV is $\mathrm{x}(0) = [0, 0, 1.5, 0, 0, 0]^T$. In addition to the discretization of the nonlinear tricopter model utilizing multiple shooting method (implemented in ACADO toolkit), a fixed integration process consisting of $2N_c$ steps is also performed based on *Runge-Kutta* 4^{th} order method. Moreover, a time-based circular reference trajectory ($[x_r, y_r, 1.5]$) of radius 1 m is selected for these experiments. In both the scenarios, first a complete circle is performed with full takeoff mass of the UAV and then, the payload in terms of four blocks are sequentially dropped (55g $\rightarrow$ 75g $\rightarrow$ 77g $\rightarrow$ 86g) at fixed time-intervals during the trajectory.

Remark I: In the two scenarios – NMPC without learning and NMPC with learning – analysed here, NMHE in the latter case is only utilized to perform the parameter estimation, i.e., only the estimation of mass (m) is fed to NMPC, not the state values. This is done in order to achieve a consistent comparison between the two cases.

3.2.10 Circular Reference Tracking

The overall position tracking performance of the UAV for both scenarios can be seen in Figure 9a and b, where the vertical magenta lines represent the instants of payload drop. In Figure 9b, one may notice a negative offset along z since the beginning for the case of NMPC without leaning. The reason for this is the slight mismatch that exists between the model and the system. Nevertheless, the offset is around 5 cm, which is reasonable compared to the size of the UAV. Moreover, it is illustrated from Figure 9a and b that the position tracking of the NMPC-NMHE framework is better than the NMPC without learning case. The controller's performance is stable and especially, the tracking along z is more accurate. This is because NMHE is able

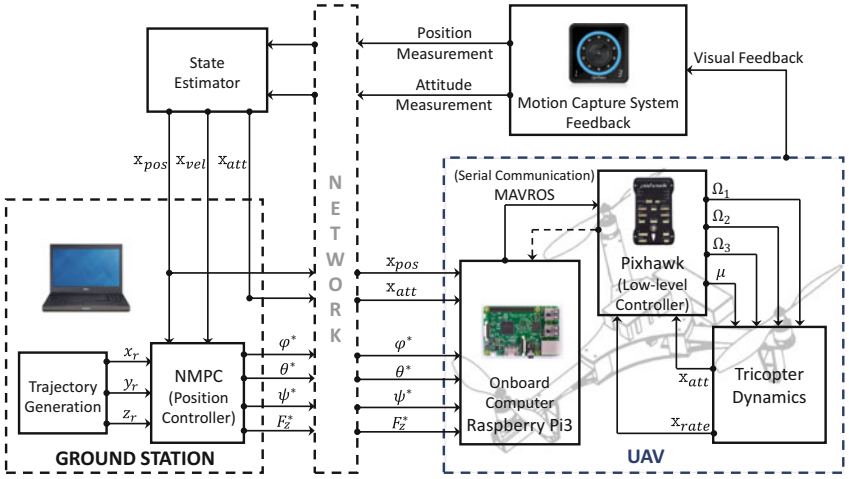
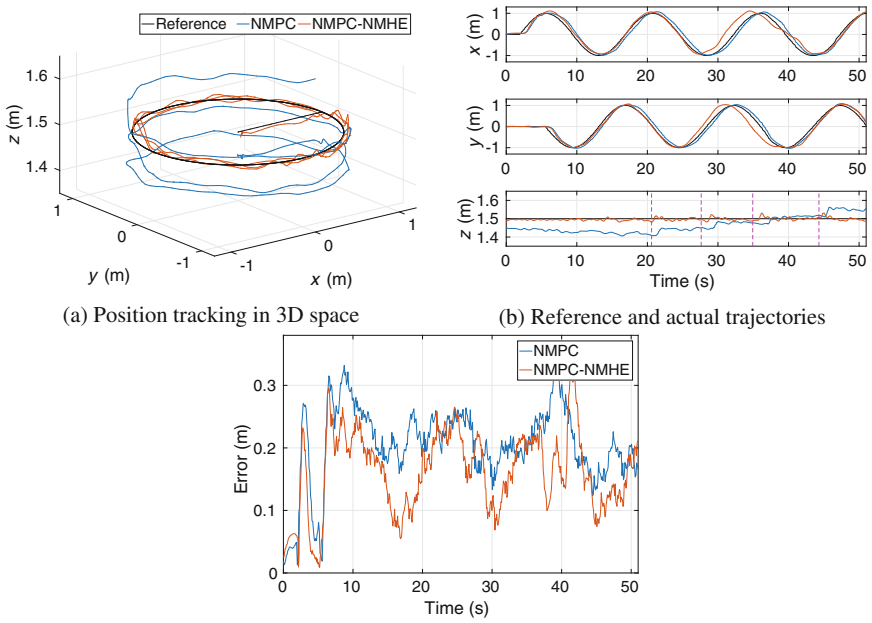


Fig. 8: Schematic diagram for real-time implementation.

to learn the change in UAV mass and thus, the offset created due to mass change diminishes with time. Also, the Euclidean errors for position tracking in the two cases are shown in Figure 9c, where their mean value over the entire run time for NMPC without and with learning are 0.2064 m and 0.1618 m, respectively.

The performance of NMPC can be appreciated by observing its F_z command throughout the trajectory for both without and with learning cases, as shown in Figure 10a, where the vertical magenta lines again represent the instants of payload drop. It is implied that for both the scenarios, the F_z command of NMPC never crosses the bounds specified in (32a), but gets adjusted at every instant of a payload drop. Additionally, the attitude angles commanded by NMPC for without and with learning are given in Figure 10b and c, respectively. It can be seen that ϕ and θ angles of the UAV are well within the specified bounds defined in (32b) and (32c), respectively, while ψ response has some irregularities following the heading command of NMPC. This behaviour is due to the PID tuning of the Pixhawk's low-level controller, but the system is stable enough to maintain the heading.

The performance of NMHE in estimating the mass is shown in Figure 10d, along with its true value, wherein the estimation is observed to stay within the specified bounds defined in (39). As NMPC is a model-based controller, a good estimate of the time-varying model parameters is crucial for its optimum performance. Moreover, in package delivery applications of the UAV that we are considering, the total mass changes with time. Therefore, it is important to estimate it, so that NMPC can adapt itself to the changing working conditions. It is worth noting that although m is a physical parameter in the system model, when it is estimated by NMHE, it also accommodates the effects of modeling uncertainties that are injected during operation. Overall, it can be interpreted as an adaptive parameter that facilitates NMPC to achieve an offset free tracking along z.



(a) Position tracking in 3D space

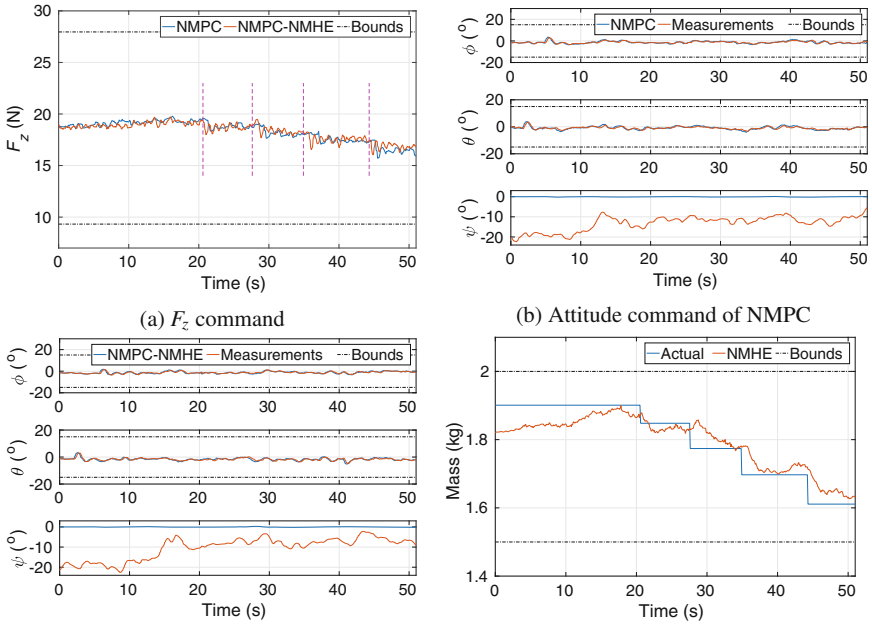
(b) Reference and actual trajectories

(c) Euclidean error

Fig. 9: Trajectory tracking performance.

In order to check the optimality of NMPC and NMPC-NMHE framework, their KKT tolerances are obtained and plotted in Figure 11a and b, respectively. As mentioned in the previous application, it is necessary to check the optimality of the solution because a single quadratic programming iteration at each sampling time instant (performed in ACADO) may result in a suboptimal solution. As visualized in Figure 11a and b, the KKT tolerances for NMPC and NMHE are small, but not zero. This is happening due to the nonlinearities in the system dynamics. It is to be noted that the tilt-rotor tricopter UAV is far more nonlinear and inherently unstable in comparison to other multirotor UAVs including quadrotors and hexacopters. This is mainly due to the odd number of rotors, which results in an unbalanced yaw moment. In addition, one may point out the difference in KKT values for NMPC and NMHE, and the reason lies in the selection of their prediction and estimation horizons, respectively. Nonetheless, their low and non-drifting magnitudes still represent the well-defined and properly scaled optimization problems of both NMPC without learning and the NMPC-NMHE framework.

Finally, the execution times for each entity in NMPC without learning and NMPC-NMHE framework are displayed in Figure 12a and b, respectively. In addition, the combined mean execution times for both without and with learning cases are 0.7763 ms and 3.7 ms, respectively. These values are less compared to the selected sampling time of 20 ms (position controller) and hence, support the implementation of both on a cheaper embedded hardware including Raspberry Pi 3.



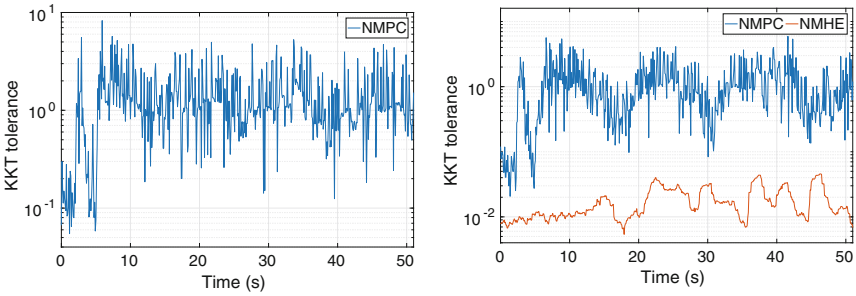
(a) F_z command

(b) Attitude command of NMPC

(c) Attitude command of NMPC with learning

(d) Estimation performance of NMHE

Fig. 10: Controller and estimator outputs.

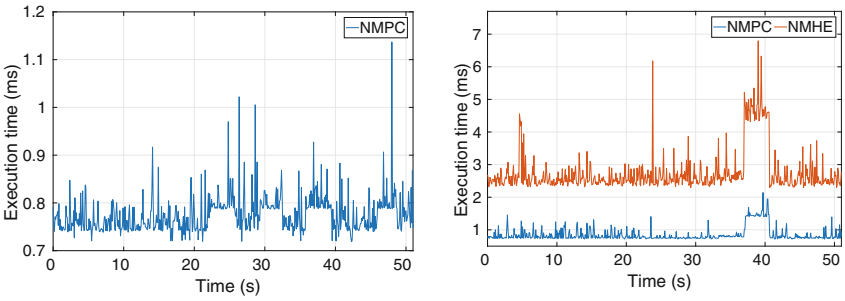


(a) NMPC without learning

(b) NMPC-NMHE framework

Fig. 11: KKT tolerance.

Remark II: The tuning of NMPC weights is a problem that is generally encountered while performing the experiments. Any minor change in the system including discharging of the battery leads to a change in the controller weights mainly along F_z. Moreover, the best way to tune the combined NMPC-NMHE framework is to first tune the NMPC separately, and then utilize those weights as a reference for the combined framework.



(a) NMPC without learning

(b) NMPC-NMHE framework

Fig. 12: Execution time.

Remark III: Selecting an appropriate estimation horizon for NMHE is problem specific as it directly affects the rate of learning. That is, for a shorter horizon, the mass learning is fast which eventually makes NMPC to be aggressive towards the change. On the other hand, for longer horizon, the NMHE gradually learns the mass parameter and hence, a smooth response is obtained from NMPC.

4 Conclusion

We have incorporated NMPC-NMHE framework for the system with uncertainties including slip variations in the off-road ground robotic vehicle due to the change in soil conditions and mass variations for the considered package delivery problem of the aerial robot. Thanks to its learning capability, the accuracy of the NMPC is enhanced by the estimation of parameters by NMHE in both the applications. The outcome of the first application, in which we have estimated the soil condition variations, is that the Euclidean error for the NMPC is about 0.0459 m, which is satisfactory for any agricultural application. In the presented second application, where we have estimated the mass of the UAV for a package delivery task, the learning-based NMPC gives better tracking performance than that of the NMPC without learning. The average Euclidean error for the learning-based NMPC (0.1618 m) is less than that of NMPC without learning (0.2064 m). Since the true values of the mass are known throughout the trajectory, we have also presented the estimation results versus their true values. Overall, the obtained results from both the applications imply that the online leaning-based NMPC substantially improves the tracking performance for the presented robotic applications.

Acknowledgements This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its Medium-Sized Centre funding scheme. The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000598.

References

1. Allgöwer, F., Badgwell, T.A., Qin, J.S., Rawlings, J.B., Wright, S.J.: Nonlinear Predictive Control and Moving Horizon Estimation — An Introductory Overview. Springer, London (1999). https://doi.org/10.1007/978-1-4471-0853-5_19
2. Ariens, D., Houska, B., Ferreau, H., Logist, F.: ACADO: toolkit for automatic control and dynamic optimization. Optimization in Engineering Center (OPTEC), 1.0beta edn. (2010). http://www.acadotoolkit.org/
3. Bouabdallah, S.: Design and Control of Quadrotors with Application to Autonomous Flying, p. 155. EPFL, Lausanne (2007)
4. Bouffard, P., Aswani, A., Tomlin, C.: Learning-based model predictive control on a quadrotor: onboard implementation and experimental results. In: 2012 IEEE International Conference on Robotics and Automation (ICRA), pp. 279–284 (2012). https://doi.org/10.1109/ICRA.2012.6225035
5. Chowdhary, G., Mühlegg, M., How, J.P., Holzapfel, F.: Concurrent Learning Adaptive Model Predictive Control, pp. 29–47. Springer, Berlin (2013). https://doi.org/10.1007/978-3-642-38253-6_3
6. Chowdhary, G., Mühlegg, M., How, J.P., Holzapfel, F.: A concurrent learning adaptive-optimal control architecture for nonlinear systems. In: 52nd IEEE Conference on Decision and Control, pp. 868–873 (2013). https://doi.org/10.1109/CDC.2013.6759991
7. Eren, U., Prach, A., Koçer, B.B., Raković, S.V., Kayacan, E., Açıkmeşe, B.: Model predictive control in aerospace systems: current state and opportunities. J. Guid. Control. Dyn. **40**(7), 1541–1566 (2017). https://doi.org/10.2514/1.G002507
8. Fum, W.Z.: Implementation of simulink controller design on Iris+ quadrotor. Ph.D. thesis, Monterey, California, Naval Postgraduate School (2015)
9. Garimella, G., Sheckells, M., Kobilarov, M.: Robust obstacle avoidance for aerial platforms using adaptive model predictive control. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5876–5882 (2017). https://doi.org/10.1109/ICRA.2017.7989692
10. Grúne, L.: NMPC without terminal constraints. IFAC Proc. Vol. **45**(17), 1–13 (2012)
11. Havoutis, I., Calinon, S.: Supervisory teleoperation with online learning and optimal control. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 1534–1540 (2017)
12. Hoffmann, G.M., Huang, H., Waslander, S.L., Tomlin, C.J.: Quadrotor helicopter flight dynamics and control: theory and experiment. In: Proceedings of the AIAA Guidance, Navigation, and Control Conference, vol. 2, p. 4 (2007)
13. Kayacan, E., Kayacan, E., Ramon, H., Saeys, W.: Distributed nonlinear model predictive control of an autonomous tractor-trailer system. Mechatronics **24**(8), 926–933 (2014)
14. Kayacan, E., Kayacan, E., Ramon, H., Saeys, W.: Nonlinear modeling and identification of an autonomous tractor-trailer system. Comput. Electron. Agric. **106**, 1–10 (2014). https://doi.org/10.1016/j.compag.2014.05.002
15. Kayacan, E., Kayacan, E., Ramon, H., Saeys, W.: Learning in centralized nonlinear model predictive control: application to an autonomous tractor-trailer system. IEEE Trans. Control Syst. Technol. **23**(1), 197–205 (2015)
16. Kayacan, E., Kayacan, E., Ramon, H., Saeys, W.: Robust tube-based decentralized nonlinear model predictive control of an autonomous tractor-trailer system. IEEE/ASME Trans. Mechatron. **20**(1), 447–456 (2015)
17. Kayacan, E., Kayacan, E., Ramon, H., Saeys, W.: Towards agrobots: Identification of the yaw dynamics and trajectory tracking of an autonomous tractor. Comput. Electron. Agric. **115**, 78–87 (2015)
18. Kayacan, E., Peschel, J.M., Kayacan, E.: Centralized, decentralized and distributed nonlinear model predictive control of a tractor-trailer system: a comparative study. In: 2016 Ameri-

can Control Conference (ACC), pp. 4403–4408 (2016). https://doi.org/10.1109/ACC.2016.7525615
19. Kraus, T., Ferreau, H., Kayacan, E., Ramon, H., Baerdemaeker, J.D., Diehl, M., Saeys, W.: Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles. Comput. Electron. Agric. **98**, 25–33 (2013)
20. Kühl, P., Diehl, M., Kraus, T., Schlöder, J.P., Bock, H.G.: A real-time algorithm for moving horizon state and parameter estimation. Comput. Chem. Eng. **35**(1), 71–83 (2011)
21. Liu, Y., Rajappa, S., Montenbruck, J.M., Stegagno, P., Bülthoff, H., Allgöwer, F., Zell, A.: Robust nonlinear control approach to nontrivial maneuvers and obstacle avoidance for quadrotor UAV under disturbances. Robot. Auton. Syst. **98**, 317–332 (2017). https://doi.org/10.1016/j.robot.2017.08.011
22. López-Negrete, R., Biegler, L.T.: A moving horizon estimator for processes with multi-rate measurements: a nonlinear programming sensitivity approach. J. Process Control **22**(4), 677–688 (2012)
23. Mehndiratta, M., Kayacan, E.: Receding horizon control of a 3 DOF helicopter using online estimation of aerodynamic parameters. Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng. (2017). https://doi.org/10.1177/0954410017703414
24. Morari, M., Lee, J.H.: Model predictive control: past, present and future. Comput. Chem. Eng. **23**(4), 667–682 (1999). https://doi.org/10.1016/S0098-1354(98)00301-9
25. Nicolao, G.D., Magni, L., Scattolini, R.: Stabilizing receding-horizon control of nonlinear time-varying systems. IEEE Trans. Autom. Control **43**(7), 1030–1036 (1998)
26. Ostafew, C.J., Schoellig, A.P., Barfoot, T.D.: Robust constrained learning-based NMPC enabling reliable mobile robot path tracking. Int. J. Robot. Res. **35**(13), 1547–1563 (2016). https://doi.org/10.1177/0278364916645661
27. Prach, A., Kayacan, E.: An MPC-based position controller for a tilt-rotor tricopter VTOL UAV. Optim. Control Appl. Methods https://doi.org/10.1002/oca.2350
28. Rao, C.V., Rawlings, J.B., Mayne, D.Q.: Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations. IEEE Trans. Autom. Control **48**(2), 246–258 (2003). https://doi.org/10.1109/TAC.2002.808470
29. Robertson, D.G., Lee, J.H., Rawlings, J.B.: A moving horizon-based approach for least-squares estimation. AIChE J. **42**(8), 2209–2224 (1996)
30. Shin, J., Kim, H.J., Park, S., Kim, Y.: Model predictive flight control using adaptive support vector regression. Neurocomputing **73**(4), 1031–1037 (2010). https://doi.org/10.1016/j.neucom.2009.10.002
31. Vukov, M., Gros, S., Horn, G., Frison, G., Geebelen, K., Jørgensen, J., Swevers, J., Diehl, M.: Real-time nonlinear MPC and MHE for a large-scale mechatronic application. Control Eng. Pract. **45**, 64–78 (2015)