Differentiable Moving Horizon Estimation for Robust Flight Control

Bingheng Wang, Zhengtian Ma, Shupeng Lai, Lin Zhao*, and Tong Heng Lee

Abstract—Estimating and reacting to external disturbances is of fundamental importance for robust control of quadrotors. Existing estimators typically require significant tuning or training with a large amount of data, including the ground truth, to achieve satisfactory performance. This paper proposes a data-efficient differentiable moving horizon estimation (DMHE) algorithm that can automatically tune the MHE parameters online and also adapt to different scenarios. We achieve this by deriving the analytical gradient of the estimated trajectory from MHE with respect to the tuning parameters, enabling end-to-end learning for auto-tuning. Most interestingly, we show that the gradient can be calculated efficiently from a Kalman filter in a recursive form. Moreover, we develop a model-based policy gradient algorithm to learn the parameters directly from the trajectory tracking errors without the need for the ground truth. The proposed DMHE can be further embedded as a layer with other deep neural networks for joint optimization. Finally, we demonstrate the effectiveness of the proposed method via simulation, where challenging scenarios such as ground effect, square-wave and sinusoidal disturbances are examined.

SUPPLEMENTARY MATERIAL

We released our source code at https://github.com/BinghengNUS/DMHE

I. INTRODUCTION

Quadrotors are increasingly engaged in various challenging tasks such as flying in swarms [1] and transporting suspended payload with unknown weights [2]. The interaction with environments constantly generate complex forces and torques acting on quadrotors, such as those from the downwash [3], ground effects [4], wall effects [5], and the time-varying drag from a suspended payload [6], to name but a few. These disturbances affect the quadrotors’ dynamic behavior significantly and thus must be compensated appropriately for robust flight control.

Estimating and reacting to external disturbances has long been the focus of quadrotor research. The authors in [7] proposed a deep neural network (DNN) to learn the aerodynamic disturbance caused by the ground effect when landing quadrotors. A similar method has been further adapted to estimate multi-quadrotor interactions using permutation-invariant DNN [8]. Although these methods can predict the aerodynamic disturbances more accurately than some empirical models, they typically need the ground truth data for training, which is usually difficult to acquire and can only be estimated roughly. It is also difficult to generalize the method to multiple different scenarios due to the large volume of data needed for training. A unified estimation framework that is independent of the trajectory and controller was developed in [9] to estimate external forces and torques. However, the estimator requires the inputs to be carefully filtered since its design does not account for noise.

More recently, an extended-state Unscented Kalman Filter (UKF) was adopted in [10] for the disturbance estimation. The extended-state equations model the time-varying disturbance forces and torques as random walks. This method generally works well across different scenarios. However, its performance heavily depends on the noise variance parameters which are hard to identify in practice. The process of manually tuning these parameters is rather obscure and requires significant effort and expert knowledge. To alleviate this tuning problem, [11] proposed an adaptive extended Kalman filter (EKF) that leverages an online expectation-maximization approach to estimate the predicted error covariance matrix, enabling it to be used in the case of unknown and time-varying noise covariance. However, the EKF lacks of robustness to the poor estimation of the noise covariance due to the linearization approximation.

Different from the above methods, we develop upon the moving horizon estimation (MHE) for the disturbance estimation. The MHE has been shown superior to UKF and EKF in terms of robustness, convergence rate, and estimation accuracy [12, 13]. In MHE, a finite horizon of the past state trajectory is estimated online by solving an optimization problem with the most current measurements. The objective function of MHE is a weighted summation of the so-called arrival cost and running cost. These weighting matrices are tuning parameters of an MHE. They can be interpreted as the inverse of the noise variance matrices, the same as those assumed in a probabilistic filter such as UKF [14]. Similar to UKF, sophisticated tuning is required to achieve optimal performance.

This paper develops a data-efficient differentiable moving horizon estimation (DMHE) algorithm that can automatically tune its parameters online and is adaptive to different flight scenarios. To achieve this, we perform a sensitivity analysis by analytically computing the gradient of the MHE estimates with respect to (w.r.t) the tuning parameters. Specifically, it is derived by implicitly differentiating through the Karush-Kuhn-Tucker (KKT) conditions of the associated MHE optimization problem. The gradient enables us to train the tuning parameters using powerful machine learning techniques. More generally, it enables the embedding of MHE into neural networks for joint optimization. There has been a growing
interest of joining the force of control-theoretic policies and machine learning techniques, such as OptNet [15], differentiable MPC [16] and Pontryagin differentiable programming [17]. Our work adds to this collection another general policy for estimation, which is of interdisciplinary interest to both the control and the machine learning community. Technical-wise, we construct an auxiliary MHE system to calculate the gradient more efficiently. Interestingly, we show that the auxiliary MHE can be solved very efficiently in a recursive form by leveraging a Kalman filter. Driven by the particular application, we also develop a model-based policy gradient algorithm to learn the tuning parameters directly from the quadrotor trajectory tracking error without the need for the ground truth.

Our main contributions are summarized in the following:

1) We propose the auto-tuning DMHE, which is demonstrated to accurately estimate external disturbances acting on quadrotors such as sudden payload change, ground effect, and downwash effect;
2) We develop an efficient method for training DMHE with gradient descent, which explores a recursive form using a Kalman filter;
3) We develop a model-based policy gradient algorithm to learn the tuning parameters directly from the quadrotor trajectory tracking errors without the ground truth of the external disturbances.

The rest of this paper is organized as follows. The quadrotor dynamics is presented in Section II. We derive the analytical gradient in Section III. Section IV presents the model-based policy gradient algorithm for learning the MHE without the ground truth data. Simulation results are reported in Section V. We discuss our future work and conclude the paper in Section VI.

II. PRELIMINARIES: QUADROTOR DYNAMICS

We aim to improve the robustness of flight control of autonomous quadrotors by estimating and reacting to the external disturbances. The quadrotor is modeled as a 6 degree-of-freedom (DoF) rigid body with mass \( m \) and moment of inertia \( J \in \mathbb{R}^{3 \times 3} \). Define \( p \in \mathbb{R}^{3} \) as the global position of CoM in inertial frame \( \mathfrak{I} \) (North-East-Down), \( v \in \mathbb{R}^{3} \) the velocity of CoM in \( \mathfrak{I} \), \( R \in \mathbb{R}^{3 \times 3} \) the rotational matrix from body frame \( B \) to \( \mathfrak{I} \), and \( \omega \in \mathbb{R}^{3} \) the angular velocity in \( B \), the dynamics of the quadrotor is given by:

\[
\begin{align*}
\dot{p} &= v, \quad \dot{v} = m^{-1} (mg \ddot{z} - R f \ddot{z} + d_f) \\
\dot{R} &= R \omega \times, \quad \dot{\omega} = J^{-1} (-\omega \times J \omega + \tau_m + \tau_c)
\end{align*}
\]

(1)

where disturbance forces \( d_f = (d_{fx}, d_{fy}, d_{fz}) \) and torques \( \tau_c = (\tau_{cx}, \tau_{cy}, \tau_{cz}) \) are expressed in \( \mathfrak{I} \) and \( B \), respectively. \( g \) is the gravitational constant, \( \ddot{z} = [0, 0, 1]^T \), \( (\cdot)^\times \) denotes the skew-symmetric operator, \( \tau_m \) is the control torques produced by the motor thrusts. We define \( x_q = [p, v, R, \omega] \) as the quadrotor state and \( \tau_c = [f, \tau_{mx}, \tau_{my}, \tau_{mz}]^T \) as the control wrench. Usually, the squared motor speeds \( u = [\Omega_1^2, \Omega_2^2, \Omega_3^2, \Omega_4^2]^T \) are the control input of the quadrotor and linearly mapped to the control wrench by \( \tau_c = K u \) where \( K \) is a 4-by-4 matrix defined by the aerodynamic coefficients \( b \) and \( k_u \), as well as the distance from each motor to center-of-mass (CoM) \( l \).

Common practice for robust control against disturbances is to estimate and compensate them in feedforward control. Next, based on the state-of-art estimator MHE, we develop algorithms of automatically tuning the MHE online to estimate the disturbances with fast dynamic response.

III. DIFFERENTIABLE MOVING HORIZON ESTIMATION

A. MHE FORMULATION FOR DISTURBANCE ESTIMATION

Since the disturbance can be from arbitrary unknown sources, the most general way is to approximate them as random walks. This has proven to be very effective for estimating time-varying unknown disturbances [10], [18].

\[
d_f = \eta_f, \quad d_c = \eta_c
\]

(2)

where \( \eta_f \) and \( \eta_c \) denote the process noises for the disturbance forces and torques, respectively. We then extend \( x_q \) with \( d = [d_f, d_c] \) to obtain \( x = [p, v, R, \omega, f_f, d_c] \). Both the rigid body dynamics (1) and the disturbance model (2) will be used in MHE for estimation.

Given the most recent measurements in a data window with a horizon of \( N \), the MHE estimates the trajectory of the extended state \( \{x_k\}_{t-N+1}^t \) by solving the following optimization problem at each time step \( t \):

\[
\begin{align*}
\min_{\hat{x}, \eta} J &= \frac{1}{2} \|x_{t-N+1} - \hat{x}_{t-N+1}\|_P^2 \\
&+ \frac{1}{2} \sum_{k=t-N+1}^t \|\hat{y}_k - y_k\|_R^2 + \frac{1}{2} \sum_{k=t-N+1}^{t-1} \|\eta_k\|_Q_k^2
\end{align*}
\]

s.t.

\[
\begin{align*}
x_{k+1} &= f(x_k, u_k, \eta_k, \Delta t) \\
y_k &= h(x_k)
\end{align*}
\]

(3)

where \( \hat{x}_{t-N+1} = \hat{x}_{t-N+1}\| \) is the MHE estimate of \( x_{t-N+1} \) at time \( t - 1 \), \( y_k \) is the measurements at time \( k \), \( h(x_k) \) is the predicted output, \( f \) is the discrete-time model for the extended system using 1st-order Euler method, \( \eta = [\eta_f, \eta_c] \), \( \Delta t \) is sampling time, \( P, R_k, \) and \( Q_k \) are weight matrices. For brevity, we define the initial and terminal subscripts to be 1 and \( N \), respectively, such that the estimate trajectory becomes \( \{x_k\}_{1}^{N} \).

We define a row vector \( \theta \) as the tuning parameters that contain all the elements of those weight matrices, i.e. \( \theta = vec(P, R_{1:N}, Q_{1:N-1}) \) where \( vec(\cdot) \) is the vectorization operator. The first term in (3) approximates the arrival cost that summarizes the past data not explicitly accounted for in the cost [19]. We observe that increasing \( P \) will slow down the dynamic response of MHE to the change of disturbance, but a very small \( P \) will make the MHE to be unstable. The second and third terms are a trade-off between the measurements and the model. If we are highly confident in the measurements based on a prior knowledge, then we increase \( R_k \) relative to \( Q_k \). Despite these rough intuitions, tuning \( \theta \) to improve the estimate of disturbance \( d \) can still be a quite tedious and intricate process.
The above intuitions indicate that the estimate trajectory \(x^0\) is an implicit function of \(\theta\) by solving the MHE problem. For a choice of \(\theta\), we refer to the problem (3) as \(\Xi(\theta)\) and evaluate \(x^0\) using a scalar differentiable loss \(L(x^0)\). Our objective is to train \(\theta\) such that the loss is minimized, which is cast as the following bilevel optimization problem:

\[
\min_{\theta} L(x^0) \quad \text{s.t.} \quad x^0 \text{ is generated by } \Xi(\theta) \tag{4}
\]

B. Analytical Gradient

We use gradient descent to solve the problem (4), which allows for tuning DMHE in end-to-end learning pipelines. The gradient of the loss \(L(x^0)\) with respect to \(\theta\) is computed using the chain rule:

\[
\frac{dL}{d\theta} = \frac{dL}{dx^0} \frac{dx^0}{d\theta} \tag{5}
\]

With (5), we illustrate the learning framework in Fig.1. Each update of \(\theta\) involves a forward pass where given \(\theta_t\), \(x^0\) is generated by solving MHE, and thus \(L(x^0)\) is formed, and a backward pass where \(\frac{dL}{dx^0}\) and \(\frac{dL}{d\theta}\) are computed.

![Fig. 1: End-to-end learning pipeline of the DMHE where blocks in grey are in the forward pass and blocks in white are in the backward pass.](image)

Since \(x^0\) can be obtained by any nonlinear optimization solver, the main challenge is how to solve \(\frac{dL}{d\theta}\). Note that the gradient \(\frac{dL}{d\theta}\) is also an implicit function of \(\theta\), which justifies the chain rule (5) for updating \(\theta\). Next, we will present an efficient way to compute the gradient analytically in a recursive form by proposing an auxiliary MHE system.

For the optimization problem \(\Xi(\theta)\) (3) with a given \(\theta\), the KKT conditions define a set of necessary optimality conditions which \(x^0\) must satisfy. We associate dual variable \(\lambda^0\) with the state constraints and the corresponding Lagrangian is thus formed as:

\[
\mathcal{L} = J + \sum_{k=1}^{N-1} \lambda_k^T (x_{k+1} - f(x_k, u_k, \eta_k)) \tag{6}
\]

Let \(\hat{x}_{k|k}\) be the estimate of the current state, \(\hat{x}_{k|N}, k \leq N\) the data smooth based on the measurements \(\{y_k\}_{k=1}^N\), and \(\hat{x}_{k+1|k}\) the one-step model prediction. Therefore, the KKT conditions take the following form:

\[
\frac{\partial \mathcal{L}}{\partial \hat{x}_{k|N}} = -H^T R_k (y_k - h(\hat{x}_{k|N}^0)) + \lambda_{k-1}^0
\]

\[
-A_k^T \lambda_k^0 = 0
\]

\[
\frac{\partial \mathcal{L}}{\partial \eta_k^0} = Q_k \eta_k^0 - B_k^T \lambda_k^0 = 0 \tag{7a}
\]

\[
\frac{\partial \mathcal{L}}{\partial \lambda_k^0} = \hat{x}_{k+1|N} - f(\hat{x}_{k|N}, u_k, \eta_k) = 0
\]

with the boundary conditions:

\[
\hat{x}_{1|N} = \hat{x}_1 + P^{-1} A_1^T \lambda_1^0 + P^{-1} H^T R_1 (y_1 - h(\hat{x}_{1|N}^0)) \tag{7b}
\]

where \(k = 2, \ldots, N\) in the first equation of (7a), \(k = 1\) in the second and third equations of (7a), \(A_k = \frac{\partial f}{\partial x_k}, B_k = \frac{\partial f}{\partial \eta_k}, H = \frac{\partial h}{\partial x_k}\), and \(G_k = \frac{\partial A_k \lambda_k^0}{\partial x_k}\) are the system matrices independent of \(\theta\), \(\lambda_N^0 = 0\) is because the terminal cost in the MHE is zero.

As mentioned before, our goal is to obtain \(\frac{\partial \hat{x}_{k|N}}{\partial \theta}\). To this end, we define the following new state, new variables, and new process noise as:

\[
\frac{\partial \hat{x}_{k|N}}{\partial \theta} = \hat{X}_{k|N}, \frac{\partial \lambda_k}{\partial \theta} = \Lambda_k, \frac{\partial \eta_k}{\partial \theta} = W_k \tag{8}
\]

In addition, the following matrices are also defined:

\[
D_k = \frac{\partial B_k Q^{-1} B_k^T}{\partial \theta} \lambda_k, E_k = \frac{\partial H^T R_k}{\partial \theta} (y_k - h(\hat{x}_{k|N}^0))
\]

\[
F = \frac{\partial P^{-1} H^T R_1}{\partial \theta} (y_1 - h(\hat{x}_{1|N}^0)) + \frac{\partial P^{-1} A_1^T}{\partial \theta} \lambda_1
\]

The partial derivatives (9) and the system matrices can be easily obtained by any software that supports symbolic computation, e.g. CasADi [20]. By eliminating \(\eta_k\) with \(\lambda_k\) in (7a) and using the above definitions (8) and (9), we can differentiate the KKT conditions (7a) and (7b) w.r.t \(\theta\), leading to the differential KKT conditions:

\[
\hat{X}_{k+1|N} = A_k \hat{X}_{k|N} + B_k Q_k^{-1} B_k^T \Lambda_k + D_k
\]

\[
\Lambda_{k-1} = A_k^T \Lambda_k + (G_k - H^T R_k H) \hat{X}_{k|N} + E_k
\]

\[
\hat{X}_{1|N} = F + P^{-1} (G_1 - H^T R_1 H) \hat{X}_{1|N} + P^{-1} A_1^T \Lambda_1
\]

where \(I\) is the identity matrix, \(\frac{\partial \eta_k}{\partial \theta} = 0\), and \(\frac{\partial \hat{x}_{k|N}}{\partial \theta} = \Lambda_N = 0\).

Next, we will show that the differential KKT (10) can elegantly calculate the unknown matrix \(\hat{X}_{k|N}\) in (8) which proves to be exactly the output of an auxiliary MHE system of the following form:

\[
\min_{\hat{X}, \hat{W}} \hat{J} = \frac{1}{2} \text{Tr} \left| \hat{X}_{1|N} - F \right|^2 P + \sum_{k=2}^{N} \text{Tr} \left( E_k \hat{X}_{k|N} \right)
\]

\[
+ \frac{1}{2} \sum_{k=1}^{N} \text{Tr} \left( \hat{X}_{k|N} \right)^2 (H^T R_k H - G_k) + \frac{1}{2} \sum_{k=1}^{N-1} \text{Tr} \left( \hat{W}_k \right)^2 Q_k
\]

s.t. \(\hat{X}_{k+1|N} = A_k \hat{X}_{k|N} + B_k W_k + D_k\)

where \(\text{Tr}(\cdot)\) denotes matrix trace. Interestingly, the equality constraint is affine and the cost function is quadratic. In addition, both the dynamics and solution of the system (11) are determined by the trajectory \(x^0\) of the original MHE (3). Hence, we obtain the following important results.

**Lemma 1**: Let \(\hat{X}_{1|N}^0\) be the output of the auxiliary MHE \(\Xi(x^0)\) (11). It satisfies the KKT conditions (10) of \(\Xi(x^0)\) and is exactly the gradient of the estimate trajectory of the original MHE system \(\Xi(\theta)\) w.r.t \(\theta\).
The proof can be found in Appendix A of [21]. Thanks to Lemma 1, we can obtain $\frac{\partial x_i}{\partial \theta}$ from $\Xi(x^\theta)$ efficiently using the following lemma.

**Lemma 2:** Given the initial condition:

$$
\dot{X}_{1|1} = F + [I - P^{-1}(G_1 - H^TR_1H)]^{-1}P^{-1} \times (G_1 - H^TR_1H) F
$$

The analytical solution of the auxiliary MHE system (11) can be obtained in the form of Kalman filter by leveraging dynamic programming [22], as below.

**Kalman filter:**

$$
\dot{X}_{k|k-1} = A_{k-1}\dot{X}_{k-1|k-1} + D_{k-1}
$$
$$
\dot{P}_k = A_{k-1}C_{k-1}^T + B_{k-1}Q_{k-1}B_{k-1}^T - P_k
$$
$$
C_k = [I - P_k(G_k - H^TR_kH)]^{-1}P_k
$$
$$
\dot{X}_{k|k} = [I + C_k(G_k - H^TR_kH)]\dot{X}_{k|k-1} + C_kE_k
$$

**Backward update of dual variables:**

$$
\Lambda_{k-1} = [I + (G_k - H^TR_kH)C_k]A_k^T\Lambda_k + E_k
$$

**Forward update of new state:**

$$
\dot{X}_{k|N} = \dot{X}_{k|k} + C_kA_k^T\Lambda_k
$$

The proof of Lemma 2 is presented in Appendix B of [21]. We summarize the above procedures in Algorithm 1. The analytical gradient makes explicit use of a Kalman filter and has a recursive form. As such, it can be solved iteratively and should be computationally efficient.

**Algorithm 1:** Solving the analytic solution of $\frac{\partial x_i}{\partial \theta}$

---

**Input:** The trajectory $x^\theta$ and the control $u$

1. Initialize $\dot{X}_{1|1}$ (12) and $\dot{P}_1 = P^{-1}$;

2. for $k \leftarrow 2$ to $N$
   3. Use Kalman filter (13a) to obtain $\dot{X}_{k|k-1}, \dot{P}_k, C_k$ and $\dot{X}_{k|k}$

4. end for

5. Set $\Lambda_N = 0$

6. for $k \leftarrow N$ to $2$
   7. Update $\Lambda_{k-1}$ backward in time using (13b)

8. end for

9. for $k \leftarrow 2$ to $N$
   10. Update $\dot{X}_{k|N}$ forward in time using (13c)

11. end for

**Return:** $\frac{\partial x_i}{\partial \theta} = \left\{ \dot{X}_{\theta|N} \right\}_1^N$

---

IV. MODEL-BASED POLICY GRADIENT

Given the analytical gradient (5), we propose a model-based policy gradient algorithm that learns the tuning parameters directly from the trajectory tracking errors, without the need for the ground truth. The algorithm is valid only when the quadrotor is under closed-loop control.

Suppose we have a general controller that takes the estimate of the quadrotor state $\hat{x}_{\theta|i}$ as the feedback and accounts for the estimate of the disturbances $\hat{d}_{\theta|i}$.

$$
u_t = u(r_t, \hat{x}_{\theta|i}, \hat{d}_{\theta|i})
$$

where $r$ is the reference trajectory. If the estimate trajectory $x^\theta$ from the MHE becomes more accurate, the tracking performance of the quadrotor under control should be improved. Otherwise, the quadrotor will deviate from the reference trajectory dramatically and even become unstable.

As such, we define the loss function for training $\theta$ as the weighted summation of the trajectory tracking errors over horizon $N$ which is the batch size.

$$
L(x^\theta) = \sum_{k=1}^{N} \gamma_k \|\hat{x}^q_{k|N} - x^q_{ref}\|_2
$$

where $\beta$ is a positive constant, $\kappa$ is a weight matrix for the tracking errors, and $\gamma_k$ is the softmax weight related to $l_k$.

We summarize the model-based policy gradient in Algorithm 2, in which $l_{mean}$ is the mean loss of the training episode, $\varepsilon$ is the learning rate, and $T_{epi}$ is the duration of each episode.

**Algorithm 2:** Model-based Policy Gradient

---

**Reference trajectory:** $p_d, v_d, \dot{v}_d$, and $b_{1d}$

1. **while** $l_{mean}$ not converged **do**

2. Initialization: $x_0, u_0$, learning rate $\varepsilon$

3. for $t \leftarrow 0$ to $T_{epi}$

4. Solve the MHE (3) to obtain $x^\theta$;

5. Obtain $\frac{\partial x_i}{\partial \theta}$ using Algorithm 1;

6. Solve the control law (14);

7. Implement the control to obtain new state;

8. Obtain $L(x^\theta)$ using (15);

9. Apply the chain rule (5) to obtain $\frac{DL}{\theta_i}$;

10. Update $\theta_{i+1} \leftarrow \theta_i - \varepsilon \frac{DL}{\theta_i}$

11. end for

12. **end while**

---

V. TRAINING IN SIMULATION

This section presents the training results to validate that the model-based policy gradient algorithm can tune the MHE online to improve both the estimation and trajectory tracking performances directly from the trajectory tracking errors. In each training episode, the quadrotor under the geometric control [23] coupled with the DME will takeoff from the ground and follow a Lemniscate trajectory while being affected by various disturbances such as the ground effect, the square-wave and sinusoidal disturbances.

We introduce two forgetting factors ($\gamma_1, \gamma_2$) in the weight matrices $R_k$ and $Q_k$, respectively, such that $R_k = \gamma_1^{N-k}R_N$ and $Q_k = \gamma_2^{N-k}Q_N$. The weight matrices are assumed to be diagonal, in which case $\theta$ contains 50 elements, i.e., $\theta = [p_i, \gamma_1, r_j, \gamma_2, q_k]$ where $p_i, i = 1, \ldots, 24$, $r_j, j =$
The variance of process noise as shown in Fig. 2 with four different initial conditions of the forgetting factor \( \gamma_1, \gamma_2 \) where condition1 = (0.4, 0.8), condition2 = (0.6, 0.8), condition3 = (0.8, 0.8), and condition4 = (0.8, 0.6); (b) Lemniscate tracking performances in training of the DMHE with the initial condition 1, which are under the disturbances shown in Fig. 2c; (c) Comparison of disturbance estimation performances between the trained DMHE and the untrained DMHE with the initial condition 1, the top figure compares the estimate of the disturbance force in z-direction of initial frame, the disturbance force during \( 0 \sim 2s \) is the ground effect which is captured by the low-order model in [24], and the bottom figure compares the estimate of the disturbance torque in z-direction of body frame.

We further test the algorithm complexity by measuring the variance of measurement noise as shown in Fig. 3 with the simulation step of 0.01s and the quadrotor model (\( (0.0183, 0.0197, 0.0322) \) kgm, \( l = 0.21m, b = 1.024 \times 10^{-7} \) and \( k_a = 1.303 \times 10^{-9} \). We apply the line search algorithm to guarantee \( \gamma_1, \gamma_2 \in [\gamma_{\text{min}}, 1] \) where the lower bound \( \gamma_{\text{min}} \) is tuned to be 0.2. The sampling time is 0.01s and the quadrotor model (1) used in the simulation environment is discretized by 4th-order Runge Kutta method with the simulation step of 0.005s. We code the simulation in Python and use IPOPT as the solver in CasADi to implement optimization. The simulation is run in a PC with the processor of AMD Ryzen 9 5950X 16-Core.

Each episode lasts for 15.71s, during which the quadrotor is controlled to track the reference trajectory (see Fig. 2b) generated by the minimum snap algorithm [25]. We set the variance of measurement noise as \( \sigma_{m}^2 = 1 \times 10^{-6} \), and the variance of process noise as \( \sigma_{p}^2 = 1 \times 10^{-2} \). Fig. 2a shows that the proposed algorithm is robust to different initial conditions and all the mean losses can converge to the same local minimum after training few episodes. Correspondingly, Fig. 2b shows that the tracking performance in training is continuously improved, especially in z-direction. This is mainly due to the evident improvement in the disturbance force estimation (see Fig. 2c). By comparison, the untrained DMHE responds slowly to the time-varying disturbances with large estimation bias and delay.

Note that the temporarily large tracking error during takeoff in the 1st training is caused by the initial increase of \( \gamma_2 \) (see Fig. 3, which begins at around 230 steps). This is due to the corresponding square-wave disturbances which subsequently lead to several impulse-like oscillations of \( \gamma_2 \). Despite this, \( \gamma_2 \) can decrease and is eventually stable at around 0.2 while \( \gamma_1 \) increases to around 0.96. In addition, all the elements in \( Q_N \) are less than those in \( R_N \), and the optimal \( \left( \gamma_1, \gamma_2 \right) \) significantly enlarge the difference between \( R_N \) and \( Q_N \). The trained weight matrices agree with the settings of the noise variances, implying that we have more confidence on the measurements than on the process.

| Horizon | 10 | 20 | 30 | 40 | 50 |
|---------|----|----|----|----|----|
| Running time | 3ms | 5.6ms | 8.7ms | 11.9ms | 13.1ms |

**TABLE 1**: Comparison of mean running time for the whole Algorithm 1 over one episode with different batch sizes.

We further test the algorithm complexity by measuring the
running time needed for the whole algorithm 1 per iteration. Table I summarizes the running time for different batch sizes. It is shown that the proposed algorithm 1 is computationally efficient as it can be run at high frequency.

Fixed tuning parameters may not work well in more challenging scenarios, where the quadrotor performs agile maneuvers such as random trajectory with high velocity up to its physical limit and the profiles of external disturbances will change significantly as well. In this case, it would be promising to model \( \theta \) using DNNs such that the resulting neural MHE can adapt to various challenging trajectories and disturbances. The neural MHE can be trained by embedding the DMHE as a layer into DNNs for joint-optimization. According to chain rule, the corresponding loss function is formulated as
\[
\text{loss}_{dnn} = \frac{d}{d\theta} \left( \text{loss}_{dnn} \right)
\]
where \( \text{loss}_{dnn} \) denotes that \( \theta \) is a function of the DNNs parameters \( \rho_{dnn} \). This new loss function can be minimized by any available machine learning tools like Adam optimizer in PyTorch [26].

VI. CONCLUSIONS

This paper proposes a data-efficient differentiable moving horizon estimation (DMHE) algorithm that can automatically tune the MHE parameters online to estimate the disturbances acting on quadrotors. At the core of our approach is the analytical gradient of the estimate trajectory of MHE w.r.t the tuning parameters, which enables auto-tuning by end-to-end learning. Simulation shows that a stable MHE with fast dynamic response can be learned directly from the trajectory tracking errors without the need for the ground truth by the proposed model-based policy gradient.

Our future work will incorporate the DMHE into DNNs for joint-optimization and apply the neural MHE to the simultaneous localization and mapping (SLAM) system in more challenging scenarios such as varying both disturbance forces and torques drastically. We believe the tightly coupled DMHE and SLAM system will further improve the robustness of the current quadrotor control and SLAM system.

REFERENCES

[1] W. Höning, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayani, “Trajectory planning for quadrotor swarms,” IEEE Transactions on Robotics, vol. 34, no. 4, pp. 856–869, 2018.
[2] S. Dai, T. Lee, and D. S. Bernstein, “Adaptive control of a quadrotor uav transporting a cable-suspended load with unknown mass,” in 53rd IEEE Conference on Decision and Control. IEEE, 2014, pp. 6149–6154.
[3] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, “The grasp multiple micro-uav testbed,” IEEE Robotics & Automation Magazine, vol. 17, no. 3, pp. 56–65, 2010.
[4] C. Powers, D. Mellinger, A. Kushleyev, B. Kothenmann, and V. Kumar, “Influence of aerodynamics and proximity effects in quadrotor flight,” in Experimental robotics. Springer, 2013, pp. 289–302.
[5] Y. Naka and A. Kagami, “Coanda effect of a propeller airflow and its aerodynamic impact on the thrust,” Journal of Fluid Science and Technology, vol. 15, no. 3, pp. JFT0016–JFT0016, 2020.
[6] S. Belkhale, R. Li, G. Kahn, R. McAllister, R. Calandra, and S. Levine, “Model-based meta-reinforcement learning for flight with suspended payloads,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1471–1478, 2021.
[7] G. Shi, X. Shi, M. O’Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, “Neural lander: Stable drone landing control using learned dynamics,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 9784–9790.
[8] G. Shi, W. Höning, Y. Yue, and S.-J. Chung, “Neural-swarm: Decentralized close-proximity multirotor control using learned interactions,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 3241–3247.
[9] T. Tomić and S. Haddadin, “A unified framework for external wrench estimation, interaction control and collision reflexes for flying robots,” in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014, pp. 4197–4204.
[10] C. D. McKinnon and A. P. Schoellig, “Estimating and reacting to forces and torques resulting from common aerodynamic disturbances acting on quadrotors,” Robotics and Autonomous Systems, vol. 123, p. 103314, 2020.
[11] Y. Huang, Y. Zhang, B. Xu, Z. Wu, and J. A. Chambers, “A new adaptive extended kalman filter for cooperative localization,” IEEE Transactions on Aerospace and Electronic Systems, vol. 54, no. 1, pp. 353–368, 2017.
[12] J. B. Rawlings and B. R. Bakshi, “Particle filtering and moving horizon estimation,” Computers & chemical engineering, vol. 30, no. 10–12, pp. 1529–1541, 2006.
[13] W. Zhang, Z. Wang, C. Zou, L. Druge, and M. Nybacka, “Advanced vehicle state monitoring: evaluating moving horizon estimators and unscented kalman filter,” IEEE Transactions on Vehicular Technology, vol. 68, no. 6, pp. 5430–5442, 2019.
[14] T. Kraus, H. J. Ferreau, E. Kayacan, H. Ramon, J. De Baerdemaecker, M. Diehl, and W. Saeyts, “Moving horizon estimation and nonlinear model predictive control for autonomous agricultural vehicles,” Computers and electronics in agriculture, vol. 98, pp. 25–33, 2013.
[15] B. Amos and J. Z. Kolter, “OptNet: Differentiable optimization as a layer in neural networks,” in International Conference on Machine Learning. PMLR, 2017, pp. 136–145.
[16] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter, “Differentiable mpc for end-to-end planning and control,” in Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 8299–8310.
[17] W. Jin, Z. Wang, Z. Yang, and S. Mou, “Pontryagin differentiable programming: An end-to-end learning and control framework,” Advances in Neural Information Processing Systems, vol. 33, 2020.
[18] B. Yüksel, C. Secchi, H. H. Bulthoff, and A. Franchi, “A nonlinear force observer for quadrotors and application to physical interactive tasks,” in 2014 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, 2014, pp. 433–440.
[19] C. V. Rao, J. B. Rawlings, and D. Q. Mayne, “Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations,” IEEE transactions on automatic control, vol. 48, no. 2, pp. 246–258, 2003.
[20] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “Casadi: a software framework for nonlinear optimization and optimal control,” Mathematical Programming Computation, vol. 11, no. 1, pp. 1–36, 2019.
[21] B. Wang, Z. Ma, S. Lai, L. Zhao, and T. H. Lee, “Differentiable moving horizon estimation for robust flight control,” arXiv preprint arXiv:2108.03212, 2021. [Online]. Available: https://arxiv.org/abs/2108.03212
[22] H. Cox, “Estimation of state variables for noisy dynamic systems,” Ph.D. dissertation, Massachusetts Institute of Technology, 1963.
[23] T. Lee, M. Leok, and N. H. McClamroch, “Geometric tracking control of a quadrotor UAV on SE(3),” in 49th IEEE conference on decision and control (CDC). IEEE, 2010, pp. 5420–5425.
[24] X. He, G. Kou, M. Cafaf, and K. K. Leang, “In-ground-effect modeling and nonlinear-disturbance observer for multirotor unmanned aerial vehicle control,” Journal of Dynamic Systems, Measurement, and Control, vol. 141, no. 7, 2019.
[25] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in 2011 IEEE international conference on robotics and automation. IEEE, 2011, pp. 2520–2525.
[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

3568