Online Residual Model Learning for Model Predictive Control of Autonomous Surface Vehicles in Real-World Environments

Arturo Gamboa-Gonzalez[1], Chunlin Li[1,2], Michael Wehner[1], and Wei Wang[1,*]

***Abstract*— Model predictive control (MPC) relies on an accurate dynamics model to achieve precise and safe robot operation. In complex and dynamic aquatic environments, developing an accurate model that captures hydrodynamic details and accounts for environmental disturbances like waves, currents, and winds is challenging for aquatic robots. In this paper, we propose an online residual model learning framework for MPC, which leverages approximate models to learn complex unmodeled dynamics and environmental disturbances in dynamic aquatic environments. We integrate offline learning from previous simulation experience with online learning from the robot's real-time interactions with the environments. These three components—residual modeling, offline learning, and online learning—enable a highly sample-efficient learning process, allowing for accurate real-time inference of model dynamics in complex and dynamic conditions. We further integrate this online learning residual model into a nonlinear model predictive controller, enabling it to actively choose the optimal control actions that optimize the control performance. Extensive simulations and real-world experiments with an autonomous surface vehicle demonstrate that our residual model learning MPC significantly outperforms conventional MPCs in dynamic field environments.**

I. Introduction

Autonomous Surface Vessels (ASVs) are playing an increasing role in both civilian and military sectors, with potential applications in transportation, defense, surveillance, and environmental monitoring [1]–[10].

For mission success, accurate trajectory tracking is crucial for ASVs. Various control schemes have been proposed to address the trajectory tracking challenge, including sliding mode controllers [11]–[13] and data-driven algorithms [14], [15]. Among them, Model Predictive Control (MPC) [4], [9], [10] is widely favored for trajectory tracking due to its ability to handle system dynamics while managing input and state constraints. The performance of MPC heavily depends on the accuracy of the system's dynamics model. Current simplified analytical model for ASVs is insufficient to accurately capture complex hydrodynamic interactions, restricting the control performance of MPC for ASVs. Additionally, in real-world environments, external disturbances like wind, waves, and currents further degrade MPC performance significantly.

Neural networks (NNs) offer a promising way to enhance the accuracy of robot dynamics models due to their ability to

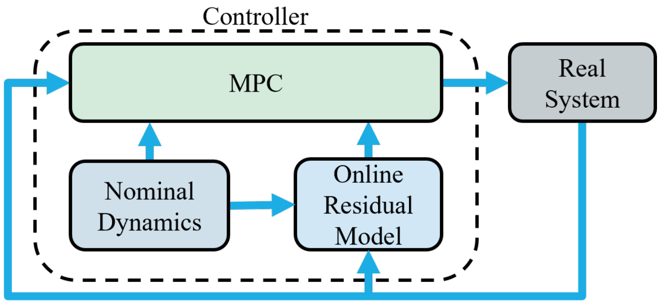

Fig. 1. Overview of the proposed online residual model learning MPC framework. The residual error is computed from the environment and nominal model, and used to update a NN in real time.

approximate nonlinear functions [16]–[21]. Previous studies [22]–[24] have leveraged NNs to predict residual models or full dynamics models, thereby improving the performance of MPC. In [23], a data-driven residual model is used to compensate for inaccuracies in the dynamics model of a highly agile quadrotor platform. However, this approach lacks online learning capabilities and only considers a single time step as input in the residual NN model. [16] addresses this single time step limitation by employing a NN with a backward horizon of state inputs to predict the dynamics model of a driverless car more accurately under varying operating conditions, though it remains constrained to offline learning. In contrast, [24] integrates both offline and online learning to develop a discrete neural model for a quadrotor that adapts continuously to previously unseen flight conditions. Nonetheless, this method does not incorporate historical state information, which may limit the convergence speed of the learning process and reduce its ability to predict more complex and dynamic operating conditions.

Training a NN to model the complex dynamics of ASVs in field environments with environmental disturbances is a challenging task, often requiring high-fidelity simulation environments and significant computational resources. There is limited research on learning the nonlinear dynamics of ASVs using NNs in the existing literature [25]. In this paper, we propose an online residual model learning framework for model predictive control that leverages approximate models to capture complex unmodeled dynamics and environmental disturbances in dynamic aquatic environments. Given the intricate nature of these dynamics, single-step predictions may fail to adequately capture the system's highly nonlinear behavior. To overcome this limitation, our NN uses a

[1]Marine Robotics Lab, Department of Mechanical Engineering, College of Engineering, University of Wisconsin-Madison.

[2]Department of Computer Science, University of Toronto, Canada.

This work was supported in part by the Department of Mechanical Engineering, the College of Engineering, and the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education funded by the Wisconsin Alumni Research Foundation.

*Corresponding author: `wwang745@wisc.edu`.

range of historical state-input pairs, enabling more accurate predictions of how complex environmental factors affect the ASV.

We successfully incorporate this residual model learning into a MPC framework and fully deploy it on an autonomous surface vehicle operating in the highly dynamic environment of Lake Mendota in Madison, Wisconsin (WI). The main contributions of our approach include:

- an online residual model learning framework for model predictive control that leverages approximate models to learn complex dynamics in aquatic field environments;
- integration of this online residual learning model into a nonlinear model predictive controller, enabling it to actively select optimal control actions that enhance performance;
- a comprehensive study involving numerical simulations and field experiments with an autonomous surface vehicle (ASV), validating the effectiveness of the online residual model learning MPC framework.

II. Robot Prototype and Dynamics Overview

In this section, we introduce the ASV prototype, as illustrated in Fig. 2, along with its nominal dynamic model. The nominal model will later be enhanced with a learned residual model for the MPC, which will be discussed in the subsequent section.

A. Autonomous Surface Vessel Prototype

The hull design of our ASV is based on a previous robot model [26], [27]. The ASV (shown in Fig. 2) measures 0.90 m in length, 0.45 m in width, 0.14 m in height, and weighs 15 kg. The ASV features four thrusters positioned around the hull (Fig. 3), allowing for holonomic motion. We upgraded the computing system to an Intel NUC 13 Pro Kit Mini Computer with 32 GB of RAM, selected for its compact size, lightweight, and high computational capacity. We upgraded the 3D LiDAR sensor to a Velodyne VLP-32C LiDAR, paired with a Microstrain 3DM-GX5-25 Inertial Measurement Unit (IMU) to perform real-time LiDAR Inertial Odometry [28], achieving centimeter-level localization accuracy. The NUC sends force commands to a STM 32 microcontroller, which translates them into PWM signals for the BlueRobotics T100 Thrusters.

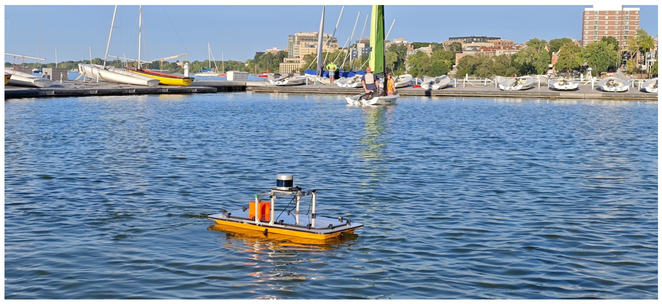

Fig. 2. The ASV prototype navigating on Lake Mendota in Madison, WI, USA.

B. Nominal System Dynamics

The system dynamics of our ASV can be approximated by the following differential equation [29]:

$$\mathbf{M}\dot{\mathbf{v}} + \mathbf{C}(\mathbf{v})\mathbf{v} + \mathbf{D}(\mathbf{v})\mathbf{v} = \boldsymbol{\tau} \tag{1}$$

where $\mathbf{v} = [u \quad v \quad r]^T$ represents the vehicle's forward, lateral, and angular velocity in the body frame, respectively, $\mathbf{M} \in \mathbb{R}^{3\times3}$ is the positive-definite symmetric mass and inertia matrix, $\mathbf{C}(\mathbf{v}) \in \mathbb{R}^{3\times3}$ is the skew-symmetric vehicle matrix of Coriolis and centripetal terms, $\mathbf{D}(\mathbf{v}) \in \mathbb{R}^{3\times3}$ is the positive-semi-definite drag matrix-valued function, and $\boldsymbol{\tau} \in \mathbb{R}^{3\times1}$ is the vector of body frame forces and moments acting on the vehicle in all three degrees of freedom.

The applied force and moment vector $\boldsymbol{\tau}$ is given by:

$$\boldsymbol{\tau} = \mathbf{B}\mathbf{u} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ \frac{a}{2} & -\frac{a}{2} & \frac{b}{2} & -\frac{b}{2} \end{bmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} \tag{2}$$

where $\mathbf{B}$ is the control matrix defining the thruster configuration, $\mathbf{u}$ is the control vector, a represents the distance between the transverse thrusters, and b is the distance between the longitudinal thrusters. f_1, f_2, f_3, and f_4 are the forces generated by the corresponding propellers, as shown in Figure 3. Each thruster is fixed and can produce both forward and reverse thrust.

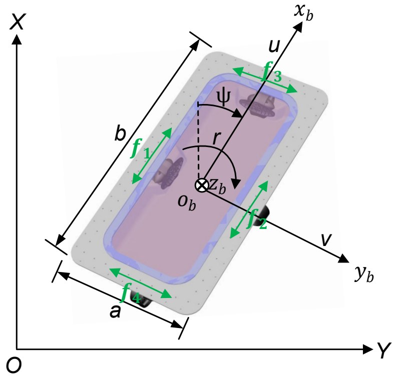

Fig. 3. The ASV coordinate system, with four thrusters positioned on each side of the robot. Each thruster can generate both forward and reverse forces.

An inertial reference frame is defined by $\boldsymbol{\eta} = [x \quad y \quad \psi]^T$, where x and y represent the position of the ASV, and ψ denotes its orientation. The kinematic equation that relates velocity components in the inertial frame to those in the body frame is given by:

$$\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\mathbf{v} \tag{3}$$

where $\mathbf{R}(\psi)$ is the transformation matrix that converts a state vector from the body frame to the inertial frame:

$$\mathbf{R}(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Finally, the system dynamics from Eqn. (1) and the kinematic relation from Eqn. (3) can be written as:

$$\begin{aligned}\dot{\boldsymbol{\eta}} &= \mathbf{R}(\psi)\mathbf{v}\\ \dot{\mathbf{v}} &= \mathbf{M}^{-1}\mathbf{B}\mathbf{u} - \mathbf{M}^{-1}(\mathbf{C}(\mathbf{v}) + \mathbf{D}(\mathbf{v}))\mathbf{v}\end{aligned} \tag{5}$$

By combining these two equations, we obtain the nominal system dynamics:

$$\dot{\mathbf{q}} = f_{\mathcal{F}}(\mathbf{q}, \mathbf{u}) \tag{6}$$

where $\mathbf{q}$ represents the state vector and $\mathbf{u}$ the control vector. Specifically, the state vector $\mathbf{q} = [\boldsymbol{\eta} \quad \mathbf{v}]^T \in \mathbb{R}^{n_\mathbf{q}}$ and the control vector $\mathbf{u} \in \mathbb{R}^{n_\mathbf{u}}$ are defined as:

$$\mathbf{q} = [x \quad y \quad \psi \quad u \quad v \quad r]^T, \tag{7}$$

$$\mathbf{u} = [f_1 \quad f_2 \quad f_3 \quad f_4]^T. \tag{8}$$

The system model in (6) describes how the state $\mathbf{q}$ evolves based on the control input $\mathbf{u}$.

C. Augmented Robot Dynamics

The nominal model $f_{\mathcal{F}}(\mathbf{q}, \mathbf{u})$ is but a simplified approximation of the true system dynamics $f_{\text{true}}(\mathbf{q}, \mathbf{u})$ which may include unmodeled complexities or parametric discrepancies. Thus, we can define the true error dynamics as:

$$f_{\text{true}}(\mathbf{q}, \mathbf{u}) = f_{\mathcal{F}}(\mathbf{q}, \mathbf{u}) + f_{\text{err}}(\mathbf{q}, \mathbf{u}) \tag{9}$$

where $f_{\text{err}}(\mathbf{q}, \mathbf{u})$ is the error from simplification and parameter inaccuracy.

Furthermore, ASVs need to operate in environments with disturbances—such as winds, waves, and currents—which are difficult to model deterministically. These disturbances are often time-varying and dependent on external conditions. Thus, we introduce an additive term $\mathbf{d}(t)$ to the system dynamics that is independent of the states and inputs:

$$\dot{\mathbf{q}} = f_{\mathcal{F}}(\mathbf{q}, \mathbf{u}) + f_{\text{err}}(\mathbf{q}, \mathbf{u}) + \mathbf{d}(t) \tag{10}$$

Stability and performance in model based control frameworks is dependent on having an accurate dynamics model. Large $f_{\text{err}}(\mathbf{q}, \mathbf{u})$ or $\mathbf{d}(t)$ can compromise the controller and lead to undesirable behavior. To address this challenge, we propose a real-time learning based controller that simultaneously estimates and learns *both* model inaccuracies and environmental disturbances.

III. Preliminaries

In this section, we discuss the preliminaries for MPC. We then formalize a trajectory tracking problem which explicitly addresses environmental disturbances and model inaccuracies. The resulting framework advises the design of the learning-based control framework, described in the subsequent section.

A. Nonlinear Model Predctive Control

In its most general form, MPC addresses an optimal control problem (OCP) by determining an input command $\mathbf{u}$ that minimizes a cost function while satisfying the nominal system dynamics $\dot{\mathbf{q}} = f_{\mathcal{F}}(\mathbf{q}, \mathbf{u})$, and adhering to constraints on both input and state variables over the current and future time steps. The OCP for MPC is formulated as a least-squares optimization, minimizing the deviation of the state trajectory $\mathbf{q}_k$ and control input $\mathbf{u}_k$ from their corresponding reference trajectories $\mathbf{q}_k^*$ and $\mathbf{u}_k^*$, respectively, over the prediction horizon N_c ($t_j \leq t \leq t_{j+N_c}$):

$$\begin{aligned}\min_{\mathbf{q}_k, \mathbf{u}_k} \sum_{k=j}^{j+N_c-1} &(||\mathbf{q}_k - \mathbf{q}_k^*||^2_{W_Q} + ||\mathbf{u}_k - \mathbf{u}_k^*||^2_{W_R} +\\ &||\mathbf{q}_{N_c} - \mathbf{q}^*_{N_c}||^2_{W_{N_Q}})\\ \text{s.t.} \quad &\mathbf{q}_j = \hat{\mathbf{q}}_j\\ \mathbf{q}_{k+1} &= f_{\mathcal{F}}(\mathbf{q}_k, \mathbf{u}_k) + f_{\mathcal{D}}(\mathbf{q}_k, \mathbf{u}_k),\ k = j, \cdots, j+N_c-1\\ \mathbf{q}_{k,\min} &\leq \mathbf{q}_k \leq \mathbf{q}_{k,\max},\ k = j, \cdots, j+N_c\\ \mathbf{u}_{k,\min} &\leq \mathbf{u}_k \leq \mathbf{u}_{k,\max},\ k = j, \cdots, j+N_c-1\end{aligned} \tag{11}$$

where $\mathbf{q}_k \in \mathbb{R}^{n_\mathbf{q}}$ represents the ASV state, $\mathbf{u}_k \in \mathbb{R}^{n_\mathbf{u}}$ denotes the control input, $\hat{\mathbf{q}}_j \in \mathbb{R}^{n_\mathbf{q}}$ is the current state estimate, $\mathbf{q}_k^* \in \mathbb{R}^{n_\mathbf{q}}$ is the reference state trajectory, $\mathbf{u}_k^* \in \mathbb{R}^{n_\mathbf{u}}$ is the reference control input, $\mathbf{q}_{N_c} \in \mathbb{R}^{n_\mathbf{q}}$ represents the terminal state, and $\mathbf{q}^*_{N_c} \in \mathbb{R}^{n_\mathbf{q}}$ represents the terminal state reference. $W_Q \in \mathbb{R}^{n_\mathbf{q} \times n_\mathbf{q}}$, $W_R \in \mathbb{R}^{n_\mathbf{u} \times n_\mathbf{u}}$, and $W_{N_Q} \in \mathbb{R}^{n_\mathbf{q} \times n_\mathbf{q}}$ are the positive definite weight matrices that penalize deviations from the desired states and inputs. Furthermore, $\mathbf{q}_{k,\min}$ and $\mathbf{q}_{k,\max}$ denote the lower and upper bounds of the states, respectively, and $\mathbf{u}_{k,\min}$ and $\mathbf{u}_{k,\max}$ denote the lower and upper bounds of the control input, respectively.

B. Problem Setup

The objective of trajectory tracking requires the incorporation of a control law that steers a mobile platform to converge to a time-parametrized reference trajectory, minimizing the difference between the desired states and the real states. For this purpose, the reference state at each time step k is defined as:

$$\mathbf{q}_k^* = \begin{bmatrix} x_k^* & y_k^* & \psi_k^* & u_k^* & v_k^* & r_k^* \end{bmatrix}, \tag{12}$$

where the superscript * denotes a reference. Thus, the controller should minimize the instantaneous tracking error vector:

$$\mathbf{e}_k = \mathbf{q}_k^* - \mathbf{q}_k \tag{13}$$

The performance of model based controllers is directly tied to the fidelity of the underlying model. However, operating environments for agile platforms often contain time-varying environmental disturbances, in addition to the true dynamics of the platform being difficult to precisely determine. To address these challenges, we propose a MPC framework which interfaces with a residual learning module. This residual model updates in real time with the objective of providing a more accurate dynamics model for the MPC controller, increasing trajectory tracking accuracy under uncertain and dynamic operating conditions.

As previously stated, two common sources of residual error are model inaccuracy $f_{\text{err}}(\mathbf{q}, \mathbf{u})$ and environmental disturbances $\mathbf{d}(t)$ (Eq. 10). We propose augmenting the nominal dynamics $f_{\mathcal{F}}(\mathbf{q}, \mathbf{u})$ with a data driven component:

$$f_{\mathcal{D}}(\mathbf{q}, \mathbf{u}, t) = f_{\text{true}}(\mathbf{q}, \mathbf{u}) + \mathbf{d}(t) \tag{14}$$

where $f_{\mathcal{D}}(\mathbf{q}, \mathbf{u}, t)$ is the residual error of the nominal model. We assume that these errors exclusively affect the acceleration-level dynamics, as the velocity-level terms are derived from a coordinate transformation and presumed to be accurate. This results in the structure of the residual being:

$$f_{\mathcal{D}}(\mathbf{q}, \mathbf{u}, t) = \begin{bmatrix} \mathbf{0}_3 \\ f_{\mathcal{D}_{\dot{\mathbf{v}}}}(\mathbf{q}, \mathbf{u}, t) \end{bmatrix} \tag{15}$$

were $f_{\mathcal{D}_{\dot{\mathbf{v}}}}(\mathbf{q}, \mathbf{u}, t)$ incorporates the second-order data-driven corrections. We augment the system dynamics with the above residual to become:

$$\dot{\mathbf{q}} = f_{\mathcal{F}}(\mathbf{q}, \mathbf{u}) + f_{\mathcal{D}}(\mathbf{q}, \mathbf{u}, t) \tag{16}$$

Equation 16 serves as the dynamic constraint within the MPC controller, enhancing the trajectory tracking performance by systematically compensating for environmental disturbances and model inaccuracy.

IV. Online Residual Model Learning MPC

In this section, we propose an online residual model learning MPC to solve (11) in real time, directly onboard the robot. This method uses a historical sequence of states and control inputs to train a residual NN that approximates the residual model $f_{\mathcal{D}}(\mathbf{q}, \mathbf{u}, t)$. Combined with the nominal model, this approach constructs a highly accurate real-time model for MPC, enabling it to actively choose the optimal control actions that optimize the control performance. The diagram of the residual model learning MPC algorithm is shown in Fig. 4.

A. Residual Neural Network Model

The diagram of the designed NN for residual model is shown in Fig. 5. The input to the NN is structured as a sequence of historical state-input pairs over a time horizon T, preceding the current time step k:

$$\mathbf{h}_k = [\mathbf{x}_k \quad \mathbf{x}_{k-1} \quad \dots \quad \mathbf{x}_{k-T}] \tag{17}$$

where $\mathbf{x}_k = [u \quad v \quad r \quad f_1 \quad f_2 \quad f_3 \quad f_4]$ represents the state-input vector at time step k. The sequence $\mathbf{h}_k$, which includes state-input pairs from $\mathbf{x}_k$ to $\mathbf{x}_{k-T}$, is provided as the input to the NN. The NN architecture consists of two dense feed-forward hidden layers, each comprised of 64 nodes with the tanh() activation function, similar to previous works [16], [24]. In this study, we settled on a threshold of $T = 2.2$s for the historical data input to balance the accuracy and complexity of the NN. We selected this horizon by analyzing the Jacobian of the resulting NN for a longer horizon ($T = 4$s) and selecting an empirical cutoff point for the coefficients as their values decrease as the horizon moves backwards.

We define the loss function as the mean squared error (MSE) between the the NN's prediction ($\boldsymbol{\epsilon}_{k-1}^{\text{pred}}$)and the approximated value ($\boldsymbol{\epsilon}_{k-1}$). Specifically, the loss is computed as:

$$L_i(\theta, \mathbf{h}_{k-1}) = ||\boldsymbol{\epsilon}_{k-1} - \boldsymbol{\epsilon}_{k-1}^{\text{pred}}||^2 \tag{18}$$

In this context, θ is set of the parameters which make up the nodes in the NN. By utilizing a series of historical state-input pairs, the NN is able to capture model inaccuracy and time-varying environmental disturbances. Moreover, we have $\boldsymbol{\epsilon}_{k-1}^{\text{pred}} = \mathbf{y}_{k-1}$, where $\mathbf{y}_{k-1}$ is the output of the residual NN model. $\boldsymbol{\epsilon}_{k-1}$ can be approximated by the time-normalized velocity error [22]:

$$\boldsymbol{\epsilon}_{k-1} = \frac{\hat{\mathbf{v}}_k - \mathbf{v}_k^-}{\Delta t} \tag{19}$$

where $\mathbf{v}_k^-$ is the predicted velocity from the nominal model at the current time step, $\hat{\mathbf{v}}_k$ is the estimated velocity, and Δt is the time step duration.

B. Offline Learning as a Pre-training

We pre-train the residual NN before deploying it in the field to reduce convergence time in real environments and minimize the risk of physical damage to the robots caused by unpredictable behavior. We developed a simulation environment based on the nominal dynamics described in (6), incorporating model-based disturbances such as waves, wind, and currents [30]. More details of the simulation environment are discussed in the next section. Within this simulation, we generate a series of random inputs, producing corresponding nominal and real-state data. This data is processed to extract the residual component (19), which is used to train the residual NN model.

The training was conducted on an Intel NUC 13, running for 3000 epochs using the Adam optimizer [31], a batch of size 32, and a variable learning rate which gradually decreased from 10^{-4} to 10^{-7}. These values were chosen empirically to minimize the training loss. The training is done with `ML_CasADi` [23], which uses `PyTorch` [32] as the base for the NNs.

C. Online Learning in the Field Environment

Once the pre-training is complete, the residual NN model is deployed into the operational environment. As the ASV navigates, the residual NN model is continuously updated by retraining it with one epoch and a small learning rate (3×10^{-7}). This process enables the residual NN model to dynamically adapt to real-time changes in environmental conditions, allowing the MPC to consistently generate optimized and accurate control actions as the ASV operates in aquatic environments. Algorithm 1 provides a detailed overview of the online learning MPC process.

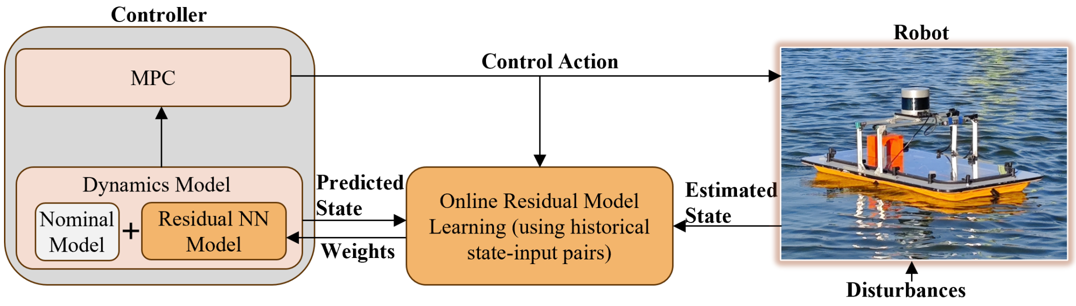

Fig. 4. Diagram of online residual model learning MPC algorithm. At each control iteration, the MPC solver uses both the nominal dynamics and the neural residual to compute the optimal control trajectory for the next N steps. The residual error is then computed by comparing the current state with the predicted state, which in turn is used to online train the residual NN model.

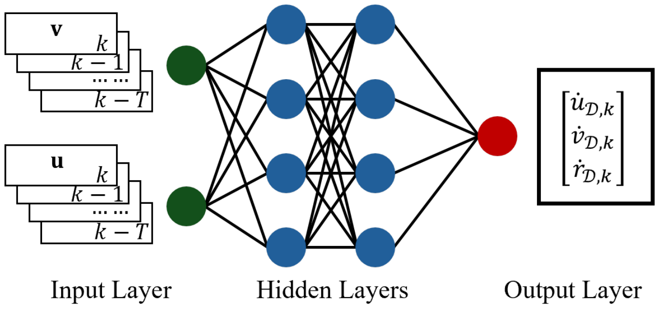
v
k
k − 1
… …
k − T
u
k
k − 1
… …
k − T
$\dot{u}_{\mathcal{D},k}$
$\dot{v}_{\mathcal{D},k}$
$\dot{r}_{\mathcal{D},k}$
Input Layer
Hidden Layers
Output Layer

Fig. 5. The residual NN uses a set of historical velocity-thruster inputs to enhance its capability of predicting the residual error.

Algorithm 1 Online Learning MPC Algorithm

```
Input: Initial state $\mathbf{q}_0$, control input $\mathbf{u}_0$, historical data $\mathbf{h}_k$
Output: Updated control input $\mathbf{u}_k$, updated NN weights
for each time step $k$ do
    Compute the residual acceleration error $\boldsymbol{\epsilon}_{k-1}$ using predicted velocity $\mathbf{v}_k$ and estimated velocity $\hat{\mathbf{v}}_k$ (19)
    Solve the MPC optimization problem to find the optimal control input $\mathbf{u}_k$
    Apply the control input $\mathbf{u}_k$ to the system
    Train the residual NN with the input $\mathbf{h}_{k-1}$ and output $\boldsymbol{\epsilon}_{k-1}$
    Update the NN model in the MPC formulation
    Update the training data $\mathbf{h}_{k-1}$ with the current state-input pair $\mathbf{q}_k, \mathbf{u}_k$
end for
```

D. MPC Implementation

The weighting matrices W_Q, W_R, and W_{N_Q} used in the MPC for the experiments are specified as follows:

$$W_Q = \text{diag}(20, 20, 5, 40, 40, 20), \tag{20}$$
$$W_R = \text{diag}(1, 1, 1, 1), \tag{21}$$
$$W_{N_Q} = \text{diag}(100, 100, 25, 200, 200, 100). \tag{22}$$

A sufficiently large terminal penalty matrix can enhance the stability of the MPC algorithm. The prediction horizon is set to $N_c = 40$ which is equivalent to 4 seconds, with a sampling time of 0.1 seconds. The constraints on the control input $\mathbf{u}$ for the experiments are defined as:

$$\mathbf{u}_{k,\max} = [\mathbf{6}_{4\times1}] \text{ N}, \quad \mathbf{u}_{k,\min} = -[\mathbf{6}_{4\times1}] \text{ N}. \tag{23}$$

The optimization problem is solved using sequential quadratic programming combined with a real-time iteration scheme, implemented through the `acados` package [33]. Quadratic Programming (QP) subproblems are derived using the Gauss-Newton Hessian approximation, with fourth-order Runge-Kutta serving as the integrator. These QPs are efficiently solved using a high-performance interior-point method provided by `HPIPM`, with full condensing and `BLASFEO` (Basic Linear Algebra Subroutines for Embedded Optimization) handling the linear algebra computations. As `acados` integrates with the `CasADi` library, we employ the `ML_CasADi` [23] library to interface with `PyTorch` [32] for NN integration.

V. Simulations and Field Experiments

In this section, we describe the simulations and field experiments extensively conducted to verify the effectiveness of our online residual model learning MPC framework.

A. Simulation Environment

We simulate environmental disturbances, including currents, winds, and waves, using model-based approaches to enhance the fidelity of disturbance representation. Waves are modeled as zero-mean oscillations with drift. An approximate state-space model based on the Pierson-Moskowitz spectrum is given by [34]:

$$\dot{\mathbf{x}}_F = \begin{bmatrix} \dot{x}_{F1} \\ \dot{x}_{F2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_e^2 & -2\lambda_\omega\omega_e \end{bmatrix} \begin{bmatrix} x_{F1} \\ x_{F2} \end{bmatrix} + \begin{bmatrix} 0 \\ K_\omega \end{bmatrix} \omega_{F1}, \tag{24a}$$

$$F_{\text{wave}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_{F1} \\ x_{F2} \end{bmatrix} + d_F, \tag{24b}$$

$$\dot{\mathbf{x}}_N = \begin{bmatrix} \dot{x}_{N1} \\ \dot{x}_{N2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_e^2 & -2\lambda_\omega\omega_e \end{bmatrix} \begin{bmatrix} x_{N1} \\ x_{N2} \end{bmatrix} + \begin{bmatrix} 0 \\ K_\omega \end{bmatrix} \omega_{N1}, \tag{24c}$$

$$N_{\text{wave}} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_{N1} \\ x_{N2} \end{bmatrix} + d_N, \tag{24d}$$

TABLE I

MODEL AND SIMULATION DISTURBANCE PARAMETER VALUES.

$\mathbf{M}_{\text{Sim}}$ $\mathbf{M}_{\text{Model}}$	$\text{diag}(\begin{bmatrix}12 \text{ kg} & 24 \text{ kg} & 1.5 \text{ kg m}^2\end{bmatrix})$ $\text{diag}(\begin{bmatrix}18 \text{ kg} & 36 \text{ kg} & 2.25 \text{ kg m}^2\end{bmatrix})$
$\mathbf{D}_{\text{Sim}}$ $\mathbf{D}_{\text{Model}}$	$\text{diag}(\begin{bmatrix}6 \text{ kg s}^{-1} & 8 \text{ kg s}^{-1} & 1.35 \text{ kg m}^2 \text{ s}^{-1}\end{bmatrix})$ $\text{diag}(\begin{bmatrix}4 \text{ kg s}^{-1} & 5.33 \text{ kg s}^{-1} & 0.90 \text{ kg m}^2 \text{ s}^{-1}\end{bmatrix})$
V_w	4.1 m/s
μ_c ω_c	1.8 1.0 N/s
ω_{F1} d_F ω_{N1} d_N	2.0 N/s 1.0 N 0.2 Nm/s 0.1 Nm

Note: The simulation values listed here are used for evaluation.

where F_{wave} represents the wave force, N_{wave} is the wave moment, and $\dot{d}_F = \omega_{F2}$, $\dot{d}_N = \omega_{N2}$. Variables ω_{F1}, ω_{F2}, ω_{N1}, and ω_{N2} are Gaussian white noise terms, used to introduce variability to the wave model. More details can be found in [27], [34]. The overall wave disturbance is then expressed as:

$$\boldsymbol{\tau}_{\text{wave}} = \begin{bmatrix} X_{\text{wave}} \\ Y_{\text{wave}} \\ N_{\text{wave}} \end{bmatrix} = \begin{bmatrix} F_{\text{wave}} \cos(\beta_{\text{wave}} - \psi) \\ F_{\text{wave}} \sin(\beta_{\text{wave}} - \psi) \\ N_{\text{wave}} \end{bmatrix} \tag{25}$$

where β_{wave} is the angle of wave oscillation and drift.

Wind disturbances are modeled based on wind speed V_w and angle of attack γ_{rw}, which generate forces and moments on the ASV. The wind disturbance vector is given by:

$$\boldsymbol{\tau}_{\text{wind}} = \begin{bmatrix} X_{\text{wind}} \\ Y_{\text{wind}} \\ N_{\text{wind}} \end{bmatrix} = \frac{1}{2}\rho_a V_{rw}^2 \begin{bmatrix} C_X(\gamma_{rw}) A_{FW} \\ C_Y(\gamma_{rw}) A_{LW} \\ C_N(\gamma_{rw}) A_{LW} L_{OA} \end{bmatrix} \tag{26}$$

where ρ_a is the air density, and C_X, C_Y, and C_N are the drag coefficients for different axes, A_{FW} and A_{LW} are the cross-sectional areas, and L_{OA} is the distance to the center.

The relative wind speed V_{rw} is given by:

$$\mathbf{v}_w = \begin{bmatrix} u_w \\ v_w \\ 0 \end{bmatrix} = \begin{bmatrix} V_w \cos(\beta_{\text{wind}} - \psi) \\ V_w \sin(\beta_{\text{wind}} - \psi) \\ 0 \end{bmatrix}, \tag{27}$$

$$\mathbf{v}_{rw} = \mathbf{v}_w - \mathbf{v}, \tag{28}$$

$$V_{rw} = ||\mathbf{v}_{rw}||, \tag{29}$$

where β_{wind} is the wind direction, and $\mathbf{v}_{rw}$ is the relative wind speed vector. More details can be found in [27], [34].

Finally, currents are modeled using a Gauss-Markov process, assuming irrotational currents [34]:

$$\dot{V}_c + \mu_c V_c = \omega_c, \tag{30}$$

where V_c is the current velocity along the direction β_c, ω_c is a Gaussian white noise process, and $\mu_c > 0$ is a parameter shaping the current. The total current velocity $\mathbf{v}_c$ is [27]:

$$\mathbf{v}_c = \begin{bmatrix} V_c \cos(\beta_c - \psi) \\ V_c \sin(\beta_c - \psi) \\ 0 \end{bmatrix}. \tag{31}$$

In the simulation environment, we generated various combinations of disturbances with different intensities to create the offline training dataset for the residual NN. These disturbance combinations were selected based on historical environmental disturbances data [35]. Furthermore, we used values that differ from Table I to better analyze the performance of our algorithm. We used both purely random and random walk variables as the input to the simulated roboat:

$$\begin{aligned} \mathbf{u}_{\text{Random}} &= \mathbf{w}_{\mathbf{u}} \\ \mathbf{u}_{\text{Walk}} &= \mathbf{B}^{\dagger} \int \mathbf{w}_{\tau} - \gamma \mathbf{B} \mathbf{u}_{\text{walk}} dt \end{aligned} \tag{32}$$

where $\mathbf{w}_{\mathbf{u}} \in \mathbb{R}^{4\times1}$, $\mathbf{w}_{\tau} \in \mathbb{R}^{3\times1}$ are the random variables, $\mathbf{B}^{\dagger}$ is the psuedo-inverse of the control matrix $\mathbf{B}$, and γ is a small decay constant to keep the inputs within a typical operational range.

B. Model Inaccuracy Error

To simulate model inaccuracy, we vary the model parameters between the simulation environment and the controller, as detailed by Table I. In our case, we apply a constant scale factor α_1, α_2 to both the mass and drag terms, respectively, such that the resulting controller parameter values are:

$$\begin{aligned} \mathbf{M}_{\text{Model}} &= \alpha_1 \mathbf{M}_{\text{Sim}} \\ \mathbf{D}_{\text{Model}} &= \alpha_2 \mathbf{D}_{\text{Sim}} \end{aligned} \tag{33}$$

We can characterize the resulting model discrepancy error along the second order terms as:

$$\begin{aligned} f_{\text{err}}(\mathbf{q}, \mathbf{u}) = (1 - \alpha_1)\mathbf{M}_{\text{Sim}}^{-1}\mathbf{B}\mathbf{u} - (1 - \alpha_1^2)\mathbf{M}_{\text{Sim}}^{-1}\mathbf{C}\mathbf{v} \\ - (1 - \alpha_1\alpha_2)\mathbf{M}_{\text{Sim}}^{-1}\mathbf{D}\mathbf{v} \end{aligned} \tag{34}$$

Combining Equations (26), (31), and (34) yields an analytical solution for the disturbance the data-driven residual approximates.

C. Simulation Experiments

To validate our algorithm, we compare the trajectory tracking performance across two approaches: the (i) **Conventional MPC** with the nominal model and (ii) **Online Residual Model Learning MPC**, both tested in a simulated environment with environmental disturbances. Each MPC formulation is tasked with following a sinusoidal trajectory at a constant translational velocity of 0.2 m/s. The parameter values of the disturbances can be found in Table I. The values were chosen to be similar to the values in the field environment (Lake Mendota in Madison, WI, USA).

Figure 6 demonstrates the trajectories and trajectory errors over time for both MPC formulations, with the controller switching from the conventional MPC to MPC with the online residual learning after 90 seconds. The MPC with online learning dynamics reduces the distance error by up to 70.5% and the heading error by 4.29% on average, excluding the time take to converge to the trajectory, demonstrating the capability of the learned residual dynamics to account for both the model inaccuracy and environmental disturbances.

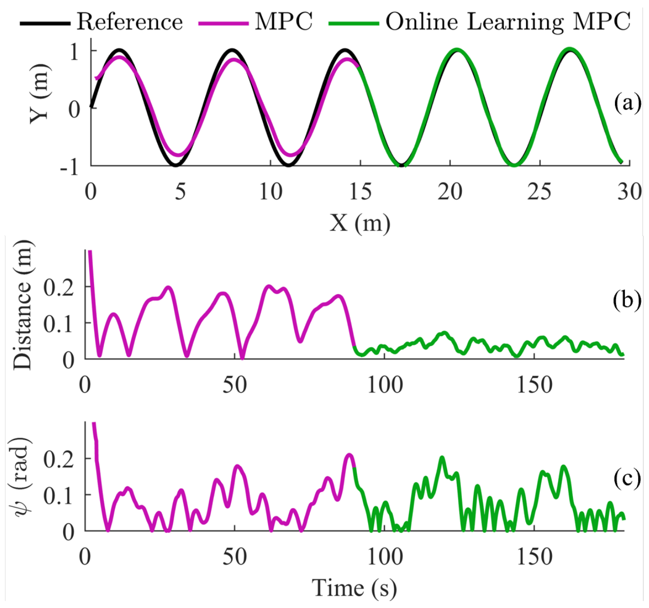

Fig. 6. Comparison between the tracking performance of the nominal MPC and the online residual model learning MPC in simulation. (a) Following a sine curve where the underlying controller changes from nominal MPC to online learning MPC. The two controller configurations run for 90 seconds each. (b) The euclidean distance tracking error. (c) The heading error.

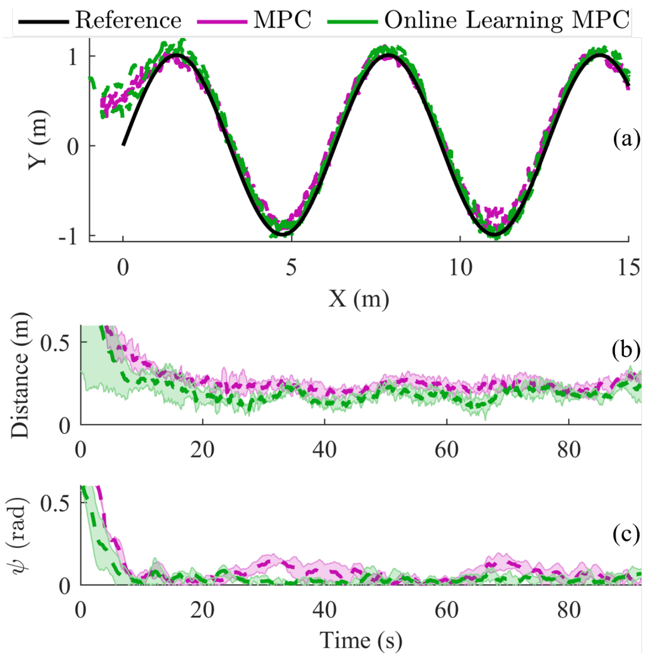

Fig. 7. Comparison between the tracking performance of the nominal MPC and the online residual model learning MPC on Lake Mendota. Data from multiple runs are aggregated and plotted as mean-standard deviation pairs. (a) Trajectories. (b) The euclidean distance tracking error. (c) The heading error.

D. Physical Experiments in the Field

The hardware described in Section II-A was used to validate the algorithm's performance in a real-world setting. Tests were conducted on Lake Mendota, launching from the University of Wisconsin Madison Hoofers Club's pier. The ASV was tasked with following a sinusoidal trajectory at a constant velocity of 0.2 m/s, with a random starting offset of up to 0.7 m from the reference trajectory.

We performed three experimental runs for each MPC configuration (Conventional MPC and Online Residual Model Learning MPC). The localization algorithm, LIO-SAM [28], utilized data from the LiDAR and IMU to estimate the ASV's position $\boldsymbol{\eta}$ and velocity $\mathbf{v}$. All algorithms were executed locally on an Intel NUC with a data update rate of 10 Hz.

Figure 7 shows the trajectories followed by the ASV under different control schemes. In each case, the system converged to the desired trajectory within approximately 10 seconds from the initial offset. The ASV with online learning MPC reduced the trajectory deviation errors by an average of 25.3% for the distance metric and 45.2% for heading angle error compared to the baseline MPC. These results highlight that online learning MPC algorithm significantly improve the tracking performance by learning the unmodeled dynamics and disturbances in real time.

Moreover, Figure 8 shows the aggregated control effort of both algorithms. Comparing the root mean square of the forces:

$$F_{\text{RMS}} = \sqrt{\frac{1}{T}\sum_{i=1}^{T} f_{1,i}^2 + f_{2,i}^2 + f_{3,i}^2 + f_{4,i}^2} \tag{35}$$

reveals that the online residual model learning MPC reduces the overal control effort by 3.24% compared to the baseline MPC. This demonstrates that the more accurate dynamics model provided by the online residual component improves tracking performance for roughly similar control efforts.

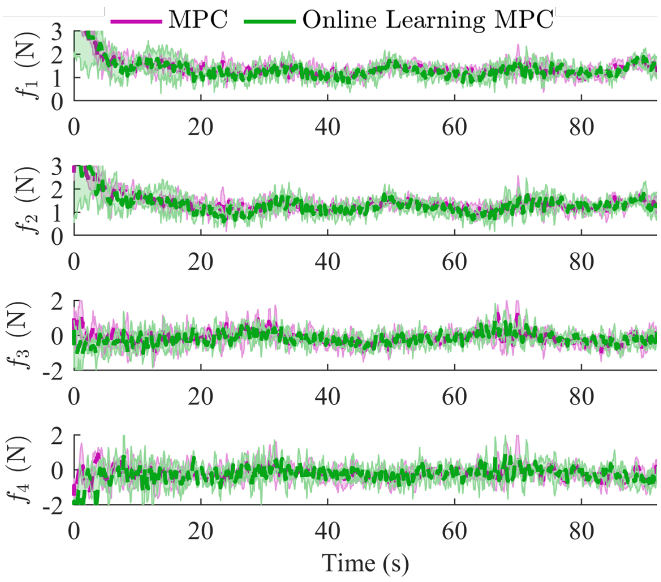

Fig. 8. Comparison between the thruster forces of the different MPC model configurations. Each subplot from top to bottom represents a different thruster. Data from multiple runs are aggregated and plotted as mean-standard deviation pairs.

VI. Conclusion

In this paper, we demonstrate our proposed online residual model learning framework to enhance Model Predictive Control (MPC) for aquatic robots operating in dynamic environments. This approach leverages approximate models to learn complex, unmodeled dynamics and environmental disturbances. We further integrated this online residual model learning into a nonlinear MPC, allowing the controller to actively select optimal control actions that improve overall performance. Our simulations and real-world experiments with an ASV demonstrate significant performance gains compared to conventional MPC methods. Future work will focus on addressing obstacle avoidance within the MPC framework and incorporating Control Barrier Functions (CBF) to ensure safety guarantees during the control process.

References

[1] Y. Qiao, J. Yin, W. Wang, F. Duarte, J. Yang, and C. Ratti, "Survey of deep learning for autonomous surface vehicles in marine environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3678–3701, 2023.

[2] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[3] R. R. Murphy, E. Steimle, C. Griffin, C. Cullins, M. Hall, and K. Pratt, "Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane wilma," *Journal of Field Robotics*, vol. 25, no. 3, pp. 164–180, 2008.

[4] Y. Qiao, J. Yin, W. Wang, F. Duarte, J. Yang, and C. Ratti, "Survey of deep learning for autonomous surface vehicles in marine environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 3678–3701, 2023.

[5] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[6] R. R. Murphy, E. Steimle, C. Griffin, C. Cullins, M. Hall, and K. Pratt, "Cooperative use of unmanned sea surface and micro aerial vehicles at hurricane wilma," *Journal of Field Robotics*, vol. 25, no. 3, pp. 164–180, 2008.

[7] M. Lindemuth, R. Murphy, E. Steimle, W. Armitage, K. Dreger, T. Elliot, M. Hall, D. Kalyadin, J. Kramer, M. Palankar, K. Pratt, and C. Griffin, "Sea robot-assisted inspection," *IEEE Robotics Automation Magazine*, vol. 18, no. 2, pp. 96–107, 2011.

[8] X. Bai, B. Li, X. Xu, and Y. Xiao, "A review of current research and advances in unmanned surface vehicles," *Journal of Marine Science and Application*, vol. 21, no. 2, pp. 47–58, 2022.

[9] W. Wang, D. Fernández-Gutiérrez, R. Doornbusch, J. Jordan, T. Shan, P. Leoni, N. Hagemann, J. K. Schiphorst, F. Duarte, C. Ratti, and D. Rus, "Roboat III: An autonomous surface vessel for urban transportation," *Journal of Field Robotics*, vol. 40, no. 8, pp. 1996–2009, 2023.

[10] W. Wang, X. Cao, A. Gonzalez-Garcia, L. Yin, N. Hagemann, Y. Qiao, C. Ratti, and D. Rus, "Deep reinforcement learning based tracking control of an autonomous surface vessel in natural waters," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3109–3115, 2023.

[11] A. Gonzalez-Garcia and H. Castañeda, "Adaptive integral terminal super-twisting with finite-time convergence for an unmanned surface vehicle under disturbances," *International Journal of Robust and Nonlinear Control*, vol. 32, no. 18, pp. 10271–10291, 2022.

[12] S. Souissi and M. Boukattaya, "Time-varying nonsingular terminal sliding mode control of autonomous surface vehicle with predefined convergence time," *Ocean Engineering*, vol. 263, p. 112264, 2022.

[13] C. Zhang and S. Yu, "Disturbance observer-based prescribed performance super-twisting sliding mode control for autonomous surface vessels," *ISA transactions*, vol. 135, pp. 13–22, 2023.

[14] Y. Weng and N. Wang, "Data-driven robust backstepping control of unmanned surface vehicles," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 9, pp. 3624–3638, 2020.

[15] N. Wang, Y. Gao, and X. Zhang, "Data-driven performance-prescribed reinforcement learning control of an unmanned surface vehicle," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5456–5467, 2021.

[16] N. A. Spielberg, M. Brown, and J. C. Gerdes, "Neural network model predictive motion control applied to automated driving with unknown friction," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 5, pp. 1934–1945, 2022.

[17] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[18] N. A. Spielberg, M. Brown, N. R. Kapania, J. C. Kegelman, and J. C. Gerdes, "Neural network vehicle models for high-performance automated driving," *Science Robotics*, vol. 4, no. 28, p. eaaw1975, 2019.

[19] L. Bauersfeld*, E. Kaufmann*, P. Foehn, S. Sun, and D. Scaramuzza, "Neurobem: Hybrid aerodynamic quadrotor model," in *Robotics: Science and Systems XVII*, RSS2021, Robotics: Science and Systems Foundation, July 2021.

[20] A. Saviolo, G. Li, and G. Loianno, "Physics-inspired temporal learning of quadrotor dynamics for accurate model predictive trajectory tracking," *IEEE Robotics and Automation Letters*, vol. 7, p. 10256–10263, Oct. 2022.

[21] S. Bansal, A. K. Akametalu, F. J. Jiang, F. Laine, and C. J. Tomlin, "Learning quadrotor dynamics using neural network for flight control," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4653–4660, 2016.

[22] G. Torrente, E. Kaufmann, P. Foehn, and D. Scaramuzza, "Data-driven mpc for quadrotors," 2021.

[23] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-time neural mpc: Deep learning model predictive control for quadrotors and agile robotic platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, 2023.

[24] A. Saviolo, J. Frey, A. Rathod, M. Diehl, and G. Loianno, "Active learning of discrete-time dynamics for uncertainty-aware model predictive control," *IEEE Transactions on Robotics*, vol. 40, p. 1273–1291, 2024.

[25] T. Wang, R. Skulstad, M. Kanazawa, G. Li, and H. Zhang, "Learning nonlinear dynamics of ocean surface vessel with multistep constraints," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 9, pp. 10847–10856, 2024.

[26] W. Wang, X. Cao, A. Gonzalez-Garcia, L. Yin, N. Hagemann, Y. Qiao, C. Ratti, and D. Rus, "Deep reinforcement learning based tracking control of an autonomous surface vessel in natural waters," 2023.

[27] W. Wang, W. Xiao, A. Gonzalez-Garcia, J. Swevers, C. Ratti, and D. Rus, "Robust model predictive control with control barrier functions for autonomous surface vessels," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6089–6095, 2024.

[28] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5135–5142, IEEE, 2020.

[29] T. Fossen, "Guidance and control of ocean vehicles," *John Wiley and Sons Ltd google schola*, vol. 2, pp. 3–27, 1994.

[30] A. Gonzalez-Garcia, H. Castañeda, and L. Garrido, "Usv path-following control based on deep reinforcement learning and adaptive control," in *Global Oceans 2020: Singapore – U.S. Gulf Coast*, pp. 1–7, 2020.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[32] 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," in *NIPS-W*, 2017.

[33] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[34] T. I. Fossen, "Handbook of marine craft hydrodynamics and motion control," *John Willy & Sons Ltd*, 2011.

[35] NOAA, "National weather service," 2024.