Quantum Machine Learning-based Test Oracle for Autonomous Mobile Robots

1st Xinyi Wang
Simula Research Laboratory and University of Oslo
Oslo, Norway

2nd Qinghua Xu
Lero Research Centre and University of Limerick
Limerick, Ireland

3th Paolo Arcaini
National Institute of Informatics
Tokyo, Japan

4rd Shaukat Ali
Simula Research Laboratory and Oslo Metropolitan University
Oslo, Norway

5th Thomas Peyrucain
PAL Robotics
Barcelona , Spain

***Abstract*—Robots are increasingly becoming part of our daily lives, interacting with both the environment and humans to perform their tasks. The software of such robots often undergoes upgrades, for example, to add new functionalities, fix bugs, or delete obsolete functionalities. As a result, regression testing of robot software becomes necessary. However, determining the expected correct behavior of robots (i.e., a test oracle) is challenging due to the potentially unknown environments in which the robots must operate. To address this challenge, machine learning (ML)-based test oracles present a viable solution. This paper reports on the development of a test oracle to support regression testing of autonomous mobile robots built by PAL Robotics (Spain), using quantum machine learning (QML), which enables faster training and the construction of more precise test oracles. Specifically, we propose a hybrid framework, `QuReBot`, that combines both quantum reservoir computing (QRC) and a simple neural network, inspired by residual connection, to predict the expected behavior of a robot. Results show that QRC alone fails to converge in our case, yielding high prediction error. In contrast, `QuReBot` converges and achieves 15% reduction of prediction error compared to the classical neural network baseline. Finally, we further examine `QuReBot` under different configurations and offer practical guidance on optimal settings to support future robot software testing.**

***Index Terms*—Autonomous Mobile Robots, regression testing, ML-based test oracle, quantum machine learning.**

I. Introduction

PAL Robotics [1] is a European company developing robots to improve daily life and solve practical problems. Their robots support domestic and industrial tasks using self- and environmental awareness for safe autonomous navigation with built-in path planners. This paper focuses on *Autonomous Mobile Robots* (AMRs), which are widely deployed in service environments such as offices, where they operate in cooperation with humans. During navigation with data collected from onboard sensors, AMRs frequently interact with the environment, including humans and other objects; this can result in failures if AMRs are not thoroughly tested.

As with other software systems, robot software continuously evolves through updates and enhancements to improve performance. These changes require regression testing to ensure that existing, updated, and newly introduced functions all operate correctly. However, the oracle problem remains a challenge in regression testing, as obtaining ground truth is often difficult [2], [3], [4]. In some cases, it can even be very costly or even impossible, especially when additional equipment and complex setup are required [5]. In addition, simulating robots to build ground truth costs substantial time and resources. To this end, machine learning (ML)-based test oracles have been employed in the literature [6], [3], [4] as a viable solution to learn expected behavior from historical data and predict correct expected behavior during testing.

This work develops an ML-based test oracle to support AMR regression testing. It specifically targets the navigation software that enables AMRs to follow an optimal path to their destination while avoiding collisions with humans or other obstacles. Traditional ML models require training on substantial amount of data, so necessitating significant computational resources. In contrast, *reservoir computing* is a computational framework that uses the natural dynamics of randomly connected neural networks to process information, with its main advantage being minimal training requirements [7]. The neural networks used in reservoir computing are typically *recurrent neural networks* (RNNs), which are well-suited for capturing temporal dependencies in sequential data [8].

Recent progress in *quantum computing* (QC) has triggered interest in *quantum reservoir computing* (QRC) [9], which replaces the RNN-based reservoir with a QC system. By combining QC's computational power with the minimal training capability of reservoir computing, QRC offers the potential for highly efficient data processing. In QC, classical data is

The work is supported by the Qu-Test project (Project #299827) funded by the Research Council of Norway and Simula's internal strategic project on quantum software engineering. X. Wang is supported by Simula's internal strategic project on quantum software engineering. S. Ali is supported by Oslo Metropolitan University's Quantum Hub. This work is also partly supported by the RoboSAPIENS project funded by the European Commission's Horizon Europe programme under grant agreement number 101133807. P. Arcaini is supported by the ASPIRE grant No. JPMJAP2301, JST. Q. Xu is supported by Research lreland grant 13/RC/209.

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00306

mapped into high-dimensional Hilbert spaces, enabling the model to better capture temporal patterns and make accurate predictions, which, in our context, means a more accurate test oracle learned with minimal training.

Thus, in this work, we propose a QRC-based model as a test oracle for the navigation software of AMRs developed by *PAL robotics*. Specifically, we train this model to perform next state prediction of the AMRs based on a sequence of historical robot states. The model is trained on data from a stable software version and later used to test the behavior of updated versions, ensuring existing functionality remains unaffected.

Although QRC has shown promise in various domains, such as trajectory prediction, quantum chemistry, and gene regulatory networks [10], [11], [12], [13], [14], [15], most existing QRC approaches are still limited to handling univariate time series data. In contrast, multivariate time series data (e.g., AMR state data), despite extensively-studied in traditional ML, remains underexplored in QRC. This gap motivates us to combine QRC with traditional ML approaches to effectively and efficiently tackle complex multivariate time series data.

To that end, we propose a hybrid framework, `QuReBot`, which is inspired by the *residual connection* technique in deep learning [16]. It has been widely applied in several deep learning algorithms, such as ResNet [17], and Transformer architectures [18] to reduce the influence of vanishing gradient by passing the input directly to the output layer, skipping the intermediate layers. In our context, `QuReBot` uses QRC for time series data prediction and a residual connection to link the input to the final output layer through a relatively simple non-linear transformation. For each input data, `QuReBot` dynamically calculates a weight parameter to control the contribution of the QRC branch and the residual connection branch (the "shortcut") based on its context.

We evaluate the performance of `QuReBot` with the AMR developed by *PAL robotics*.[1] Results show that QRC alone fails to converge and results in high prediction error. `QuReBot` converges successfully and achieves a 15% lower MSE compared to the classical baseline. Finally, we compare `QuReBot` across various configurations and provide practical recommendations for optimal settings for future robot development.

II. Background

A. Quantum Computing

The quantum computing paradigm leverages quantum mechanical principles to solve certain problems that are difficult for classical computers. It performs computations using *quantum bits* (i.e., *qubits*), instead of bits, to process information. Unlike a bit, which can only be either 0 or 1, a qubit can exist in a *superposition* of state $|0\rangle$ and $|1\rangle$ at the same time, until it is *measured* and collapses into one of the basis states with different probabilities. The *quantum state* (i.e., also called as *pure state*) of a single qubit can be represented as

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}, \quad |\alpha_0|^2 + |\alpha_1|^2 = 1 \qquad (1)$$

[1]All experiment results and code for replication can be found at [19].

TABLE I: Descriptions of commonly used quantum gates

Gate	Description
Hardamard *(H)*	Placing a qubit into an equal superposition, i.e., there is a 50% probability of being measured in state either $\|0\rangle$ or $\|1\rangle$.
Pauli-Z (Z)	Rotating a qubit around the z-axis with π radians.
Pauli-X (X)	Rotating a qubit around the x-axis with π radians.
Pauli-Y (Y)	Rotating a qubit around the y-axis with π radians.
$RX(\theta)$	Rotating a qubit around the x-axis with θ radians.
$RY(\theta)$	Rotating a qubit around the y-axis with θ radians.
$RZ(\theta)$	Rotating a qubit around the z-axis with θ radians.
Controlled-X (CNOT/CX)	A two-qubit gate with a control and a target qubit. If the control qubit is $\|1\rangle$, the target qubit rotates around the x-axis with π radians.
Controlled-Z (CZ)	A two-qubit gate with a control and a target qubit. If the control qubit is $\|1\rangle$, the target qubit rotates around the z-axis with π radians.
Two qubit rotation gates *RXX/RYY/RZZ*	A two-qubit gate rotating the two qubits around z axes simultaneously with θ radians.

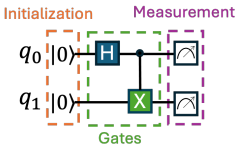


Fig. 1: Example of quantum circuit

where the probability of the qubit being $|0\rangle$ is $|\alpha_0|^2$ while that of being $|1\rangle$ is $|\alpha_1|^2$. The quantum state of a multi-qubit system with N qubits can be represented by

$$|\psi\rangle = \sum_{i=0}^{2^N-1} \alpha_i |i\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{2^N-1} \end{bmatrix}, \quad \sum_{i=0}^{2^N-1} |\alpha_i|^2 = 1 \qquad (2)$$

where $|i\rangle$ denotes the i-th computational basis state in binary.

Another important concept in quantum computing is *entanglement*, where two or more qubits share a single joint quantum state and depend on each other. For example, in the entangled two-qubit quantum state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big) \qquad (3)$$

the system collapses into either $|00\rangle$ and $|11\rangle$ with equal probability when they are measured.

Gate-based quantum computing is a widely used model to perform computations by applying a sequence of *gates* (i.e., *unitary operators*) to a *quantum circuit*, which is composed of several qubits. Table I shows some commonly used gates. In a quantum circuit, each qubit is first initialized into a classical basis state (e.g., $|0\rangle$), then quantum gates are applied to manipulate their quantum states (e.g., creating superposition and entanglement) to perform the desired computation. For example, Fig. 1 shows a two-qubit circuit, where the two qubits are first initialized in the $|00\rangle$ state. An H gate is applied on the q_0, placing it into superposition. Next, a CX gate entangles the two qubits. Finally, the two qubits are measured, causing the system to collapse into a classical basis state.

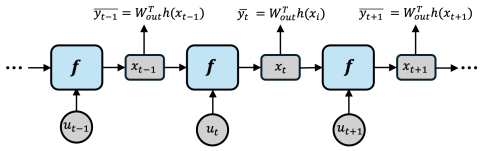


Fig. 2: Framework of Reservoir Computing (RC)

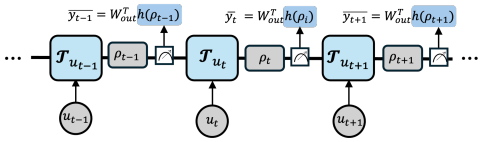


Fig. 3: Framework of Quantum Reservoir Computing (QRC)

B. Quantum Reservoir Computing (QRC)

Classical Reservoir Computing. *Reservoir Computing* (RC) is a computational framework for temporal information processing. RC leverages a fixed, non-linear system called *reservoir* to map the input sequence into higher-dimensional spaces, similar to neural network hidden layers. Then, a *readout* layer is employed to produce the desired output sequence [8]. Unlike neural networks where all weights are trained, RC keeps the *reservoir* fixed and only trains the *readout* layer. Consequently, RC is computationally more efficient and requires less data for training compared to neural networks.

Specifically, given an input sequence $\{u_t\}_{t=0}^{M-1}$, RC aims to predict the corresponding output sequence $\{y_t\}_{t=0}^{M-1}$. Fig. 2 shows the general framework of RC. To encode the temporal information of sequences, the *reservoir* in RC maintains an internal state x_t, referred to as the *reservoir state* hereafter. The update rule of x_t at timestep t is given by

$$x_t = f(x_{t-1}, u_t) \tag{4}$$

where f is a non-linear, fixed function that represents the temporal evolution of the reservoir state. Similar to the recurrent unit of RNN, f incorporates both information from the previous state x_{t-1} and the current input u_t to predict the current state x_t. Subsequently, RC maps x_t to the output space through a *readout* layer, which is essentially a linear transformation defined as

$$\bar{y_t} = W_{out} h(x_t) \tag{5}$$

where W_{out} is a weight matrix trained to reduce the difference between the predicted output $\bar{y_t}$ and the target output y_t. h is a function that observes signals from the state x_t.

Quantum Reservoir Computing. *Quantum Reservoir Computing* (QRC) extends the classical reservoir in RC with a quantum system as depicted in Fig. 3. Following the general framework of RC, we inject values of input sequence $\{u_t\}_{t=0}^{M-1}$ at successive timesteps into the quantum circuit to calculate corresponding internal *reservoir states* (i.e., quantum states in this context), which are then used to predict the output values through a *readout* layer. As discussed in Sect. II-A, the (pure) quantum state of a quantum system with N qubits

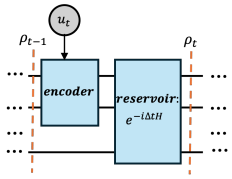


Fig. 4: $\mathcal{T}_{u_t}$ Circuit

is represented as a 2^N-dimensional vector $|\psi\rangle$. Since a QRC system can be considered as a statistical mixture of pure states, we represent it as a $(2^N \times 2^N)$-dimensional density matrix ρ. The time evolution of the quantum system under an input u_t is described by

$$\rho_t = \mathcal{T}_{u_t}(\rho_{t-1}) = e^{-i\Delta t H} \rho(t-1) e^{i\Delta t H} \tag{6}$$

where ρ_t represents the internal reservoir state at time t. $\mathcal{T}_{u_t}$ represents the evolution process in Fig. 4 where the input value u_t is first encoded and applied to the quantum system, which evolves for time Δt under the unitary operator $e^{-iH\Delta t}$ defined by the system Hamiltonian H.[2] Notably, H is configurable, and it decides the system dynamics. During evolution, the injected information will spread through the whole system. The function h in Eq. 5 is implemented by measurements on all qubits. We measure the system and calculate the expectation values on each qubit, obtaining

$$h(\rho_t) = [\langle Z_0 \rangle, \ldots, \langle Z_{N-1} \rangle] \tag{7}$$

After getting sufficient pairs of $\big(h(\rho_t), y_t\big)$, we can train a simple linear regression model to determine the parameters of W_{out} by minimizing the mean squared error between predicted outputs $\bar{y_t}$ and target outputs y_t:

$$\min \sum_{t=0}^{M} (\bar{y_t} - y_t)^2 \tag{8}$$

However, during implementation, the computational cost of this process is considerable. First, due to the probabilistic nature of the quantum system, multiple repetitions are necessary to approximate expectation values accurately. Second, the above framework requires measurements on all qubits between every two timesteps to get $h(\rho_t)$, which is destructive as it causes the whole system to collapse and disrupt the system dynamics. Consequently, the entire process must be repeated from the beginning for each timestep. For example, as illustrated in Fig. 5, to obtain $h(\rho_t)$ at timestep t, the system evolves from u_0 to u_t before measurement. Then, to get $h(\rho_{t+1})$, the system is reset and evolved again from u_0 to u_{t+1}. This *restarting protocol* is highly inefficient and requires substantial resources.

To address this issue, a *rewinding protocol* is proposed based on the *echo state property* in RC, which illustrates that, after sufficient evolution, reservoir state becomes independent of its

[2]Hamiltonian defines the energy of a quantum system, and is mathematically expressed as a hermitian matrix.

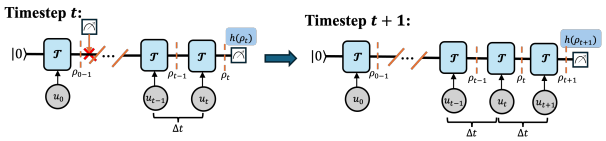


Fig. 5: Restarting Protocol

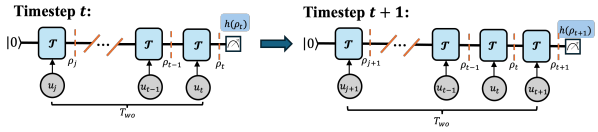


Fig. 6: Rewinding Protocol

initial state, depending only on recent inputs [20]. Accordingly, in implementation, we introduce the washout time T_{wo} as the timesteps necessary for RC to "forget" the initial state. Thus, as shown in Fig. 6, instead of restarting from t_0 every time, the system only needs to reconstruct the last T_{wo} steps, which significantly reduces the required resources.

III. Industrial Context

PAL Robotics is a Spanish company with world-leading expertise in developing robots for deployment in the service industry (e.g., warehouses and retail). They support businesses and are also actively involved in research and development projects with research organizations in this domain. Examples of robots built by PAL include legged robots for research purposes, robots designed to assist in retail environments (e.g., for stocking goods), and Autonomous Mobile Robots (AMRs), which are widely used in various logistics applications [1].

Although the work presented in this paper can be applied to various types of robots produced by PAL, we here focus on AMRs, specifically the TIAGo OMNI robot [21]. This robot is widely used in both research and industrial settings for its accurate navigation and advanced sensing capabilities. It is commonly deployed in service environments such as office spaces and hospitals, where it operates alongside humans. Consequently, it must navigate in a way that ensures human safety, while effectively completing its assigned tasks. The robot employs two navigation algorithms: the first one is designed to determine an optimal route to its destination, while the second one focuses on detecting obstacles and dynamically adapting the robot's path. This adaptive behavior is supported by real-time data from multiple onboard sensors that continuously monitor the surrounding environment.

Like any other software system, a robot's software evolves, requiring regression testing to ensure that the updated software continues to meet both its functional and non-functional requirements. Such testing requires a test oracle to determine whether the robot's behavior remains correct, i.e., navigation remains optimal and safe in our context. However, building a test oracle is challenging, as the ground truth is often complicated to obtain. To address this challenge, machine learning-based test oracles are often employed [6], [3], [4], typically trained on historical data. In line with this approach, we investigate the use of QML to develop a test oracle that supports the regression testing of the TIAGo OMNI robot. Specifically, we design a hybrid framework that integrates neural networks and QRC. The quantum dynamics within QRC enhances the neural networks' ability to learn more precise patterns in data compared to classical machine learning. As a result, the hybrid model can model the navigation behavior of the robot more precisely and accurately, thereby serving as a precise test oracle to support regression testing.

IV. Approach

We introduce the overview of `QuReBot`. First, we describe how we integrate QC with a classical neural network to predict the next states of a robot based on its previous states. Second, we elaborate on how we design the core of `QuReBot`, i.e., the quantum reservoir circuit.

A. Overview

Fig. 7 shows an overview of the training process in `QuReBot`. For training, we gather navigation data during operation from a stable release of the TIAGo OMNI robot. As shown in Block a of Fig. 7, we consider a sequence of $t+1$ observed historical state $\mathcal{S} = \{\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_t\}$, where each state contains features related to its position, orientation, linear velocity, and angular velocity. Specifically, each state consists of d features as in $\mathbf{s}_j = \{s_j^{(0)}, s_j^{(1)}, \ldots, s_j^{(d-1)}\}$. Considering the rewinding protocol illustrated in Sect. II-B, we select features of the last T states (i.e., washout time) as the input of our model (i.e., $\mathcal{S}_t = \{\mathbf{s}_{t-T+1}, \mathbf{s}_{t-T+2}, \ldots, \mathbf{s}_t\}$). We also define the number of future time steps to predict as horizon K. The output of the model is the position of the robot in the next K time steps $\mathcal{P}_t = \{\mathbf{p}_{t+1}, \mathbf{p}_{t+2}, \ldots, \mathbf{p}_{t+K}\}$ and $\mathbf{p}_j = (p_j^{pos_x}, p_j^{pos_y})$.

`QuReBot` is a QRC-based approach to predict the robot's position in the next K states. Inspired by the success of skip connections in deep neural networks, we adopt a similar strategy in `QuReBot` to enhance prediction effectiveness. In deep learning, skip connections typically link the input directly to the output layer, bypassing intermediate layers. The adoption of skip connections can potentially preserve critical information in the input, mitigating severe context loss in deep neural networks or complex quantum reservoirs. As shown in Fig. 7, our approach contains two branches: the *QRC branch* (i.e., Block b) and the *skip connection branch* (i.e., Block c).

In the QRC branch, to process the input data into quantum circuits, we first normalize each state as $\hat{\mathbf{s}}_j = \{\hat{s}_j^{(0)}, \hat{s}_j^{(1)}, \ldots, \hat{s}_j^{(d-1)}\}$ to fit the requirements of the quantum encoders. The normalized data then undergoes time evolution within the reservoir circuits. After measurement, we get expectation values for each qubit, i.e., a real-valued vector defined as:

$$\mathbf{r}_t = \mathcal{F}_{QRC}(\hat{\mathcal{S}}_t) = \{r_t^{(0)}, r_j^{(1)}, \ldots, r_j^{(D-1)}\} \tag{9}$$

where $\mathcal{F}$ is the QRC process. More details in Sect. IV-B.

The skip connection branch provides a "shortcut" to bypass the complex quantum reservoir, consisting of relatively simple

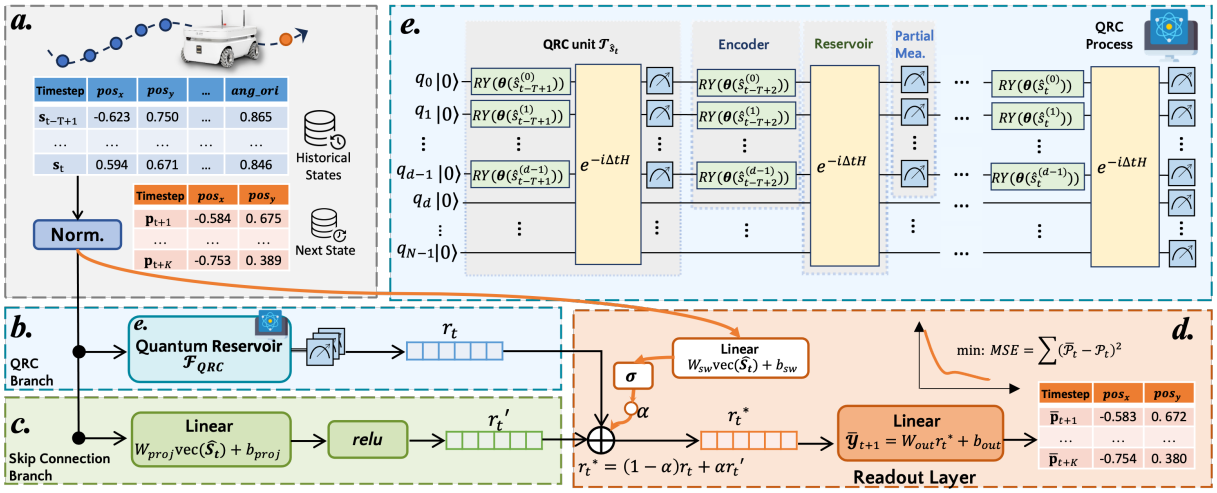


Fig. 7: Overview of QuReBot

transformations. As depicted in Block b of Fig. 7, the normalized state $\hat{s}_j$ is consecutively fed into a linear layer and an activation layer (ReLU), yielding a vector $\mathbf{r}_t'$ with the same dimension as $\mathbf{r}_t$. The transformation equations are as follows

$$\mathbf{r}_t' = W_{proj} \cdot \mathrm{vec}(\hat{\mathcal{S}}_t) + b_{proj}\,, \quad \mathrm{vec}(\hat{\mathcal{S}}_t) \in \mathbb{R}^{Td} \tag{10}$$

$$\mathbf{r}_t' = relu(\mathbf{r}_t') \tag{11}$$

where W_{proj} and b_{proj} are weight matrices.

As shown in Block d of Fig. 7, QuReBot calculates a weighted sum of both outputs from the QRC branch ($\mathbf{r}_t'$) and ($\mathbf{r}_t$). Standard residual connections add $\mathbf{r}_t'$ and $\mathbf{r}_t$ directly, implicitly treating both inputs equally, whereas QuReBot dynamically adjust the influence of each branche based on the characteristics of the input. Specifically, QuReBot computes a scalar value α from the input $\hat{\mathcal{S}}_t$ as

$$\alpha = W_{sw} \cdot \mathrm{vec}(\hat{\mathcal{S}}_t) + b_{sw} \tag{12}$$

Then, a *sigmoid function* σ is applied to map the weight into a value between 0 and 1.

$$\alpha = \sigma(\alpha) \tag{13}$$

With α computed, QuReBot computes the weighted sum $\mathbf{r}_t^*$ of $\mathbf{r}_t'$ and $\mathbf{r}_t$ as in

$$\mathbf{r}_t^* = (1-\alpha)\mathbf{r_t} + \alpha \mathbf{r}_t' \tag{14}$$

Finally, a readout layer takes $\mathbf{r}_t^*$ as input and predicts the target output. Specifically, the readout layer is a linear transformation defined as

$$\bar{\mathcal{P}}_t = W_{out} \cdot \mathbf{r}_t^* + b_{out} \tag{15}$$

$\bar{\mathcal{P}}_t$ represents the predicted sequence of next K time states. The objective of this hybrid model is to minimize the mean squared error between the predicted output $\bar{\mathcal{P}}_t$ and the target output $\mathcal{P}_t$. Suppose M input-output pairs are given, the objective is formally formulated as

$$\min \mathrm{MSE} = \frac{1}{M} \sum_{t=0}^{M-1} (\bar{\mathcal{P}}_t - \mathcal{P}_t)^2 \tag{16}$$

B. Quantum Reservoir Computing Implementation

In this section, we introduce the QRC process $\mathcal{F}_{QRC}$ (i.e., Block e in Fig. 7). The first issue we need to address is to feed the sequence of the input states $\hat{\mathcal{S}}_t$ into the quantum system. In QRC, each state is injected into the quantum circuit at consecutive time steps. At each time step, each normalized feature value $\hat{s}_t^{(j)} \in [0,1]$ is fed to the system by setting the quantum state of one qubit to $|\psi_{\hat{s}_t^{(j)}}\rangle = \sqrt{1-\hat{s}_t^{(j)}}\,|0\rangle + \sqrt{\hat{s}_t^{(j)}}\,|1\rangle$. To set this quantum state, we need to apply an RY gate (see Table I) on the j-qubit with rotation $\theta = 2\arcsin\sqrt{\hat{s}_t^{(j)}}$. As shown in the *Encoder* box in Block e of Fig. 7, we select a subset of qubits to encode the feature values. Each qubit is encoded with one value.

Then, all qubits in the quantum system undergo time evolution through an input-independent unitary operator as the reservoir circuit, which is commonly a variational quantum circuit. The parameters in the reservoir circuit are randomly assigned and fixed throughout the process. Various structures for reservoir circuits have been proposed. Due to the existing hardware restrictions on current quantum computers, a class of hardware-efficient [22] quantum reservoir circuits has been proposed. These circuits are constructed from repeated layers. Each layer includes an optional *rotation layer*, consisting of single-qubit rotation gates, and an *entangling layer*, built from multi-qubit gates. The number of repetitions of these layers determines the circuit *depth*. In this paper, we consider the following four quantum circuits as quantum reservoir circuits.

(1) CNOT Reservoir: It does not include a rotation layer and contains no randomly assigned parameters. It is built only

from CX gates, that are arranged in a circular structure to entangle all the qubits. Fig. 8a shows an example circuit with three qubits in a single repetition.

(2) Rotation Reservoir: It contains a rotation layer and an entangling layer in a single repetition. The rotation layer applies a single-qubit rotation gate on each qubit, randomly selected from the gate set $\{RX,\ RY,\ RZ\}$. Each rotation angle θ_i is randomly sampled from the range $[0, 2\pi)$ and fixed. The entangling layer is constructed by CZ gates arranged in a circular entangling structure to connect all qubits. Fig. 8b shows an example circuit with three qubits in a single repetition.

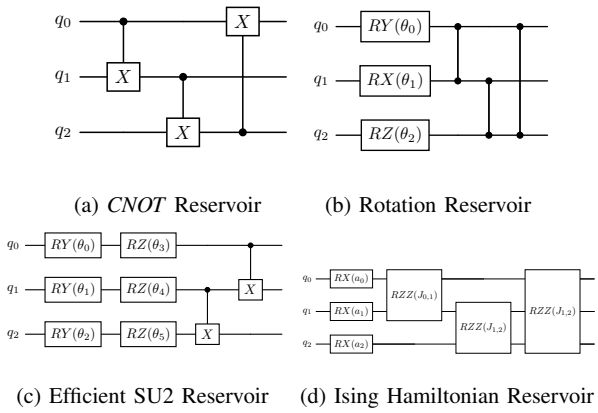
(3) Efficient SU(2) Reservoir: It is based on a widely used ansatz in Qiskit for variational quantum algorithms [23]. Each repetition consists of a rotation layer followed by an entangling layer. In the rotation layer, we apply two single-qubit rotation gates on each qubit, which are RY and RZ gates. The rotation angles θ_i are randomly selected from $[0, 2\pi)$ and remain fixed throughout the process. The entangling layer is constructed by CX gates arranged in a "reverse linear" entangling structure, where every two adjacent qubits are entangled starting from the higher-indexed qubit. Fig. 8c shows an example circuit with three qubits in a single repetition.

(4) Ising Hamiltonian Reservoir: It corresponds to Ising Hamiltonian dynamics:

$$H = \sum_{j=1} a_j X_j + \sum_{j<k} J_{jk} Z_j Z_k \tag{17}$$

where a_j is called the *local field coefficient* specifying how strongly qubit j interacts with the field along the X axis. J_{jk} shows the *coupling coefficient* between qubit j and k, which describes the interaction between the two qubits along Z axis. Thus, in the rotation, we apply an RX gate on each qubit with parameter a_j as the rotation angle. In the entangling layer, RZZ gates are applied to connect qubits in a circular structure, with rotation angle J_{jk}. The values of both a_j and J_{jk} are randomly sampled from the range $[-1, 1]$. Fig. 8d shows an example circuit with three qubits in a single repetition.

In this model, we select a reservoir circuit with a set of fixed parameters to train the model. The entangling layer ensures the inputs injected into the circuit spread throughout the system. As shown in Sect. II-B, each QRC unit $\mathcal{T}_{s_t}$ contains an encoder and a reservoir circuit. Although we apply the rewinding protocol, constructing a QRC circuit of T time steps leads to a long circuit, which is challenging as quantum systems can lose coherence quickly, making them unreliable. To address this, partial measurements (i.e., instead of full measurement) between every two timesteps are performed to balance memory and practicality [24]. Specifically, we measure the qubits where inputs are injected while leaving the remaining *ancilla qubits* unmeasured so they can continue evolving and preserve past input information. The measured qubits will get reset to inject new input.



(a) *CNOT* Reservoir (b) Rotation Reservoir

(c) Efficient SU2 Reservoir (d) Ising Hamiltonian Reservoir

Fig. 8: Quantum Reservoir Circuits

After the system evolves T time steps, we measure all the qubits at Z basis to obtain the final vector $\mathbf{r}_t = \{r_t^{(0)},\ r_j^{(1)},\ \ldots,\ r_j^{(D-1)}\}$, which also includes information collected from partial measurements during previous timesteps.

V. Experiment Design

A. Research Questions

RQ0. *How is the performance of QRC with different quantum reservoirs*?
We conducted a pilot study to evaluate the performance of using only the QRC algorithm for predicting the robot's next state. In addition, we aim to assess the impact of different quantum reservoir circuits on the performance.

RQ1. *How is the performance of* `QuReBot` *compared to the classical baseline model*?
This RQ compares the performance of `QuReBot` and a simple classical neural network model. We also compare the two models under various configurations to evaluate our model's robustness.

RQ2. *What is the performance of* `QuReBot` *across different input features and prediction horizons*?
This RQ evaluates the performance of `QuReBot` under different configurations. Based on the results, we provide practical guidelines of optimal input features and prediction horizons to use for researchers and practitioners using `QuReBot`.

B. Datasets and Features

We employed the open-source PAL Robotic OMNI Base Simulation environment [25] to create the dataset, which integrates ROS 2 and Gazebo to provide a realistic 3D simulation of the TIAGo OMNI Base robot. The robot navigated through a set of waypoints in the PAL Robotics office environment using the ROS 2 Nav2 stack. We recorded a total of 14,236 time steps at a rate of 10 Hz frequency, to train our model. Each time step captured seven features in three categories position (i.e., pos_x, pos_x), orientation (i.e., ori_z, ori_w), and velocity (i.e., vel_x, vel_y, vel_{ang}). To investigate the effect of feature selection on `QuReBot`, we create three feature sets to

train our model: FS_7 (all features), FS_5 (without orientation), and FS_4 (without velocity). Notice that we do not consider the exclusion of position-related features, as they are indispensable for our task, i.e., predicting the next position of the robot.

C. Baselines

We consider two baselines: (1) `QRC-only`: This is the baseline QRC model without enhancements. In contrast to `QuReBot`, this model removes the skip connection branch. The inputs undergo the quantum evolution, and the final outputs are predicted based solely on the measurement of the quantum circuit with a linear regression; (2) `Skip-only`: This is a pure classical model with a simple neural network. It differs from `QuReBot` by removing the QRC branch. The inputs are passed through a linear layer, a *ReLU* activation function, and another linear layer to predict the outputs.

D. Parameter settings

QRC circuit configuration: To reduce the influence of randomness in the reservoir, we fix a single random seed to generate the variational circuits, which are then consistently used across all QRC components. To balance performance and computational cost, we set the depth for the quantum reservoir circuit to 10. Also, we use one extra qubit as the ancilla qubit.

Experiment configuration: For `QRC-only`, we did a pilot study and implemented QRC with all four reservoir circuits described in Sect. IV-B. We configure the model with feature set FS_7 and horizon $K = 1$. For `QuReBot`, we use the Ising Hamiltonian reservoir in the QRC branch. For both `Skip-only` and `QuReBot` models, we implement all three feature sets (i.e., FS_7, FS_5, FS_4) with prediction horizons ranging from 1 to 5.

Training parameters: To fully utilize the dataset, we conduct 4-fold cross-validation by splitting the dataset into four equal subsets based on the time sequence. Each experiment uses one fold for testing and the remaining three for training. Moreover, we select 20% of the training datasets as the validation datasets. The model is trained with mean squared error as the loss function, with a learning rate of 1×10^{-4} and a batch size of 32. Training runs for up to 500 epochs, with early stopping applied if the validation loss does not improve for 10 consecutive epochs. The final model is selected based on the lowest validation loss. To mitigate randomness, each experiment is repeated 10 times. As a result, a total of $5 \times 3 \times 4 \times 10 = 600$ models were trained for both `QuReBot` and `Skip-only`, respectively, for evaluation.

E. Experiment setup

We implement the QRC component of `QuReBot` with the *Quantumreservoirpy* package [26], based on the Qiskit framework. We use the Qiskit Aer simulator to execute quantum circuits. The quantum computing process is run on AMD EPYC Naples 7601 (SMT2) w/2TB of RAM and 4TB of NVMe scratch. `QuReBot` and the baselines are trained on AMD EPYC Milan 7763 64-core w/ 8 qty Nvidia Volta A100/80GB.

F. Evaluation metrics and statistical tests

To evaluate the effectiveness of `QuReBot` and the baselines, we calculate the average *mean squared error* as:

$$\mathcal{L}_{MSE} = \frac{1}{|TS| \cdot K \cdot |\,\mathbf{p}\,|} \sum_{ts=1}^{|TS|} \sum_{j=1}^{K} (\mathbf{p}_{ts,j} - \bar{\mathbf{p}}_{ts,j})^2 \qquad (18)$$

where $|TS|$ represents the size of the test dataset, K represents the horizon in prediction, and $\bar{\mathbf{p}}_{ts,j}$ refers to the predicted vector of the output (i.e., the position).

In RQ0, for each reservoir circuit, we gather $\mathcal{L}_{MSE}$ of `QRC-only` model across 4 folds and 10 repetitions for each reservoir circuit. To assess whether there are significant performance differences among various reservoir circuits, we apply the Kruskal-Wallis test. A p-value less than 0.05 indicates a statistically significant difference among the circuits; otherwise, no significant difference is observed.

In RQ1, to implement `QuReBot` and the `Skip-only` model under various configurations, for each configuration we calculate $\mathcal{L}_{MSE}$ across 4 folds and 10 repetitions. To compare the performance of the two models, we use the Mann-Whitney U test combined with $\hat{A}_{12}$ effective size. If the yielded p-value is less than 0.05, it means that there is a significant difference in performance between the two models. In this case, we calculate the $\hat{A}_{12}$ statistics. If the $\hat{A}_{12}$ value is less than 0.5, it shows that `QuReBot` outperforms the `Skip-only` model, while a value above 0.5 indicates the opposite.

In RQ2, we compare the overall performance of `QuReBot` with different feature sets and horizons respectively. Considering the feature sets, we group `QuReBot` with the same feature set (but varying prediction horizons) and compare these groups. We first use the Friedman test to determine if there are significant differences among all three groups. A p-value less than 0.05 indicates statistically significant differences. Then, we perform pairwise comparisons using the Wilcoxon signed rank test with $\hat{A}_{12}$ effect size. If p-value is smaller than 0.05, it shows there is a significant difference between the pair. Then, the $\hat{A}_{12}$ statistic is calculated to determine the strength of the statistical test result. If the $\hat{A}_{12}$ value is less than 0.5, it shows that `QuReBot` with the first feature set is likely to be better than the second one, and vice versa. The same statistical procedures are applied to compare `QuReBot` models across different prediction horizons.

VI. Results and analyses

A. RQ0: Performance of `QRC-only` with different reservoirs

We evaluate the performance of the `QRC-only` model using the same parameters as in training `QuReBot`. However, during training, validation loss failed to converge within 500 epochs, and predictions deviated significantly from actual values. This may be due to the nature of the input data: robot states recorded at fine-grained time steps show minimal variation between consecutive states. When a highly dynamic quantum reservoir processes such low-variation input, the measured output may not effectively capture subtle transitions. Moreover, the multivariate nature of the time series further

TABLE II: RQ1 – Comparison of `QuReBot` and `Skip-only` model with various feature sets and horizons.

Horizon (K)	FS_7		FS_5		FS_4	
	p-value	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$	p-value	$\hat{A}_{12}$
$K=1$	0.0491	0.3719	0.0017	0.2963	0.0078	0.3269
$K=2$	0.0480	0.3713	0.0037	0.3113	0.0119	0.3362
$K=3$	0.1673	—	0.0007	0.2787	0.0229	0.3519
$K=4$	0.0331	0.3613	0.0004	0.2694	0.0315	0.3600
$K=5$	0.1673	—	0.0006	0.2781	0.0331	0.3613

complicates the task of identifying meaningful patterns. Without additional context, the linear readout layer struggles to learn meaningful patterns to predict output accurately.

This behavior was consistently observed on the `QRC-only` model with all four reservoir circuits. We computed $\mathcal{L}_{MSE}$ using 4-fold cross-validation with 10 repetitions. The average $\mathcal{L}_{MSE}$ over 40 runs is around 15.6 for all four reservoir circuits, indicating similar prediction error across them. Statistical tests show that the p-value is larger than 0.05, indicating that there is no significant difference among those structures.

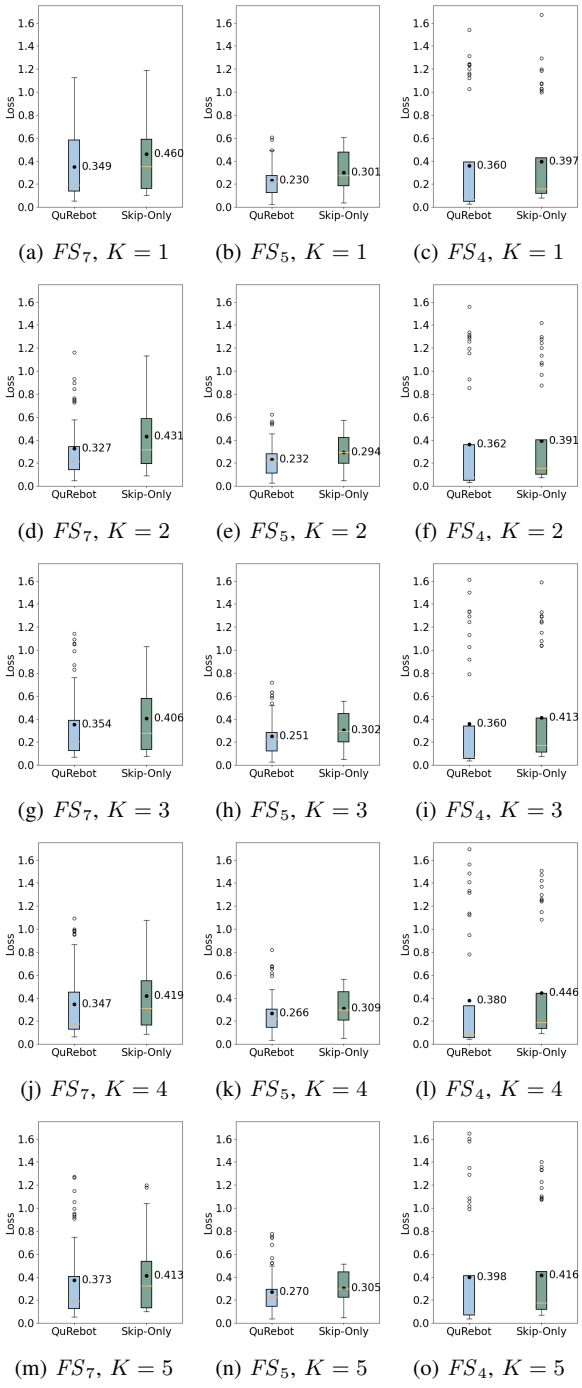
Answer to RQ0: `QRC-only` fails to converge, resulting in high prediction error. In addition, using different reservoir circuits in `QRC-only` has no significant influence on its predictive effectiveness in our case.

B. RQ1: Performance of `QuReBot` comparing to `Skip-only` model

We compare `QuReBot` with the `Skip-only` model to evaluate the contribution of the QRC branch in enhancing the prediction accuracy. Based on the findings in RQ0 (Sect. VI-A), we observe that the structure of the reservoir circuit has minimal impact in our scenario. Thus, we implement `QuReBot` using the widely adopted Ising Hamiltonian Reservoir [27], [28]. In addition, since we evaluate `QuReBot` in multi-step prediction tasks, we vary the prediction horizon K from 1 to 5. Moreover, we conduct feature ablation experiments by using different sets of features to analyze if `QuReBot` can preserve performance through reservoir dynamics despite a reduced number of features.

During training, all configurations of both `QuReBot` and the `Skip-only` model converged within 500 epochs. As Fig. 9 shows, the average $\mathcal{L}_{MSE}$ of `QuReBot` is consistently lower than `Skip-only`, and its boxplots are generally positioned below those of `Skip-only`. Furthermore, we conduct statistical tests on the two models for each configuration (see Table II). In most cases, the p-values are below 0.05 and $\hat{A}_{12}$ is less than 0.5, indicating that `QuReBot` significantly outperforms `Skip-only`. The only exceptions are when using the full feature set (FS_7) at $K=3$ and $K=5$, where the differences are not statistically significant.

These results suggest that, with the same input, the QRC branch enhances prediction accuracy in multi-step prediction tasks. Notably, $\hat{A}_{12}$ values for FS_7 at $K=1$, $K=2$, and $K=4$ are higher than in configurations of other feature sets



(a) FS_7, $K=1$ (b) FS_5, $K=1$ (c) FS_4, $K=1$

(d) FS_7, $K=2$ (e) FS_5, $K=2$ (f) FS_4, $K=2$

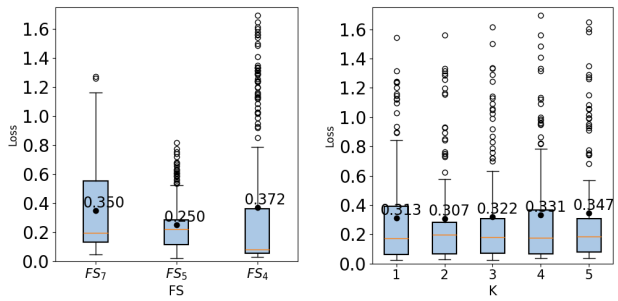
(g) FS_7, $K=3$ (h) FS_5, $K=3$ (i) FS_4, $K=3$

(j) FS_7, $K=4$ (k) FS_5, $K=4$ (l) FS_4, $K=4$

(m) FS_7, $K=5$ (n) FS_5, $K=5$ (o) FS_4, $K=5$

Fig. 9: RQ1 – $\mathcal{L}_{MSE}$ of `QuReBot` and the `Skip-only` model across different feature sets and horizons. The black dot is the average value of $\mathcal{L}_{MSE}$ in each box.

(except for FS_4 at $K=5$), indicating a larger performance gap between `QuReBot` and `Skip-only` when fewer features are available. This likely reflects the QRC's ability to compensate for reduced input dimensionality with quantum dynamics



(a) Performance across FS (b) Performance across K

Fig. 10: RQ2 – $\mathcal{L}_{MSE}$ of QuReBot with various feature sets and horizons.

to extract richer patterns, mapping features into a higher-dimensional space, and supporting robust predictions [9], [28].

Finally, the $\mathcal{L}_{MSE}$ of QuReBot (see Fig.9) shows that the Skip connection branch significantly improves prediction effectiveness of QuReBot compared to QRC-only, enabling stable convergence and achieving lower loss values. Across all configurations, QuReBot achieves, on average, a 15% improvement in prediction effectiveness over the Skip-only. It shows that quantum dynamics in QRC indeed helps in extracting patterns from features and enhancing prediction effectiveness.

Answer to RQ1: The skip connection branch enhances prediction accuracy over QRC-only. Also, QuReBot outperforms Skip-only in most configurations, highlighting the QRC branch's ability to capture meaningful patterns. These results indicate that both branches contribute to the improved performance of QuReBot.

C. RQ2 – Performance of QuReBot across different configurations

We compare QuReBot across its different configurations (feature sets FS and horizons K) to help selecting an optimal configuration given testing requirements.

First, we compare the performance of QuReBot with different feature sets FS. In Fig. 10a, each box contains QuReBot with the same feature set across varying prediction horizons. Comparing the performance of the three boxes, we observe that, without outliers, QuReBot with FS_7 displays the widest range, with the highest maximum value. FS_5 achieves the lowest average $\mathcal{L}_{MSE}$ and the smallest variation. We then conduct the Friedman test on the three feature sets. Results show that the p-value is smaller than 0.05, indicating a significant difference among their performance. Next, we perform a pairwise Wilcoxon signed rank test and $\hat{A}_{12}$ effect size; we find that both QuReBot with FS_4 and FS_5 significantly outperforms that with FS_7, while there is no significant difference between FS_4 and FS_5. These results suggest that QuReBot improves the prediction effectiveness even with fewer features. A potential reason is that the data related to the robot's velocity and orientation may introduce interference, negatively affecting the performance when predicting the next positions. Thus, when predicting next states with QuReBot for similar datasets, one can choose one feature set (either with orientation or with velocity).

TABLE III: RQ2 – Wilcoxon signed rank test among K ($>$ indicates row is significantly better than column; $<$ indicates worse; – means not significant).

	$K=1$	$K=2$	$K=3$	$K=4$	$K=5$
$K=1$	–	–	–	$>$	$>$
$K=2$	–	–	–	–	$>$
$K=3$	–	–	–	–	$>$
$K=4$	$<$	–	–	–	–
$K=5$	$<$	$<$	$<$	–	–

Next, we investigate the performance of QuReBot across varying prediction horizons K. Fig. 10b depicts the general performance of QuReBot grouped by horizon. Each box contains QuReBot with different feature sets at a fixed horizon. We also conduct statistical tests among the five horizons. The Friedman test reveals a significant difference in performance ($p < 0.05$) among them. The results of the Wilcoxon signed rank test are shown in Table III. We can observe that QuReBot with $K = 1$, $K = 2$, and FS_3 consistently outperforms that with $K = 5$, while there is no significant difference among $K = 1$, $K = 2$, and $K = 3$. Also, QuReBot with $K = 1$ is significantly better than $K = 4$, but $K = 4$ does not differ significantly from the remaining configurations, placing it in an intermediate position. These findings are supported by Fig. 10b, as the average values of $K = 1$, $K = 2$, and $K = 3$ are lower than the others, and a small increasing trend can be observed with the increase in horizon length. Those results indicate that QuReBot achieves reliable performance for short-term predictions (up to 3 steps), but prediction effectiveness may degrade for longer horizons, particularly beyond 3 steps.

Answer to RQ2: Using fewer, well-chosen features and shorter prediction horizons improves the effectiveness and reliability of QuReBot for robotic testing.

VII. Lessons Learned

A. Importance of feature selection in practice

RQ1 (Sect. VI-B) reveals that even with a reduced number of input features, quantum dynamics can help maintain high prediction effectiveness. Moreover, RQ2 (Sect. VI-C) reveals that using more features in QuReBot does not necessarily lead to better performance. These results indicate the importance of optimal feature selection for maximizing model performance. Moreover, using fewer features reduces the required number of qubits for the QRC circuit, resulting in lower-depth quantum circuits, lowering QC computational costs during training and testing.

B. Robustness of QuReBot

We performed 4-fold cross-validation by dividing the dataset into four equal phases based on the time sequence of a

complete robot operation. These segments correspond to a starting phase, two task execution phases, and a finishing phase, each exhibiting different characteristics in robot states. Despite the variation across phases, the results show that QuReBot maintains consistent performance, with MSE values remaining below 1.8 across all configurations, which is within an acceptable range. It also exhibits the robustness of QuReBot for different scenarios. More detailed analysis of QuReBot across different phases is planned as our future work, to identify which phases in our approach can lead to relatively better or worse performance.

C. Practical implication

Our results showed that QuReBot, a quantum-classical machine learning model, is promising for improving regression testing of AMRs using the strengths of both quantum and classical components. For PAL Robotics, our findings have several implications. First, it provides early but practical insights into the use of QML to support the AMR development. Second, our work serves as an initial step toward developing the necessary skills for applying QML in real-world robotic contexts. Third, it opens the door for PAL Robotics to explore a wider range of tasks that can benefit from QML in its practice.

VIII. Threats to validity

Internal Validity. QuReBot adopts a non-linear transformation as the residual connection branch. Using alternative ML models (e.g., RNN and Convolutional Neural Network) can potentially influence QuReBot's performance. However, using more advanced ML approaches for the residual connection branch might incur more resource overhead, contradicting its role as an alternative "shortcut" for QRC. Nonetheless, we plan to perform more comprehensive investigations on the influence of different ML methods.

Conclusion Validity. The inherent randomness of the neural networks and QRC can potentially influence our experimental results. To alleviate its influence and provide reliable results, we repeated each experiment 10 times and conducted statistical testing to assess the significance of differences.

External Validity. QuReBot is evaluated on one subject system—TIAGo OMNI robot. The performance of QuReBot might vary for different robots. However, QuReBot is a data-driven, hardware-agnostic approach that can be applied to most systems producing similar data structures. Moreover, we collected data from a complex and representative scenario (i.e., in a real office), where robots typically operate. We will further study the generalizability of QuReBot and investigate its effectiveness in various scenarios in the future.

IX. Related Work

Robotic systems' characteristics, such as autonomy, frequent interaction with the environment, including humans, make their testing critical [5]. Several testing techniques are adopted for robots [29], such as model-based testing [30], [31], simulation-based testing [32], [33], [34], formal verification [35], [36], and runtime monitoring [37], [38]. Regression testing ensures that updates or changes in the robotic software do not negatively affect the existing functionality of the robot, and also ensures new functionality is correct [5], [39], [40]. This paper proposes a hybrid quantum machine learning algorithm to support regression testing of an AMR from PAL Robotics. It is the first exploration of applying quantum algorithms to test an industrial robot in a real context.

Studies have used ML algorithms related to test oracles, including extracting test verdicts [41], [42], identifying metamorphic relations [43], [44], [45], and predicting expected outputs [46], [47], [41], [48], [3]. Predicting expected output is related to our work, as we generate the expected behavior of the robots as the test oracle. Related studies use a quantum extreme learning machine (QELM) to generate a test oracle for elevators [49], and a quantum neural network (QNN) for assessing the validity of test inputs of a cancer registry system [50]. Our work differs as we target robots, where we collect robot states during its operation and predict the future states. This involves processing temporal information, for which QELM is not designed, since it is memoryless.

QRC has gained interest [10], [11], [12], [13], [28] due to its inherent ability to process temporal information based on the nature of quantum dynamics. QRC has been applied to some classical tasks, including the NARMA benchmark task [51], an object classification task [52], a quantum chemistry problem [14], gene regulatory networks [53], and mobile user trajectory prediction [15]. Our work differs from these works in these ways: (1) we address next state prediction for a robot, including multi-step prediction. Also, previous studies usually focus on single-variable signals, whereas our input is multi-dimensional state information (e.g., position, velocity) as well as we predict more than one signal, which makes our task more difficult. (2) we propose a hybrid framework that combines both QRC and a "skip connection" inspired by residual connection in deep learning, significantly enhancing the QRC performance.

X. Conclusion and future work

This paper generated an ML-based test oracle for the TIAGo OMNI robot developed by *PAL Robotics*. Particularly, we propose a hybrid framework, QuReBot, that combines the QRC algorithm and a neural network. Specifically, we use this framework for the next state prediction of the robot. We evaluate QuReBot with the navigation dataset of the robot in an office map. We compare QuReBot with a QRC baseline and a neural network baseline. Results show that QuReBot outperforms both models. In addition, we compare the performance of QuReBot under various configurations (i.e., input feature sets and prediction horizons). Then, we provide practical guidance on optimal configurations with higher prediction effectiveness and discuss practical implications. In the future, we will implement QuReBot on real quantum computers to evaluate the effect of quantum noise as well as evaluate its robustness in other industrial and realistic contexts.

References

[1] "PAL Robotics," https://pal-robotics.com/, 2025, [Online; accessed 2025-07-31].

[2] Z. He, Y. Chen, E. Huang, Q. Wang, Y. Pei, and H. Yuan, "A system identification based oracle for control-CPS software fault localization," in *Proceedings of the 41st International Conference on Software Engineering*, ser. ICSE '19. IEEE Press, 2019, pp. 116–127. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00029

[3] A. Arrieta, J. Ayerdi, M. Illarramendi, A. Agirre, G. Sagardui, and M. Arratibel, "Using machine learning to build test oracles: an industrial case study on elevators dispatching algorithms," in *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. IEEE, 2021, pp. 30–39.

[4] A. Gartziandia, A. Arrieta, J. Ayerdi, M. Illarramendi, A. Agirre, G. Sagardui, and M. Arratibel, "Machine learning-based test oracles for performance testing of cyber-physical systems: An industrial case study on elevators dispatching algorithms," *Journal of Software: Evolution and Process*, vol. 34, no. 11, p. e2465, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2465

[5] A. Afzal, C. L. Goues, M. Hilton, and C. S. Timperley, "A study on challenges of testing robotic systems," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, 2020, pp. 96–107.

[6] A. Fontes and G. Gay, "Using machine learning to generate test oracles: A systematic literature review," in *Proceedings of the 1st International Workshop on Test Oracles*, 2021, pp. 1–10.

[7] J. B. Butcher, D. Verstraeten, B. Schrauwen, C. R. Day, and P. W. Haycock, "Reservoir computing and extreme learning machines for non-linear time-series data analysis," *Neural networks*, vol. 38, pp. 76–89, 2013.

[8] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, 2019.

[9] P. Mujal, R. Martínez-Peña, J. Nokkala, J. García-Beni, G. L. Giorgi, M. C. Soriano, and R. Zambrini, "Opportunities in quantum reservoir computing and extreme learning machines," *Advanced Quantum Technologies*, vol. 4, no. 8, p. 2100027, 2021.

[10] K. Fujii and K. Nakajima, "Harnessing disordered-ensemble quantum dynamics for machine learning," *Physical Review Applied*, vol. 8, no. 2, p. 024030, 2017.

[11] L. C. G. Govia, G. J. Ribeill, G. E. Rowlands, H. K. Krovi, and T. A. Ohki, "Quantum reservoir computing with a single nonlinear oscillator," *Phys. Rev. Res.*, vol. 3, p. 013077, Jan 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.3.013077

[12] R. Martínez-Peña, J. Nokkala, G. L. Giorgi, R. Zambrini, and M. C. Soriano, "Information processing capacity of spin-based quantum reservoir computing systems," *Cognitive Computation*, pp. 1–12, 2020.

[13] L. Domingo, G. Carlo, and F. Borondo, "Optimal quantum reservoir computing for the noisy intermediate-scale quantum era," *Physical Review E*, vol. 106, no. 4, p. L043301, 2022.

[14] ——, "Taking advantage of noise in quantum reservoir computing," *Scientific Reports*, vol. 13, no. 1, p. 8790, 2023.

[15] Z. Mlika, S. Cherkaoui, J. F. Laprade, and S. Corbeil-Letourneau, "User trajectory prediction in mobile wireless networks using quantum reservoir computing," *IET Quantum Communication*, vol. 4, no. 3, pp. 125–135, 2023.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern recognition*, vol. 90, pp. 119–133, 2019.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[19] "QuReBot," https://github.com/qiqihannah/qurebot/, 2025, [Online; accessed 2025-08-1].

[20] L. Grigoryeva and J.-P. Ortega, "Echo state networks are universal," *Neural Networks*, vol. 108, pp. 495–508, 2018.

[21] PAL Robotics, "TIAGo OMNI Base Robot," https://wiki.ros.org/Robots/TIAGo-OMNI-base, [Online; accessed 31-July-2025].

[22] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou, "qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor," *PRX Quantum*, vol. 2, no. 2, p. 020310, 2021.

[23] qiskit, "qiskit," https://quantum.cloud.ibm.com/docs/en/api/qiskit/qiskit.circuit.library.EfficientSU2, 2025, [Online; accessed 2025-07-31].

[24] T. Yasuda, Y. Suzuki, T. Kubota, K. Nakajima, Q. Gao, W. Zhang, S. Shimono, H. I. Nurdin, and N. Yamamoto, "Quantum reservoir computing with repeated measurements on superconducting devices," *arXiv preprint arXiv:2310.06706*, 2023.

[25] PAL Robotics, "TIAGO OMNI Base ROS 2 Simulation," https://github.com/pal-robotics/omni_base_simulation, [Online; accessed 31-July-2025].

[26] O. T. Kulseng, S. Miao, F. G. Fuchs, and A. J. Stasik, "QuantumReservoirPy: A software package for time series prediction," *J. Open Source Softw.*, vol. 10, no. 110, p. 7994, 2025. [Online]. Available: https://doi.org/10.21105/joss.07994

[27] A. Kutvonen, K. Fujii, and T. Sagawa, "Optimizing a quantum reservoir computer for time series prediction," *Scientific reports*, vol. 10, no. 1, p. 14687, 2020.

[28] N. Götting, F. Lohof, and C. Gies, "Exploring quantumness in quantum reservoir computing," *Physical Review A*, vol. 108, no. 5, p. 052427, 2023.

[29] H. Araujo, M. R. Mousavi, and M. Varshosaz, "Testing, validation, and verification of robotic and autonomous systems: a systematic review," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 2, pp. 1–61, 2023.

[30] G. Kanter and J. Vain, "Model-based testing of autonomous robots using testit," *Journal of Reliable Intelligent Environments*, vol. 6, no. 1, pp. 15–30, 2020.

[31] M. Lindvall, A. Porter, G. Magnusson, and C. Schulze, "Metamorphic model-based testing of autonomous systems," in *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*. IEEE, 2017, pp. 35–41.

[32] C. Lu, S. Ali, and T. Yue, "Epitester: Testing autonomous vehicles with epigenetic algorithm and attention mechanism," *IEEE Trans. Softw. Eng.*, vol. 50, no. 10, pp. 2614–2632, Oct. 2024. [Online]. Available: https://doi.org/10.1109/TSE.2024.3449429

[33] C. S. Timperley, A. Afzal, D. S. Katz, J. M. Hernandez, and C. Le Goues, "Crashing simulated planes is cheap: Can simulation detect robotics bugs early?" in *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2018, pp. 331–342.

[34] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand, "Can robot navigation bugs be found in simulation? an exploratory study," in *2017 IEEE International conference on software quality, reliability and security (QRS)*. IEEE, 2017, pp. 150–159.

[35] X. Zhao, M. Osborne, J. Lantair, V. Robu, D. Flynn, X. Huang, M. Fisher, F. Papacchini, and A. Ferrando, "Towards integrating formal verification of autonomous robots with battery prognostics and health management," in *International Conference on Software Engineering and Formal Methods*. Springer, 2019, pp. 105–124.

[36] M. Webster, D. Western, D. Araiza-Illan, C. Dixon, K. Eder, M. Fisher, and A. G. Pipe, "A corroborative approach to verification and validation of human–robot teams," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 73–99, 2020.

[37] J. Huang, C. Erdogan, Y. Zhang, B. Moore, Q. Luo, A. Sundaresan, and G. Rosu, "ROSRV: Runtime verification for robots," in *International Conference on Runtime Verification*. Springer, 2014, pp. 247–254.

[38] A. Desai, T. Dreossi, and S. A. Seshia, "Combining model checking and runtime verification for safe robotics," in *International Conference on Runtime Verification*. Springer, 2017, pp. 172–189.

[39] J. Wienke and S. Wrede, "Performance regression testing and run-time verification of components in robotics systems," *Advanced Robotics*, vol. 31, no. 22, pp. 1177–1192, 2017.

[40] D. Honfi, G. Molnár, Z. Micskei, and I. Majzik, "Model-based regression testing of autonomous robots," in *International SDL Forum*. Springer, 2017, pp. 119–135.

[41] R. Braga, P. S. Neto, R. Rabêlo, J. Santiago, and M. Souza, "A machine learning approach to generate test oracles," in *Proceedings of the XXXII Brazilian Symposium on Software Engineering*, ser. SBES '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 142–151. [Online]. Available: https://doi.org/10.1145/3266237.3266273

[42] F. Gholami, N. Attar, H. Haghighi, M. V. Asl, M. Valueian, and S. Mohamadyari, "A classifier-based test oracle for embedded software," in *2018 Real-Time and Embedded Systems and Technologies (RTEST)*. IEEE, 2018, pp. 104–111.

[43] B. Hardin and U. Kanewala, "Using semi-supervised learning for predicting metamorphic relations," in *Proceedings of the 3rd International Workshop on Metamorphic Testing*, 2018, pp. 14–17.
[44] P. Zhang, X. Zhou, P. Pelliccione, and H. Leung, "RBF-MLMR: A multi-label metamorphic relation prediction approach using RBF neural network," *IEEE Access*, vol. 5, pp. 21 791–21 805, 2017.
[45] D. J. Hiremath, M. Claus, W. Hasselbring, and W. Rath, "Automated identification of metamorphic test scenarios for an ocean-modeling application," in *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 2020, pp. 62–63.
[46] A. K. Monsefi, B. Zakeri, S. Samsam, and M. Khashehchi, "Performing software test oracle based on deep neural network with fuzzy inference system," in *High-Performance Computing and Big Data Analysis*, L. Grandinetti, S. L. Mirtaheri, and R. Shahbazian, Eds. Cham: Springer International Publishing, 2019, pp. 406–417.
[47] R. Zhang, Y.-w. Wang, and M.-z. Zhang, "Automatic test oracle based on probabilistic neural networks," in *Recent Developments in Intelligent Computing, Communication and Devices: Proceedings of ICCD 2017*. Springer, 2018, pp. 437–445.
[48] A. Gartziandia, A. Arrieta, J. Ayerdi, M. Illarramendi, A. Agirre, G. Sagardui, and M. Arratibel, "Machine learning-based test oracles for performance testing of cyber-physical systems: An industrial case study on elevators dispatching algorithms," *Journal of Software: Evolution and Process*, vol. 34, no. 11, p. e2465, 2022.
[49] X. Wang, S. Ali, A. Arrieta, P. Arcaini, and M. Arratibel, "Application of quantum extreme learning machines for QoS prediction of elevators' software in an industrial context," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024. New York, NY, USA: Association for Computing Machinery, 2024, pp. 399–410. [Online]. Available: https://doi.org/10.1145/3663529.3663859
[50] X. Wang, S. Ali, P. Arcaini, N. R. Veeraragavan, and J. F. Nygård, "Quantum neural network classifier for cancer registry system testing: A feasibility study," *ACM Trans. Softw. Eng. Methodol.*, Sep. 2025. [Online]. Available: https://doi.org/10.1145/3769302
[51] K. Nakajima, K. Fujii, M. Negoro, K. Mitarai, and M. Kitagawa, "Boosting computational power through spatial multiplexing in quantum reservoir computing," *Physical Review Applied*, vol. 11, no. 3, p. 034021, 2019.
[52] Y. Suzuki, Q. Gao, K. Pradel, K. Yasuoka, and N. Yamamoto, "Natural quantum reservoir computing for temporal information processing," *Scientific Reports*, vol. 12, 01 2022.
[53] W. Xia, J. Zou, X. Qiu, F. Chen, B. Zhu, C. Li, D. Deng, and X. Li, "Configured quantum reservoir computing for multi-task machine learning," *Science Bulletin*, vol. 68, no. 20, pp. 2321–2329, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095927323005807