Efficient Deep Q-Learning for Industrial Equipment Calibration in Elevator Manufacturing

Apostolos Evangelidis, Nikolaos Dimitriou, *Member, IEEE*, Paschalis Charalampous, Theofilos D. Mastos, and Dimitrios Tzovaras, *Senior Member, IEEE*

***Abstract*—Industrial equipment calibration is an essential element for the proper functioning of any production plant. Without frequent and proper calibration, the quality and efficiency of the overall production process are threatened. Despite its significance, it is often performed manually based on qualitative measures that may lead to suboptimal behavior. In this article, we propose a deep reinforcement learning (RL)-based methodology to automate the calibration process and evaluate it in an elevator control use case. Moreover, to overcome data scarcity we develop a simulation environment based on generative modeling that creates synthetic RL episodes. The proposed methodology relies on a minimal set of sensors and actuators, i.e., a webcam, an Arduino board, three stepper motors, and an edge computational unit. Nevertheless, experimental evaluations indicate that it can perform in real-time applications achieving accurate calibration results.**

***Index Terms*—Automatic calibration, deep Q-learning, parameter estimation, reinforcement learning (RL).**

I. Introduction

THE calibration of industrial machinery holds significant importance as it directly influences various parameters within the production process, ultimately determining the quality of the output. When machinery is not calibrated correctly, it can lead to deviations in measurements, inaccuracies in processing, and ultimately affect the quality of the final product. Thus, the overall waste rate is increased while the total yield is reduced. Despite the advancements in automation, many configuration tasks still rely on manual execution, which can introduce human error and inconsistencies. Consequently, this manual approach may result in suboptimal calibration outcomes, potentially compromising the efficiency and effectiveness of the production process. Therefore, proper calibration is essential for maintaining high-quality standards and maximizing productivity in industrial settings.

Deep learning (DL) has, beyond doubt, had an immense impact on a broad range of applications and has revolutionized and achieved state-of-the-art results in fields, such as computer vision and natural language processing. Besides the ability of DL to efficiently solve high-level mathematical problems, such as object detection and machine translation, it has also greatly benefited and led to substantial developments in Industry 4.0 [1]. The deployment of various DL-based defect detection or even defect prediction algorithms on the shop floor, along with the rapid improvement of edge processing capabilities, has certainly contributed toward the achievement of zero defect manufacturing [2].

More recently, another emerging subfield of machine learning (ML), namely reinforcement learning (RL), has attracted considerable attention in the research community. RL models the learning problem as a Markov decision process (MDP) and tries to find an optimal policy to maximize the cumulative sum of discounted future rewards. Even though it has been successfully applied to solve closed-environment problems with finite state spaces, such as chess [3], and go [4], there still exist several challenges when approaching more complex real-world problems that need to be addressed. The scarcity of available training data as well as the limited exploration that can be performed due to functional constraints and safety requirements of industrial applications, pose additional obstacles in successfully deploying such algorithms in manufacturing.

In this article, we aim to overcome the limitations and shortcomings of the current industrial equipment calibration practices through the development of a novel automated machine parameter calibration methodology that exploits recent advancements in deep RL (DRL). Contrary to most classical control algorithms, RL only aims to maximize cumulative future rewards, and thus, does not rely on a mathematical description of the underlying dynamics of the system. The model-free nature of RL makes it suitable for a diverse range of domains where an accurate description of the environment is not available. Hence, the adaptation of a model-free machine calibration methodology has numerous potential benefits. Moreover, a simulation environment that can be used for offline RL training is developed through the generation of synthetic RL data. Potential benefits of this approach are the enrichment of the previously gathered experience for data augmentation and the exploration of areas

Manuscript received 2 May 2023; revised 24 April 2024; accepted 11 June 2024. Date of publication 3 July 2024; date of current version 7 October 2024. This work was supported by the European Commission through Project OPTIMAI funded by the European Union H2020 programme under Grant 958264. Paper no. TII-23-1560. *(Corresponding author: Nikolaos Dimitriou.)*

Apostolos Evangelidis, Nikolaos Dimitriou, Paschalis Charalampous, and Dimitrios Tzovaras are with the Information Technologies Institute, Centre for Research and Technology Hellas (CERTH-ITI), 57001 Thessaloniki, Greece (e-mail: apostolosev@iti.gr; nikdim@iti.gr; pcharalampous@iti.gr; dimitrios.tzovaras@iti.gr).

Theofilos D. Mastos is with KLEEMANN HELLAS SA, 61100 Kilkis, Greece (e-mail: t.mastos@kleemannlifts.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2024.3417254.

Digital Object Identifier 10.1109/TII.2024.3417254

of the state space that cannot be accessed due to functional constraints.

The effectiveness of the employed methodology is demonstrated in an elevator control use case, where the objective is to fine-tune the parameters of a control valve-block to attain the desired movement of a hydraulic elevator. With the aid of a single webcam, along with a robust soft sensor (SS) model the velocity and displacement of the elevator are accurately estimated in real time. In order to fit the estimated displacement curve to a reference one, the RL agent acts by simultaneously rotating all valve-block adjustments, aiming to maximize the negative squared distance between these two curves. Moreover, to evaluate the benefits of the developed simulation environment, we investigate the effect of incorporating synthetic data into the RL calibration framework. Experimental results indicate that the proposed system can successfully configure the parameters of the valve-block, and thus, obtain the desired response. The main contributions of this article are summarized as follows.

1) An automated machine parameter calibration methodology based on RL is proposed.
2) A robust SS model to estimate displacement and velocity from low-resolution frames is introduced.
3) A simulation environment for synthetic RL data generation is developed.
4) The methodology is evaluated on an elevator control use case.

The rest of this article is organized as follows. In Section II, we provide an overview of existing control-based approaches for parameter calibration. In addition, we discuss industrial applications utilizing both DRL and generative methodologies. In Section III, we present the considered use case in detail along with the low cost instrumentation setup, while Section IV develops the mathematical framework of the proposed machine calibration methodology. In Section V, the experimental results validating the applied methodology are exhibited. Finally, Section VI concludes this article.

II. Related Work

In this section, we present some relevant industrial applications that are closely related to the results of this article. Initially, we examine conventional control-oriented calibration methodologies. Subsequently, we discuss both industrial applications employing DRL and generative approaches.

A. Existing Calibration Approaches and Limitations

Classical control algorithms play a pivotal role in the calibration of industrial equipment. With their robustness and reliability, they continue to be widely adopted across diverse industrial sectors, contributing to enhanced efficiency, quality, and safety in manufacturing environments. In [6], the authors develop a dynamic intelligent control system to optimize the rate of penetration (RoP) of a drilling process. They split the process in two stages, namely, ROP modeling and ROP optimization, and validate their methodology in a real-world drilling process. In another relevant application [7], an event-triggered control mechanism is developed along with theoretical guarantees regarding its performance and is evaluated in a high-speed linear motor control case study. Moreover, the authors in [8], presented a practical antiwindup control methodology for open-loop systems and utilized it to optimize the performance of a twin-rotor turbofan engine. While the effectiveness of control algorithms has been established through their broad applicability in various industrial applications [9], [10], they still suffer from shortcomings that mainly stem from their dependence on a mathematical model of the environment in which they operate. The inability of a mathematical model to capture and describe several phenomena, such as the complexities and dynamics of the real-world process, sudden changes in operating conditions or abrupt disturbances, coupled with the tuning of the appropriate control parameters; compose restricting factors that may lead to suboptimal performance.

B. Reinforcement Learning in Industry 4.0

Unlike classical control, RL focuses on achieving high-level objectives through the optimization of a reward function and does not depend on a mathematical description of the environment, thus offering the potential to surpass current limitations. In [11], a Q-learning agent to adjust the weights of a state-of-the-art wireless network graph routing algorithm is introduced. The method is validated through simulations that show a reduction in the average network latency while the expected network lifetime remains approximately the same. In another application [12], a knowledge-assisted deep deterministic policy gradient algorithm is proposed to address the damage that may be caused to machines by the actions taken by a RL agent during random exploration. The suggested methodology is evaluated in a cooperative wind farm control use case. In the same domain, the authors in [13] formulate the wind farm power tracking problem as a zero-sum game, and use a preview-based deep RL method to handle the uncertainties inherently existent in the considered environment. In another line of applications, the authors in [14] introduced a two-stage RL-based strategy relying on deterministic policy gradients to mitigate the voltage violations caused by the uncertainty of electric vehicle charging stations, whereas in [15] a Q-learning scheme to predict plug-in hybrid electric vehicle charging station loads was developed. In addition, to mitigate the effect of imbalanced faulty and normal samples in a fault diagnosis use case, an RL-based sample selection strategy (DiagSelect) was suggested in [16] and is evaluated on both synthetic and real industrial data. A plant-wide performance optimization technique for large-scale industrial processes through the decomposition of the main optimization problem into multiple local subproblems solved by multiple distributed RL agents was introduced in [17]. Another multiagent approach was developed in [18] toward the achievement of a Self-X cognitive manufacturing network with a higher level of automation.

C. Overcoming Data Scarcity With Synthetic Generation

Despite the increasing adaptation of RL, there still exist some challenges that need to be addressed that are mainly related to the inherent scarcity of available data in industrial settings.

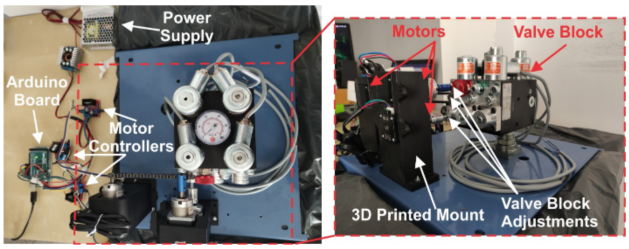


Fig. 1. Hardware setup for the actuation of the motors and the adjustment of the EVs.

Generative modeling approaches have shown remarkable success and have been increasingly applied in a broad range of different domains after being popularized through the introduction of generative adversarial networks (GANs) [20], and variational autoencoders (VAEs) [19]. In the industrial setting, generative models have been mainly used for data augmentation so as to overcome the commonly encountered data scarcity problem. The authors in [21] developed a defect exaggeration approach using GANs to facilitate the identification of tiny surface defects and demonstrate that the proposed method can improve the accuracy of the state-of-the-art approaches. Moreover, in [22] finite element method (FEM) simulations and GANs are combined to obtain missing faulty samples and enlarge the existing bearing fault dataset, thus leading to improved defect detection accuracy. Finally, in [23] a deep generative model that relies on long short term memory (LSTM) networks and an attention mechanism, named AMBi-GAN, are introduced for multivariate Industrial Internet of Things (IIoT) times series anomaly detection.

III. Use Case Description

In this section, we describe the investigated use case concerning the valve block calibration process, highlighting the multiple benefits that an automated calibration methodology can provide. Moreover, a detailed description of the low-cost hardware setup is also given.

A. Case Study

The process of calibrating the parameters of an industrial machine is crucial as it directly affects the quality of the overall production process. A critical stage in the elevator manufacturing industry is the calibration of the hydraulic elevator control valves (EVs), controlling the upward and downward movement of the elevator. The calibration is considered successful if the response matches a given reference curve that varies according to the process parameters (elevator type, maximum load) and the specific needs of each installation.

Misadjustments of the EVs may lead to unstable operation, undesirable acceleration, sudden stops before the destination is reached, and may even cause damage to the whole arrangement. Due to the complex nature of the process, it is usually performed by an expert who has been thoroughly trained and has gained the necessary experience and intuition to operate the control valves. The valve block is connected to some weights simulating the movement of the elevator cabin within a controlled testing space, and after repetitive trials, the operator adjusts the block's parameters in order to obtain the desired response. It must be noted that the manual calibration is time-consuming and may lead to inaccurate results due to the introduction of the human factor in the process.

All the previously mentioned factors rationalize the development of an automated configuration process that would substantially improve the calibration process in terms of speed and accuracy.

B. Instrumentation

For the considered use case, we utilize the 3/4" EV100 block from Blain Hydraulics GmbH. Three Nema 17 stepper motors are placed on a 3-D-printed supporting arrangement, which was constructed for the needs of the present study, in order to be mounted at specific positions of the valve block controlling that way the acceleration time to the downward speed, the deceleration time from the downward speed to the leveling speed, and the downward leveling speed, respectively. In addition, an Arduino Mega 2560 Rev3 is used to perform the actuation of the employed motors, whereas both the training and inference of the RL agent are performed on an Nvidia GeForce RTX 3060 GPU. Finally, a low-cost Creative Live! Cam Sync 1080p V2 webcam is used to capture frames during the movement of the weights. The employed hardware setup is shown in Fig. 1.

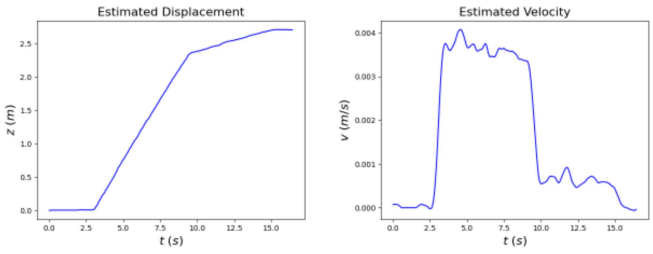
IV. Proposed Methodology

In this section, we describe the developed methodology in detail. First, the proposed SS module for velocity and displacement estimation is presented. Subsequently, we provide the mathematical formulation of the automated parameter calibration methodology and the generative framework to create synthetic data along with the offline RL training algorithm.

A. Soft Sensing Module

The first step in automating the calibration process is to obtain an accurate estimation of the actual displacement and velocity of the elevator-weights in real time. To this end, and to avoid extra instrumentation costs incurred by a motion encoder, we develop a robust soft sensing module that, given consecutive frames captured by a low-cost webcam at its input, can estimate both the displacement and velocity of the weights. A lightweight network is used that exploits depthwise separable convolutions introduced in [24] for faster inference and residual connections [25]. The proposed system is simple, easy to use, does not require any specialized knowledge to be set up, and can thus be easily deployed on the shop floor.

1) Dataset Construction: Since successful ML models need thousands of annotated data samples, accurate annotation by an expert is a time-consuming process that may incur bottlenecks and slowdowns. To obtain ground truth measurements, we adopt the following automated labeling strategy: Let $\mathbf{F}_t \in \mathbb{R}^{h \times w \times 3}$ be the captured frame at time t, where h and w are the frame's height and width, respectively. For weight tracking, we make use of the reliable CSRT tracker [5], which is initialized by



(a) (b)

Fig. 2. (a) Estimated displacement and (b) estimated velocity of the weights.

manually specifying the bounding box $\mathbf{b}_0 = (\mathbf{p}_0^1, \mathbf{p}_0^2, \mathbf{p}_0^3, \mathbf{p}_0^4)$ of the elevator-weights within the initial frame $\mathbf{F}_0$, where $\mathbf{p}_0^i \in \mathbb{R}^2$ for $i = 1, 2, 3, 4$, are the pixel coordinates of the box's four corners. After the initialization, the tracker outputs the bounding box $\mathbf{b}_t$ containing the location of the object of interest at time step t. The magnitude of the weight's displacement $\hat{z}_t \in \mathbb{R}$ is calculated as the magnitude of the average displacement of the bounding box's four corners.

Let $\{\hat{z}_t\}_{t=1}^T$ be the set of observed displacement values for a single trial, where T is the number of time steps needed for the weights to reach their final position. To obtain a robust estimate of the weight's velocity, i.e., the displacement's derivative, we fit a local linear model $h_t(x) = \lambda_t x + \beta_t$ for each interval $[t-k, t+k]$ for some $k \ll T$ using least squares regression. Subsequently, the velocity can be analytically estimated as

$$\hat{v}_t = \frac{1}{\Delta t}\frac{d\,h_t}{d\,x}(x = t) = \frac{\lambda_t}{\Delta t} \tag{1}$$

where Δt is the inverse of the webcam's frame rate. The estimated displacement and velocity curves for a downward movement trial are displayed in Fig. 2. Notice how the four different stages of movement, namely, the acceleration stage, the constant downward speed stage, the deceleration stage, and the leveling speed stage can be clearly identified. The estimation process is schematically summarized in Fig. 3.

Finally, in order to construct a dataset, the captured frames are transformed into $\hat{\mathbf{F}}_t = \mathbf{H}\mathbf{F}_t$ using the perspective correction transform $\mathbf{H}$, such that only the moving parts within the image are retained and the irrelevant information is discarded. The effect of such a perspective transform is shown in Fig. 4. The original frame $\mathbf{F}_t$ captured by the webcam can be seen on the left side of Fig. 4, whereas the transformed frame $\hat{\mathbf{F}}_t$ can be seen on the right. Once all the recorded frames have been transformed and the velocity and displacement curves for all trials have been estimated, we construct the soft sensing dataset $\mathcal{D}_{ss} = \{(\hat{\mathbf{F}}_i, \mathbf{y}_i)\}_{i=1}^{N_{ss}}$, where N_{ss} is the total number of available samples and $\mathbf{y}_i = [\hat{z}_i\ \hat{v}_i]^T$.

2) Proposed Architecture: Since the overall calibration process will operate in real time, it is crucial to develop compact lightweight models that are suitable for edge deployment without sacrificing performance. To this end, inspired by MobileNet, we develop a network that consists of multiple stacked downsampling blocks that utilize depthwise separable convolutions. That way, inference is significantly sped up as the total number of trainable parameters is substantially reduced.

The proposed residual downsampling blocks consist of two convolutional layers followed by batch normalization (BN) [26] and a rectified linear unit (ReLU) activation, as shown in Fig. 5. Following common practice, after each block the spatial dimensions of input $\mathbf{x}$ are reduced in half whereas the number of channels is doubled. A feature extractor $\mathbf{G}(\mathbf{x}) \in \mathbb{R}^d$ is defined by stacking multiple such blocks along with a final adaptive average pooling layer, and the network's output is computed through a linear transform, as in (2), where $\mathbf{A}_{ss} \in \mathbb{R}^{d\times 2}$ and $\mathbf{b}_{ss} \in \mathbb{R}^2$ are trainable parameters

$$\mathbf{y} = \mathbf{A}_{ss}^T\mathbf{G}(\mathbf{x}) + \mathbf{b}_{ss}. \tag{2}$$

To capture the dynamic behavior of the weights necessary for the development of a robust velocity estimator, the input is constructed by stacking multiple consecutive frames from the same trial such that $\mathbf{x}_i = [\hat{\mathbf{F}}_i\ \hat{\mathbf{F}}_{i-1}\ \ldots\ \hat{\mathbf{F}}_{i-l+1}]$. Subsequently, the parameters of both the feature extractor and the final linear layer are optimized using Adam to minimize the empirical risk

$$\mathcal{J}_{ss} = \frac{1}{N_{ss}}\sum_{i=1}^{N_{ss}} \|\mathbf{A}_{ss}^T\mathbf{G}(\mathbf{x}_i) + \mathbf{b}_{ss} - \mathbf{y}_i\|_2^2. \tag{3}$$

B. RL Formulation

We formulate the machine parameter calibration problem as a finite horizon MDP that can be described by a tuple $(\mathcal{X}, \mathcal{A}, P, R)$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ the action space, $P : \mathcal{X}\times\mathcal{A} \to \mathcal{X}$ is the transition probability distribution characterizing the underlying dynamics of the system, and $R : \mathcal{X} \to \mathbb{R}$ is the reward function. Given that we aim to regulate the motion of the weights, it is necessary to define a sufficiently informative state that adequately captures the dynamics of the system. Similar to the soft sensing case, we construct the state vector $\mathbf{x}_t$ by stacking l consecutive prior frames. As a result, the state space is defined as

$$\mathcal{X} = \left\{\mathbf{x}_t \in \mathbb{R}^{h\times w\times 3\times l} \,|\, \mathbf{x}_t = [\hat{\mathbf{F}}_t, \hat{\mathbf{F}}_{t-1}, \ldots, \hat{\mathbf{F}}_{t-l+1}]\right\}. \tag{4}$$

Interaction with the environment is achieved by rotating the three valve adjustments, and thus, the action space can be defined as

$$\mathcal{A} = \{\mathbf{a} = (a_1, a_2, a_3) \mid a_i \in \{-\epsilon, 0, \epsilon\},\ i = 1, 2, 3\}. \tag{5}$$

Concretely, at each time step, each valve adjustment can either be rotated clockwise or anticlockwise by a predefined amount ϵ or is left unchanged. By construction, the cardinality of the action set, i.e., the number of different actions that can be taken at each time step, is $m = 27$. Let v_t, z_t be the reference velocity and displacement curves, respectively, that we wish to achieve. The reward function is defined as the negative squared distance between the estimated velocity and displacement using the soft sensing model and the corresponding reference values as

$$r_t = R(\mathbf{x}_{t+1}) = -\left\|\mathbf{A}_{ss}^T\mathbf{G}(\mathbf{x}_{t+1}) + \mathbf{b}_{ss} - \begin{bmatrix} z_{t+1} \\ v_{t+1}\end{bmatrix}\right\|_2^2. \tag{6}$$

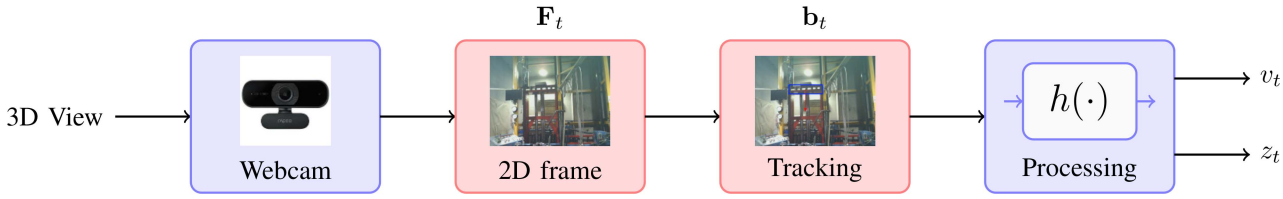


Fig. 3. Schematic overview of the complete velocity and displacement estimation process. The webcam first captures a frame $\mathbf{F}_t$, the elevator-weight bounding box $\mathbf{b}_t$ is localized though tracking, and finally after processing the coordinates of the bounding box the displacement $\hat{z}_t$ and velocity $\hat{v}_t$ are estimated.

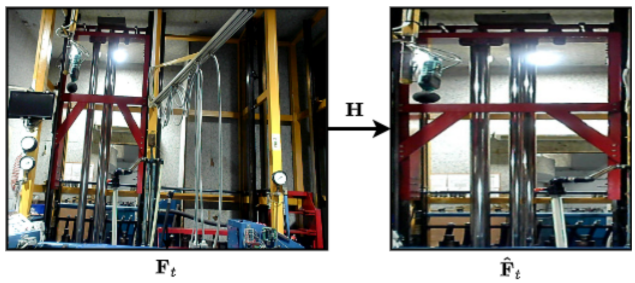


Fig. 4. Illustration of the perspective correction transform. Using the matrix $\mathbf{H}$, the initialy captured frame $\mathbf{F}_t$ is transformed into $\hat{\mathbf{F}}_t$.

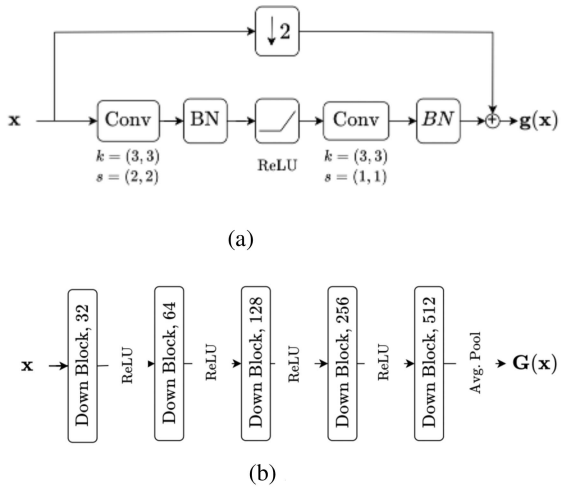


Fig. 5. Schematic overview of the defined architecture. (a) Residual downsampling module. (b) Architectural diagram of the feature extractor $\mathbf{G}(\mathbf{x})$. (a) Downsampling block. (b) Feature extractor.

The choice of this reward function was made such that the maximization of the cumulative sum of future rewards is equivalent to the minimization of the mean-squared error between the estimated and reference curves.

The objective is to find a policy $\pi : \mathcal{X} \to \mathcal{A}$ that maximizes the expected sum of discounted future rewards $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$, where $\mathbf{a}_t \sim \pi(\cdot \,|\, \mathbf{x}_t)$ for each $t = 1, \ldots, T$ and $\mathbf{x}_{t+1} \sim P(\cdot \,|\, \mathbf{x}_t, \mathbf{a}_t)$. Due to the episodic nature of the considered use case we set $\gamma = 1$.

1) Deep Q-Learning: The sequential decision-making process can be solved by learning estimates of the state-action value function $Q_\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, defined as the expected sum of future rewards when taking action $\mathbf{a}_t$ from state $\mathbf{x}_t$ and following policy π thereafter:

$$Q_\pi(\mathbf{x}, \mathbf{a}) = \mathbb{E}_\pi \left[\sum_{k=1}^{T-t} r_{t+k} \,|\, \mathbf{x}_t = \mathbf{x}, \mathbf{a}_t = \mathbf{a} \right]. \tag{7}$$

Once an accurate estimate has been obtained, the optimal policy can easily be derived by selecting the highest valued action for each state.

To construct a sufficiently representative function approximator we use the same deep architecture as in the soft sensing case and exploit the knowledge that has been gained by only changing the final linear layer and keeping the feature extractor intact, $Q_{\mathbf{w}}(\mathbf{x}, \mathbf{a}) = \mathbf{A}_{rl}^T \mathbf{G}(\mathbf{x}) + \mathbf{b}_{rl}$. By reusing already trained weights, we drastically reduce the number of trainable parameters, and thus, facilitate the training process. The parameter vector $\mathbf{w} = [\mathbf{A}_{rl}^T \; \mathbf{b}_{rl}] \in \mathbb{R}^{(d+1)\times m}$ is optimized by minimizing the Bellman equation residual shown as

$$\mathcal{J}_{\text{rl}} = \mathbb{E}_{\mathcal{D}_{rl}} \left[\left(Q_{\mathbf{w}}(\mathbf{x}_t, \mathbf{a}_t) - r_t - \max_{\mathbf{a} \in \mathcal{A}} Q_{\bar{\mathbf{w}}}(\mathbf{x}_{t+1}, \mathbf{a}) \right)^2 \right] \tag{8}$$

where $\mathcal{D}_{\text{rl}}$ is the replay experience buffer consisting of all previously encountered transitions of the form $\tau = (\mathbf{x}_t, \mathbf{a}_t, r_t, \mathbf{x}_{t+1})$ and $\bar{\mathbf{w}}$ are the parameters of the target network.

2) Exploration Strategy: Exploration is a fundamental part of any RL algorithm, as agents highly depend on it to obtain informative data about the environment in which they operate. The more exploration is performed, the more state-action pairs are encountered, and better estimates of the state-action value function can be obtained. Hence, it is crucial to develop efficient exploration strategies as they directly affect the agent's performance.

Due to hardware constraints regarding the actuation time of the stepper motors and the requirement for real-time performance, the rotation angle ϵ is kept sufficiently small. This choice comes with both benefits and impediments. On the one hand, smaller revolution steps may achieve more accurate calibration results, whereas, on the other, they may delay the overall process as more steps are needed to rotate the adjustments by the desired amount. In addition, small random rotations do not significantly alter the state, and are therefore not suited for exploration.

In order to overcome these shortcomings, we propose a variant of the frequently used ε-greedy exploration strategy. Instead of selecting a random action from $\mathcal{A}$ with probability ε at each time step and taking it once, we repeat the same action multiple consecutive times to achieve larger rotations and hence adequate variation in the state vector $\mathbf{x}_t$. Finally, we vary the exploration

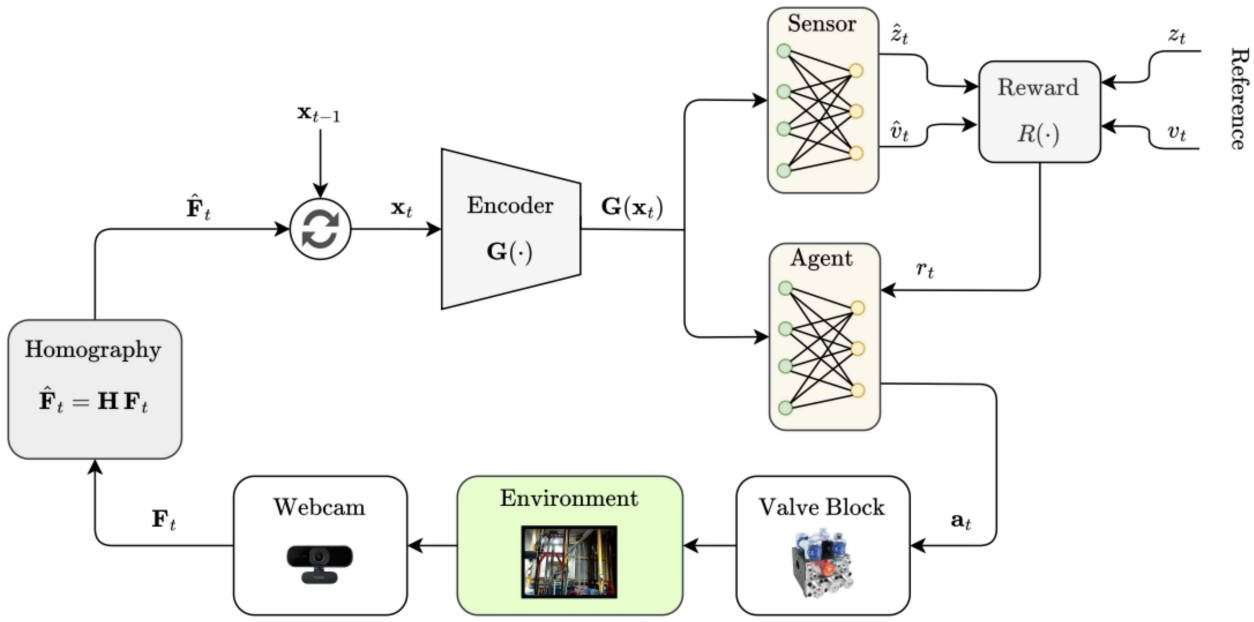


Fig. 6. Operational flow of Algorithm 1 starts with the transformation of the camera sensor's captured frame based on the estimated perspective transformation. Subsequently, the state buffer $\mathbf{x}_t$ is updated and fed into the feature encoder $\mathbf{G}$. Utilizing the SS model, the algorithm computes the reward r_t, while the RL agent determines the optimal action $\mathbf{a}_t$. Upon selecting the action, adjustments are made to the parameters of the valve block, thereby influencing the movement of the elevator weights.

ratio by linearly decaying it over the number of episodes as

$$\varepsilon_i = \varepsilon_0 + \max\left\{1,\ \frac{i}{M}\right\}(\varepsilon_M - \varepsilon_0) \tag{9}$$

where ε_0 is the initial exploration ratio and M the number of training episodes. The pseudo-code for the proposed parameter calibration methodology is described in Algorithm 1, whereas the whole process is schematically represented in Fig. 6.

Algorithm 1: Deep Q-Learning Control.

Input: Number of episodes N_e, state vector length l
Learning rate α, batch size K

Initialize the buffer $\mathcal{D}_{\text{rl}} = \{\varnothing\}$
for $i = 1 : N_e$ **do**
  for $t = 0 : T_i - 1$ **do**
    if $t < l$ **then**
      $\hat{\mathbf{F}}_{t+1} = \mathbf{H}\,\mathbf{F}_{t+1}$
    else
      $\mathbf{x}_t = [\hat{\mathbf{F}}_{\mathbf{t}} \ldots, \hat{\mathbf{F}}_{\mathbf{t-l+1}}]$
      $\mathbf{a}_t = \text{SelectAction}(\mathbf{x}_t,\ \varepsilon_i)$
      $\hat{\mathbf{F}}_{t+1} = \mathbf{H}\mathbf{F}_{t+1}$
      $\mathbf{x}_{t+1} = [\hat{\mathbf{F}}_{\mathbf{t+1}} \ldots, \hat{\mathbf{F}}_{\mathbf{t-l+2}}]$
      $r_t = R(\mathbf{x}_{t+1})$
      $\mathcal{D}_{\text{rl}} \leftarrow \mathcal{D}_{\text{rl}} \cup \{(\mathbf{x}_t,\ \mathbf{a}_t,\ \mathbf{x}_{t+1},\ r_t)\}$
      $\mathcal{B} = \text{Sample}(\mathcal{D}_{\text{rl}})$
      $y_k = r_k + \max_{\mathbf{a}} Q_{\bar{\mathbf{w}}}(\mathbf{x}_{k+1},\ \mathbf{a}),\quad \forall k \in \mathcal{B}$
      $\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\alpha}{K}\nabla_{\mathbf{w}_t} Q_{\mathbf{w}_t}(\mathbf{x}_t,\ \mathbf{a}_t)\sum_{k=1}^{K}(y_k - Q_{\mathbf{w}_t}(\mathbf{x}_t,\ \mathbf{a}_t))$
    end if
  end for
end for

C. Generative Modeling

An omnipresent challenge in successfully deploying DL models for industrial applications is the scarcity of available training data. Collecting large amounts of data necessary for training efficient DL models is prohibitive as it would significantly interfere with the production process, thus slowing it down. Consequently, the development of data-efficient learning algorithms is of utmost importance. To this end, using the data collected from only a few calibration trials, we develop a simulation environment through which we are able to generate novel data.

We split the generation task in two steps. First, given the velocity and displacement curves we wish to simulate, we generate the corresponding frame sequence. In the second step, we use the generated frames to construct the current and future states at each time step and predict the action that led to this transition. This framework allows the generation of novel RL transitions that are used to facilitate the training process.

1) Frame Generation: At the first step of the generation process, we aim to transform a 1-D displacement curve to the corresponding sequence of 2-D frames. The constrained nature of the examined use case and the high similarity between image samples during the motion of the weights allows us to approach this problem in a supervised learning fashion. Specifically, we intend to invert the soft sensing process by predicting frames $\hat{\mathbf{F}}_i$ considering the value of the displacement $\hat{z}_i$.

Taking into account the disproportional dimensionality between the frames and the scalar displacement values, we conclude that a decoder is the optimal architectural choice for the frame generation network. We convert the proposed

downsampling blocks of Fig. 5 to upsampling blocks by replacing both convolutional layers with transpose convolutions and instead of downsampling, bicubic interpolation is used to increase the spatial dimension of the input tensor. A linear layer is initially applied to map the scalar value to a higher dimensional vector, and the multiple upsampling blocks are then cascaded until the desired output resolution is reached. Finally, depthwise separable convolutions are replaced with regular convolutions since time efficiency is not critical.

In particular, let $\mathbf{D}(z)$ be the decoder model for the reconstruction of the frames conditioned on the value of the displacement. Exploiting the existing soft sensing dataset, we optimize the parameters of $\mathbf{D}(z)$ by minimizing the reconstruction loss

$$\mathcal{J}_{\text{FR}} = \frac{1}{N_{ss}} \sum_{i=1}^{N_{ss}} \left\| \hat{\mathbf{F}}_i - \mathbf{D}(z_i) \right\|_1 \tag{10}$$

where the ℓ_1 norm instead of the ℓ_2 norm is used to measure the discrepancy between the actual and the predicted frames as it is known to promote sharper image reconstruction results.

2) Action Prediction: The next step of the generation process is to employ the generated frames to estimate the actions that led to the movement we wish to simulate. Our reasoning behind this is the following: If the underlying dynamics of the system are deterministic, i.e., if taking action $\mathbf{a}_t$ from state $\mathbf{x}_t$ deterministically leads to a specific next state $\mathbf{x}_{t+1}$ without ambiguity, then it is possible to invert this relation and predict the action given the current and future states.

Based on this assumption, we construct a network $\mathbf{E}(\mathbf{x}_t, \mathbf{x}_{t+1})$ that takes both the current and future states at its input and predicts the action that was taken. We apply the same deep architecture that we used for Q-function approximation. We replace all depthwise convolutions with regular convolutions and add a softmax activation after the last layer since we only aim to perdict actions and not state-action values. The network's parameters are optimized by minimizing the loss

$$\mathcal{J}_{\text{AP}} = \frac{1}{N_{rl}} \sum_{\tau \in \mathcal{D}_{\text{rl}}} \text{CE}\left(\mathbf{a}_t, \mathbf{E}(\mathbf{x}_t, \mathbf{x}_{t+1})\right) \tag{11}$$

where $\text{CE}(\cdot, \cdot)$ is the categorical cross entropy between the one-hot encoded action vector and the corresponding predicted value and N_{rl} the number of RL transitions used for training. The process of generating novel RL data is summarized in Algorithm 2, whereas the decoder–encoder structure for action prediction is schematically represented in Fig. 8.

3) Movement Randomization: The movement of the weights can be described by four stages: 1) the acceleration to the constant downward speed stage, 2) the downward speed stage, 3) the deceleration to the leveling speed stage, and 4) the leveling speed stage, during all of which either the velocity or acceleration of the movement remain constant. To create random but realistic velocity sequences for RL data generation, we add uniform random noise to the aforementioned acceleration and velocity values of the reference curve. Moreover, we also add low-pass filtered white Gaussian noise to introduce small, smooth variations. Then, applying numeric integration, the calculation of the corresponding displacement sequence is achieved. Five examples of randomized velocity curves along with three generated frames at different stages can be seen in Fig. 7.

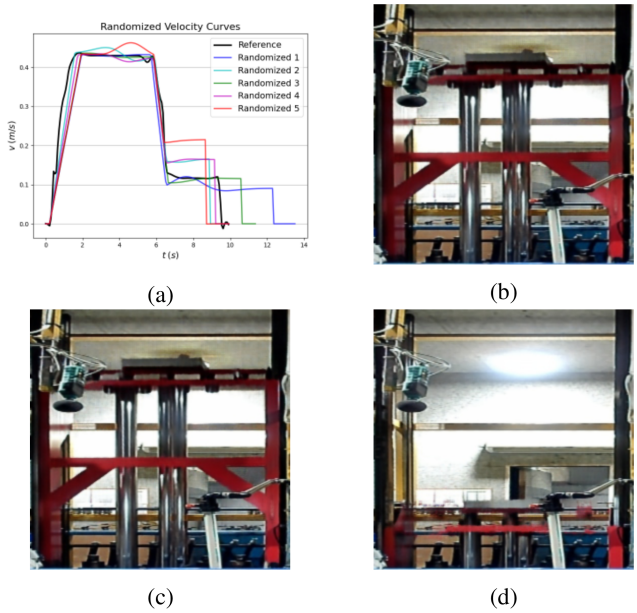


Fig. 7. (a) Randomized velocity curves used for synthetic RL data generation, (b)–(d) three exemplary generated frames using the developed simulation environment at different stages of the movement.

Algorithm 2: Synthetic RL Data Generation.

Input: Number of episodes to generate N_e
Length of the state vector l
Output: Generated transition buffer $\mathcal{D}_{\text{gen}}$

Initialize the buffer $\mathcal{D}_{\text{gen}} = \{\varnothing\}$
for $i = 1 : N_e$ **do**
  Randomize curves $\{\hat{z}_t\}_{t=1}^{T_i}$, $\{\hat{v}_t\}_{t=1}^{T_i}$
  for $t = 0 : T_i - 1$ **do**
    $\hat{\mathbf{F}}_{t+1} = \mathbf{D}(z_{t+1})$
    if $t \geq l$ **then**
      $\mathbf{x}_t = [\hat{\mathbf{F}}_{\mathbf{t}} \ldots, \hat{\mathbf{F}}_{\mathbf{t-l+1}}]$
      $\mathbf{x}_{t+1} = [\hat{\mathbf{F}}_{\mathbf{t+1}} \ldots, \hat{\mathbf{F}}_{\mathbf{t-l+2}}]$
      $\mathbf{a}_t = \mathbf{E}(\mathbf{x}_t\ \mathbf{x}_{t+1})$
      $r_t = -\|[\hat{z}_{t+1}\ \hat{v}_{t+1}]^T - [z_{t+1}\ v_{t+1}]^T\|_2^2$
      $\mathcal{D}_{\text{gen}} \leftarrow \mathcal{D}_{\text{gen}} \cup \{(\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1}, r_t)\}$
    end if
  end for
end for

V. Experimental Results

In this section, the developed parameter calibration methodology, as well as the effect of using synthetic RL training data, generated through the proposed simulation environment, are evaluated on an actual elevator control valve block calibration use case. The developed methodology is deployed and tested on-site in a real industrial setting, both in terms of performance and reliability.

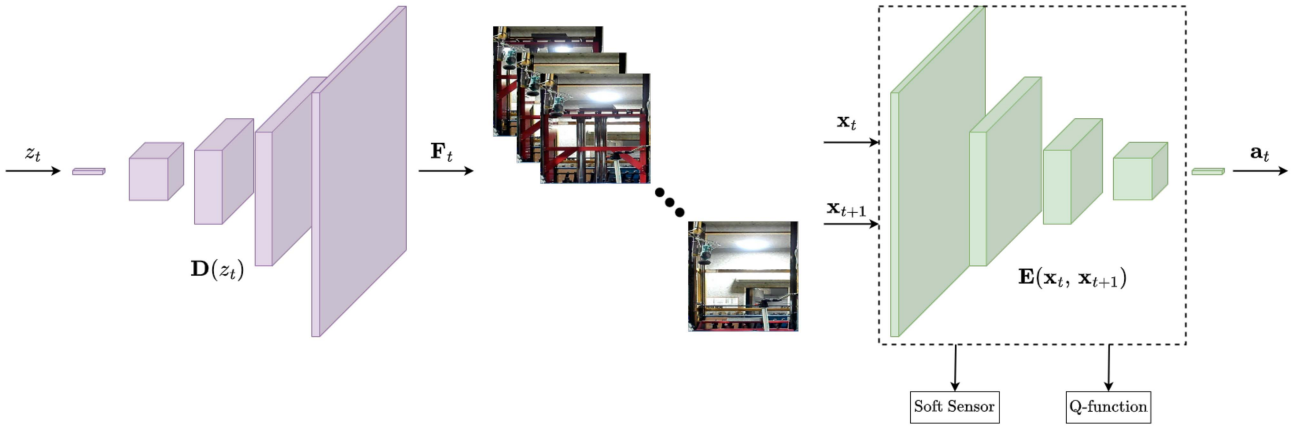


Fig. 8. Proposed generative modeling architecture. The decoder $\mathbf{D}(z_t)$ transforms a sequence of displacement values into the corresponding frames and subsequently the encoder $\mathbf{E}(\mathbf{x}_t, \mathbf{x}_{t+1})$ estimate the actions that led to this movement. The architecture of the encoder is the same as in the soft sensing and Q-function approximation cases with only the last layer changed.

TABLE I
MISADJUSTMENTS OF THE 3 INITIAL TRAINING STATES

	State 1	State 2	State 3
Valve 1	-180^o	$+180^o$	-180^o
Valve 2	-180^o	$+180^o$	-270^o
Valve 3	-90^o	$+90^o$	$+45^o$

A. Training

We use ten recorded episodes to train the proposed soft sensing module for online velocity and displacement estimation and to develop the simulation environment. Let $\mathcal{D}_{\text{rl}}$ be the set of all such RL transitions. Using the velocity randomization scheme described in IV-C3, we additionally generate ten synthetic episodes and denote the set of all generated RL transitions as $\mathcal{D}_{\text{gen}}$. To evaluate the effect of using synthetic RL data, we compare the performance of three agents: *Agent-1*, who is randomly initialized; *Agent-2*, who is initialized using offline Q-learning on real recorded transitions sampled from $\mathcal{D}_{\text{rl}}$; and *Agent-3*, who is initialized using offline Q-learning on both real and generated transitions sampled from $\mathcal{D}_{\text{rl}} \cup \mathcal{D}_{\text{gen}}$. To obtain a reference point, the valve block is manually calibrated by a trained operator. This is done so as to control the misadjustments that will be imposed during both the training and evaluation stages such that the three agents can be justly compared. Using that reference, we define three starting states by rotating the valve adjustments by certain amounts exhibited in Table I. For each agent and for each state, two consecutive trials are performed, resulting in a total of six episodes used for training, during which the agent's parameters are fine-tuned using Algorithm 1. The initial exploration ratio is set to 0.7 and is linearly decayed according to (9) until it reaches 0.2. For both soft sensing and Q-learning, the model used is constructed by cascading five residual downsampling blocks. It is noted that deeper architectures cannot be considered due to real-time performance requirements. The number of frames utilized to construct the state vector is set to $l = 5$, which was experimentally determined to create a balance between the information content and the computational efficiency of the training algorithm. For all episodes, we employ a learning rate equal to 10^{-4} and batch size 1.

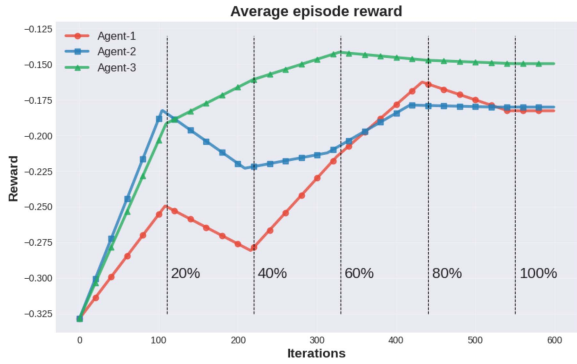


Fig. 9. Average reward achieved over the course of 5 consecutive episodes (≈ 600 iterations).

B. Performance Evaluation

Once the training process has finished and in order to quantitatively evaluate the performance of the developed system and compare the three agents, we rotate the valve adjustments 1, 2, and 3 by $+180^o$, -180^o, and $+45^o$, respectively, from the reference position. This configuration results to a decreased leveling speed which is a commonly encountered deficiency of the EVs. From this starting state, five consecutive trials are performed for each agent and their performance is assessed and compared based on the achieved accuracy (in terms of mse) and the number of trials needed to correct the imposed misadjustments. The exploration ratio is set to 0.02 and is kept constant for all testing trials. The average episode reward achieved by the three agents is shown in Fig. 9.

We notice that all three agents are able to calibrate the parameters of the valve block as the reward function exhibits a clear increasing trend over the course of the episodes used for testing. Agents 1 and 2 behave similarly and converge to approximately the same value that does not significantly change during the

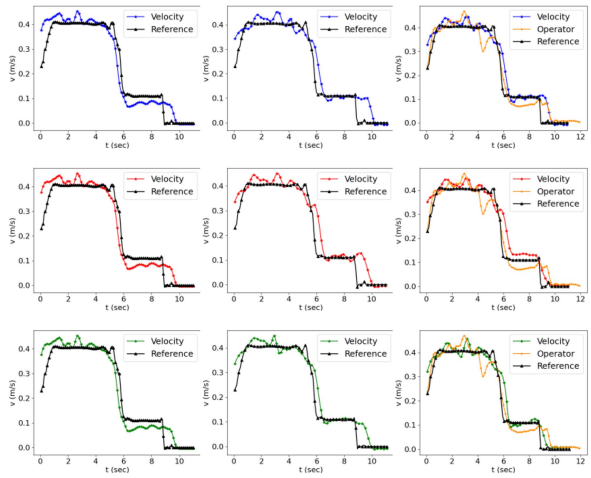


Fig. 10. Actual velocity response compared to the reference. The first, second, and third rows correspond to *Agent-1*, *Agent-2*, and *Agent-3,* respectively. The first column corresponds to the initial velocity response, the second to the response after the third trial, and the third to the response after the final trial.

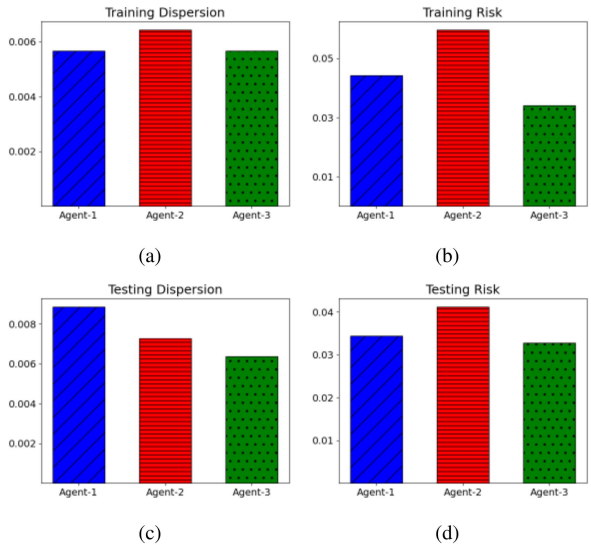


Fig. 11. Barplots of the reliability metrics during the training and testing phases. (a) and (b) Training dispersion and training risk. (c) and (d) Testing dispersion and testing risk.

final iterations of the last evaluation trial. Agent-3 converges faster and achieves a higher final reward value compared to the other two agents. The actual velocity response achieved after each trial is compared to the reference and displayed in Fig. 10. After the final trial, the duration of the leveling speed of the velocity response achieved by Agent-1 is slightly longer than the reference. The leveling speed achieved by Agent-2 is marginally higher than the desired one, whereas Agent-3 achieves an almost perfect match. We attribute the improved behavior of Agent-3 to the richer and more diverse set of episodes contained in the generated RL transition buffer $\mathcal{D}_{\text{gen}}$ that was used as training data for its initialization.

Moreover, to evaluate the effectiveness of the automated calibration methodology against the current manual calibration practice, we compare the performance of the three agents to

TABLE II
CALIBRATION MSE AFTER EACH TRIAL

	Agent-1	*Agent-2*	*Agent-3*	Operator
Trial 1	0.2494	**0.1822**	0.1916	-
Trial 2	0.2808	0.2230	**0.1608**	-
Trial 3	0.2149	0.2121	**0.1415**	-
Trial 4	0.1624	0.1786	**0.1471**	-
Trial 5	0.1826	0.1800	**0.1495**	0.3209

that of an experienced operator. Beginning from four randomly chosen initial states, the operator performs the necessary adjustments to calibrate the valve block. In Table II, the mse after different stages of the evaluation process between the actual velocity response and reference is reported for each agent and for the trained operator. We notice that Agent 3 consistently performs the best as it achieves significantly better results, whereas it also converges in fewer trials than the other two agents. Compared to the human operator, all three agents achieve significantly smaller mse which we attribute to the quantitative nature of the automated calibration process compared to the qualitative measures used during manual calibration. The velocity response achieving the smallest mse via manual calibration is shown in the third column of Fig. 10. It is evident that the operator slightly underestimates the value of the leveling speed.

C. Reliability Evaluation

Reliability composes a critical aspect in RL algorithms in order to ensure consistent performance, safety, and cost-effectiveness. In real-world industrial applications, RL agents must reliably operate over time to meet production requirements and maintain safety standards, as it is crucial to minimize the risk of unexpected failures or suboptimal performance. To evaluate and assess the reliability of the developed calibration methodology, we compute two measures proposed in [27], namely the *Dispersion* and *Risk* of the reward distribution.

Dispersion: Dispersion denotes the width of the distribution, reflecting the degree of spread around the mean or expected value. Hence, RL agents achieving narrower dispersion tend to perform more consistently. It can encapsulate the agent's sensitivity to several factors, such as random initialization of the environment and hyperparameter settings. To measure this spread, we utilize robust statistical methods and compute the inter quartile range

$$\text{IQR} = P_{75} - P_{25} \tag{12}$$

where P_{25} and P_{75} are the 25th and 75th percentiles of the reward distribution, respectively.

Risk: Risk refers to the magnitude and reach of the lower tail of the distribution. It complements dispersion measures, such as IQR that truncate the distribution's tails. To measure the distribution's risk, we use the conditional value at risk (CVaR) that is defined as

$$\text{CVaR}_a(R) = \mathbb{E}[R \,|\, R < P_a(R)] \tag{13}$$

where $a \in (0, 1)$ is a given parameter and P_a is the $a-$quantile of the distribution of the reward R. Concretely, $\text{CVaR}_a(R)$ quantifies the expected loss in worst-case scenarios.

To compare the reliability of the three agents, we analyze the dispersion and risk of their reward distributions during both the training and testing phases. The results are illustrated in Fig. 11. During training, Agents 1 and 3 demonstrate similar levels of dispersion, whereas Agent 2 performs notably worse. Regarding risk, Agent 3 significantly outperforms the others, which is crucial in industrial settings where sudden and steep performance drops during training could lead to equipment damage, impacting production. Conversely, Agent 2 exhibits the highest training risk. During the testing phase, Agent-1 displays the highest dispersion, followed by Agent-2, while Agent-3 is again the most consistent. Just like in the training stage, Agent-2 exhibits the highest risk, whereas Agent-3 performs the best. Overall, the reliability of Agent-3 is superior both in terms of dispersion and risk compared to the other two. We attribute this to the larger and more diverse dataset consisting of real and synthetic RL transitions that were used for offline, off-policy learning.

VI. Conclusion

In this article, an automated methodology to optimally configure the parameters of industrial machinery based on RL is developed and evaluated on a challenging elevator control valve block calibration use case. The displacement and velocity of the movement are estimated in real time using an SS model, whereas the parameters of the block are calibrated using deep Q-learning. Finally, to successfully handle the scarcity of industrial data and the existing exploration constraints, a generative model to create synthetic RL transitions is developed.

The efficacy of the proposed methodology is evaluated through the deployment of the developed system on-site. A minimal number of trials is needed to successfully perform the calibration of the EVs, whereas we also notice that the use of synthetic RL data substantially improves the performance of the process in terms of both accuracy and time efficiency. Finally, the soft sensing model produces accurate estimates of both the velocity and displacement in real time, thus avoiding the integration complexity and cost of a motion encoder.

A notable finding from this study is the enhanced effectiveness of RL agents through the utilization of synthetic data for offline, off-policy learning. Unlike classical methodologies, RL is oriented toward accomplishing high-level objectives and can therefore be seamlessly adapted to handle diverse tasks in different industrial domains. Leveraging the synergy between RL and generative modeling approaches holds substantial promise for industrial applications, particularly in scenarios characterized by data scarcity.

Some future directions include the integration of more advanced RL algorithms into the calibration methodology, the investigation of the effect that the choice of architecture has, and the adaptation of the proposed system on other industrial calibration use cases for further benchmarking.

Acknowledgment

The opinions expressed in this article are those of the authors and do not necessarily reflect the views of the European Commission.

References

[1] L. Leontaris et al., "A blockchain-enabled deep residual architecture for accountable, in-situ quality control in industry 4.0 with minimal latency," *Comput. Ind.*, vol. 149, Aug. 2023, Art. no. 103919.
[2] D. Powell, M. C. Magnanini, M. Colledani, and O. Myklebust, "Advancing zero defect manufacturing: A state-of-the-art perspective and future research directions," *Comput. Ind.*, vol. 136, Apr. 2022, Art. no. 103596.
[3] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, pp. 1140–1144, Dec. 2018.
[4] D. Silver et al., "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, Oct. 2017.
[5] A. Lukežic, T. Vojír, L. C. Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter with channel and spatial reliability," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 4847–4856.
[6] C. Gan, W. -H. Cao, K. -Z. Liu, and M. Wu, "Dynamic optimization-based intelligent control system for drilling rate of penetration (ROP): Simulation and industrial application," *IEEE Trans. Ind. Inform.*, vol. 20, no. 3, pp. 3695–3702, Mar. 2024.
[7] J. Chen, L. Lyu, Z. Fei, W. Xia, and X. -M. Sun, "Event-triggered adaptive robust control for a class of uncertain nonlinear systems with application to mechatronic system," *IEEE Trans. Ind. Informat.*, vol. 19, no. 12, pp. 11800–11808, Dec. 2023.
[8] S. -X. Wen, Z. -R. Pan, K. -Z. Liu, and X. -M. Sun, "Practical anti-windup for open-loop stable systems under magnitude and rate constraints: Application to turbofan engines," *IEEE Trans. Ind. Electron.*, vol. 70, no. 4, pp. 4128–4137, Apr. 2023.
[9] M. Wang, C. Zhao, J. Xia, and J. Sun, "Periodic event-triggered robust distributed model predictive control for multiagent systems with input and communication delays," *IEEE Trans. Ind. Inform.*, vol. 19, no. 11, pp. 11216–11228, Nov. 2023.
[10] T. Liu, S. Hao, Y. Wang, and J. Na, "Predictive state observer-based set-point learning control for batch manufacturing processes with delay response," *IEEE Trans. Ind. Electron.*, vol. 71, no. 1, pp. 788–797, Jan. 2024.
[11] G. Künzel, L. S. Indrusiak, and C. E. Pereira, "Latency and lifetime enhancements in industrial wireless sensor networks: A Q-learning approach for graph routing," *IEEE Trans. Ind. Inform.*, vol. 16, no. 8, pp. 5617–5625, Aug. 2020.
[12] H. Zhao, J. Zhao, J. Qiu, G. Liang, and Z. Y. Dong, "Cooperative wind farm control with deep reinforcement learning and knowledge-assisted learning," *IEEE Trans. Ind. Inform.*, vol. 16, no. 11, pp. 6912–6921, Nov. 2020.
[13] H. Dong and X. Zhao, "Wind-farm power tracking via preview-based robust reinforcement learning," *IEEE Trans. Ind. Inform.*, vol. 18, no. 3, pp. 1706–1715, Mar. 2022.
[14] X. Sun and J. Qiu, "A customized voltage control strategy for electric vehicles in distribution networks with reinforcement learning method," *IEEE Trans. Ind. Inform.*, vol. 17, no. 10, pp. 6852–6863, Oct. 2021.
[15] M. Dabbaghjamanesh, A. Moeini, and A. Kavousi-Fard, "Reinforcement learning-based load forecasting of electric vehicle charging station using Q-learning technique," *IEEE Trans. Ind. Inform.*, vol. 17, no. 6, pp. 4229–4237, Jun. 2021.
[16] S. Fan, X. Zhang, and Z. Song, "Imbalanced sample selection with deep reinforcement learning for fault diagnosis," *IEEE Trans. Ind. Inform.*, vol. 18, no. 4, pp. 2518–2527, Apr. 2022.
[17] J. Li, J. Ding, T. Chai, and F. L. Lewis, "Nonzero-sum game reinforcement learning for performance optimization in large-scale industrial processes," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 4132–4145, Sep. 2020.
[18] P. Zheng, L. Xia, C. Li, X. Li, and B. Liu, "Towards self-x cognitive manufacturing network: An industrial knowledge graph-based multi-agent reinforcement learning approach," *J. Manuf. Syst.*, vol. 61, pp. 16–26, Oct. 2021.
[19] P. D. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.
[20] I. Goodfellow et al., "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Oct. 2020.
[21] J. Lian et al., "Deep-learning-Based small surface defect detection via an exaggerated local variation-based generative adversarial network," *IEEE Trans. Ind. Inform.*, vol. 16, no. 2, pp. 1343–1351, Feb. 2020.
[22] Y. Gao, X. Liu, and J. Xiang, "FEM simulation-based generative adversarial networks to detect bearing faults," *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4961–4971, Jul. 2020.
[23] F. Kong, J. Li, B. Jiang, H. Wang, and H. Song, "Integrated generative model for industrial anomaly detection via bidirectional LSTM and attention mechanism," *IEEE Trans. Ind. Inform.*, vol. 19, no. 1, pp. 541–550, Jan. 2023.

[24] A. Howard et al., "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Seoul, South Korea, 2019, pp. 1314–1324.
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.
[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, Lille, France, 2015, pp. 448–456.
[27] S. C. Y. Chan et al., "Measuring the reliability of reinforcement learning algorithms," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–36.

Apostolos Evangelidis received the Diploma in electrical and computer engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2020.

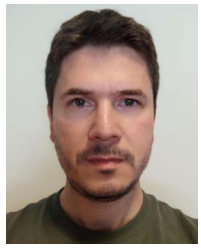
He has been a Research Assistant with the Information Technologies Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece, since 2021. His research interests include sparse representations, wavelet analysis, and machine learning.

Nikolaos Dimitriou (Member, IEEE) received the Diploma and Ph.D. degree in electrical and computer engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2007 and 2014, respectively.

Since 2014, he has been a Postdoctoral Research Associate with the Informatics and Telematics Institute, Centre for Research and Technology Hellas, Thessaloniki, Greece. His research interests include machine vision and deep learning with a focus on industrial applications.

Paschalis Charalampous received the Diploma degree in mechanical engineering, master's degree in processes & advanced materials technologies, and Ph.D. degree in manufacturing procedures from the Mechanical Engineering Department, Aristoteles University of Thessaloniki, Thessaloniki, Greece, in 2013, 2015, and 2018, respectively.

He has academic teaching experience with the Department of Mechanical Engineering in Aristoteles University of Thessaloniki, as well as in the International Hellenic University, Thermi, Greece. He has authored or coauthored more than 30 papers in international scientific journals and more than 20 papers in international conferences. His research interests include manufacturing engineering, material science, mechanical engineering design, and additive manufacturing procedures.



Theofilos D. Mastos received the BA degree in economics and international business from the University of Macedonia, Thessaloniki, Greece, in 2009, the M.Sc. degree in environmental management and policy from the University of the Aegean, Aegean, Greece, in 2013, and the Ph.D. degree in sustainable supply chain management from the University of Macedonia, Mitilini, Greece, 2024.

He spent 1 year as a visiting student with the University of Bristol, Bristol, U.K., and at the University of Greenwich, U.K. He is currently an EU Project Manager and Sustainability Researcher with KLEEMANN HELLAS SA, Kilkis, Greece. He is also an Acedemic Fellow with the International Hellenic University teaching Project Management at undergraduate level. He has participated in a number of conferences and seminars and he has authored or coauthored more than 15 scientific articles. His research interests include sustainable supply chain management and Industry 4.0 technologies, sustainable manufacturing and climate change adaptation and resilience planning in manufacturing, among others.

Dr. Mastos is a Member of the Economic Chamber of Greece and the European Operations Management Association.

Dimitrios Tzovaras (Senior Member, IEEE) received the Diploma and Ph.D. degrees in electrical and computer engineering from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1992 and 1997, respectively.

He is currently a Senior Researcher Grade A' and the President of the Board with the Centre for Research and Technology Hellas, Thessaloniki, Greece. His research interests include visual analytics, 3-D object recognition, search and retrieval, behavioral biometrics, assistive technologies, information and knowledge management, computer graphics, and virtual reality.