Exploiting radio access information to improve performance of remote-controlled mobile robots in MEC-based 5G networks

Winnie Nakimuli[a,*], Jaime Garcia-Reinoso[b], J. Enrique Sierra-Garcia[c], Pablo Serrano[a]

[a]*Departamento de Ingenieria Telematica, Universidad Carlos III de Madrid, Madrid,* 28911*, Spain*
[b]*Automatics Department, Universidad de Alcala, Spain*
[c]*Department of Electromechanical Engineering, University of Burgos, Spain*

Abstract

The Industry 4.0 paradigm aims to bring real-time production data analytics, cloud computing, and cyber-physical inter-connectivity to today's industries. This evolution fosters network-based use cases, such as remote-controlled mobile robots with stringent network latency requirements. 5G networks have become a promising technology for such use cases. However, 5G wireless communication is affected by time-varying random delays, which impact the use-case's reliability, stability, and performance. Thus, there is an urgent need to design mechanisms to compensate for these time-varying delays at the remote end. The delay estimation can be achieved by leveraging the MEC framework on top of 5G (MEC-based 5G). Thus, this work presents a novel architecture to exploit the radio network information provided by the MEC framework to improve the performance of remote-controlled mobile robots leveraging 5G. This radio network information is used to estimate the current network delays. Accordingly, these estimated delays together with the delayed information sent by the robot are availed to the robot controller at the remote end for compensation. Besides, the message-sequence flow between the different architecture components is analyzed in detail, and the modeling equations are described. Extensive simulations prove the effectiveness of the proposed approach. Our approach is compared with the network delay estimation based on the Kalman filter. An improvement of at least 55% and 33% in the tracking error and control effort, respectively, are observed for delay values $\geq 150ms$.

Keywords: Industry 4.0, 5G, MEC, Remote-Controlled Systems

1. Introduction

The fourth industrial revolution (i.e., Industry 4.0) is envisioned to provide: increased automation, interconnectivity between physical devices and computers (cyber-physical systems), and big data technologies to manufacturing industries and businesses [1]. These envisioned capabilities of Industry 4.0 have led to the creation and support of a myriad of use cases classified with stringent latency, reliability, and availability requirements. Due to the strict network requirements imposed by some of these use cases, 5G and some of its key enabling technologies

*Corresponding author
Email addresses: wnakimul@pa.uc3m.es (Winnie Nakimuli), jaime.garciareinoso@uah.es (Jaime Garcia-Reinoso), jesierra@ubu.es (J. Enrique Sierra-Garcia), pablo@it.uc3m.es (Pablo Serrano)

(i.e., Multi-access Edge Computing (MEC) and Network Function Virtualization (NFV)) have emerged as the fastest growing and most suitable solution in this arena [2, 3]. Furthermore, 5G, when enabled with the MEC technology (henceforth referred to as MEC-based 5G), comes with the availability of networking, computing, and real-time context information close to the edge of the network (i.e., within the Radio Access Network (RAN)) and closer to the end-user physical devices [4]. In addition, the MEC platform has been proposed to (i) reduce latency, (ii) improve bandwidth, (iii) offload processing functions, and (iv) support scalability in industry 4.0 use cases [5, 6]. Hence, this MEC paradigm has accelerated the deployment of remote-controlled industry 4.0 use cases, whose control loops impose stringent latency requirements. The controller components of these industry 4.0 use cases are located remotely from the physical devices. A communication network is utilized for communicating between the physical devices and the remote controllers. However, although there are enough reserved resources to accommodate the communication of all devices with the remote controller, this communication network introduces random network delays (i.e., jitter) affecting the system stability (since the stringent latency requirements for the control loops are no longer satisfied). This could lead to performance degradation and catastrophic effects depending on the considered remote-controlled industry 4.0 use case. Hence, it is of paramount importance that these network delays introduced by the communication networks are accurately estimated and compensated for at the remote controller end for such industry 4.0 use cases.

To address these requirements, the MEC platform offers a service called the Radio Network Information (RNI) service [7, 8]. This MEC service can provide authorized MEC applications with updated information about the prevailing radio network conditions (e.g., network delay, packet loss, and packet discard rate). These authorized MEC applications could be remote-controlled industry 4.0 use cases, for instance. Hence, they can use this up-to-date information to dynamically adapt their operations at the remote controller end to maintain the system stability and improve performance after changing conditions of the network. Furthermore, through the MEC platform, these authorized industry 4.0 applications can also either consume or provide specific services to other industry 4.0 applications [9].

This paper presents a 5G remote-controlled industry 4.0 use case leveraging the up-to-date radio network delay information provided by the MEC RNI (MRNI) API [8] to improve its path tracking performance in the presence of variable network delays. Specifically, we utilize a remote-controlled Automated Guided Vehicle (AGV) use case over a MEC-based 5G network. However, this up-to-date network delay estimation (NDE) data provided by the MRNI service (henceforth referred to as "NDE-MRNI approach") can also be exploited in other use cases. Accordingly, in this article, considering the relevant key performance indicators (KPIs) for the use case and the presence of random delays, we leverage the NDE-MRNI approach to estimate these network delays at the remote end. Thus, the estimated network delays are utilized to compute the actual current position of the AGV (since the received position information from the AGV is outdated due to the network delays). The remote controller component employs this updated position information to calculate the appropriate velocity commands to send to the AGV, improving the use case's path tracking performance.

To examine this NDE-MRNI approach, we implement a remote-controlled AGV use case whose controller (in this case, the Master programmable logic controller (Master PLC)) is virtualized and located remotely at the edge of the 5G network. On the other hand, the slave PLC is located inside the physical AGV. The 5G network is utilized for communicating between the virtualized Master PLC controller and the Slave PLC. This use case architecture is presented in Fig. 1. From Fig. 1, the AGV utilizes the information generated by the slave PLC to send its

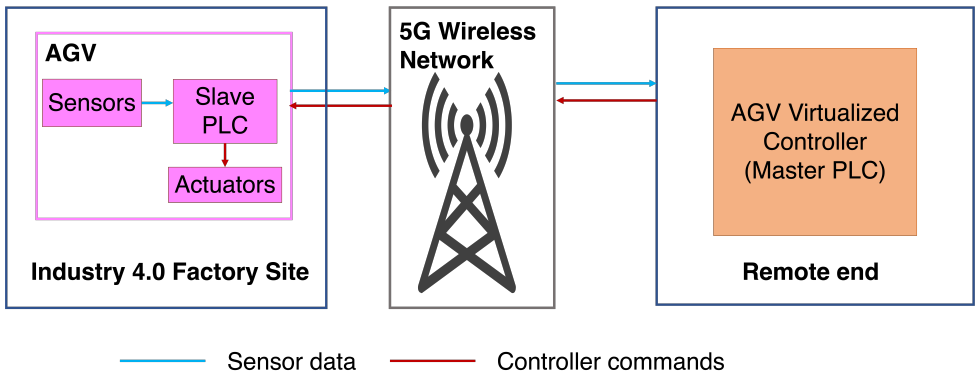


Figure 1: Design of a 5G Remote-controlled AGV use case

current position (included in the sensor data) by means of its 5G network card to the remote controller through the 5G network. Thus the remote controller generates the appropriate velocity commands given the desired AGV path trajectory. These controller commands are sent to the slave PLC component inside the AGV through the 5G network and applied by the actuators. Besides, the work done in this paper is motivated by our work in [10], where we analyzed the impact of random delays and packet losses on a remote-controlled AGV use case utilizing a real private 5G network.

To the authors' best knowledge, no studies focus on analyzing the effect of the NDE-MRNI approach in improving the path tracking performance of remote-controlled AGV use cases, henceforth, the significant contributions of this paper are manifold:

- Design and simulation of a model architecture for a 5G remote-controlled AGV use case, whose controller components are virtualized and located at the edge of the 5G network.
- Augmenting of this architecture by (i) deploying the MEC platform to provide the MRNI service, (ii) Exploiting the radio-access network delay information provided by the MRNI service to estimate the current network delays (i.e., the NDE-MRNI approach) at the remote end, and (iii) Leveraging these estimated network delays to update the AGV position information and thus employing this updated position information to compute the appropriate velocity commands at the remote controller end. Besides, this phase involved identifying the relevant key performance indicators (KPIs) to assess the effectiveness of the NDE-MRNI approach.
- Performance evaluation of this NDE-MRNI approach in improving the path tracking capabilities of the remote-controlled AGV use case, considering two different channel distributions and path trajectories. In addition, comparison of this performance improvement against the optimal baseline NDE technique (i.e., Kalman filter) utilized in remote-controlled AGV use cases.

The rest of this paper is organized as follows. Section 3 provides a description of the considered use case modeling. Section 4 describes the use case path tracking architecture when enabled with the MRNI service. In addition, a detailed description of how the NDE-MRNI approach is implemented is provided. Section 5 provides the different simulation approaches, including the

baseline approach and settings considered to evaluate the performance of the NDE-MRNI approach. Section 6 presents and discusses the results from the several experiments carried out to validate the NDE-MRNI approach under different simulation and channel parameters. Section 2 presents the related work. Finally, Section 7 provides a conclusion of this article while giving some direction for future works.

2. Related Work

Network time delay has been cited as the primary issue affecting remote-controlled systems that can lead to system instability and performance degradation [11]. To this end, several control methods have been proposed to handle time-delay in these kinds of applications [11]: Passivity-based, wave-variable-based, adaptive, robust, and neural network control approaches. However, the stability analysis of most of these approaches is limited by one or more of the following: lack of experimentation analysis (i.e., mainly numerical studies) due to controller algorithms' complexity [12] and constrained delays (i.e., either zero, constant, bounded, or limited random scope) [13]. The work in this paper differs from these approaches by (i) maintaining a simple controller algorithm and (ii) considering unconstrained time-varying network delays. On the other hand, the authors in [14] consider the noisy characteristics of the round trip time (RTT) signals of wireless communication channels (in particular, WiFi networks) to model and compensate for the network delays experienced in remote-controlled AGVs. Henceforth to compensate for these delays, they propose the Kalman filter approach to estimate the network delays and then compensate for them at the controller end. In this article, we add to this contribution in the following ways: (i) Leveraging a new approach in this arena, i.e., the NDE-MRNI approach to estimate the network delays, (ii) Moreover; this NDE-MRNI approach does not require any modifications to the physical AGV, and (iii) Comparing the performance of this NDE-MRNI approach (considering the relevant use case KPIs) with the Kalman filter approach proposed in previous works.

Furthermore, on the one hand, the MEC technology has been cited as one of the critical enablers of 5G; bringing the cloud environment to the edge of the network, with all the inherent advantages of this kind of approach, for example, lower latency and radio network condition context-awareness [15, 16, 17]. Moreover, Abbas et al. [18] identify the lack of context-aware applications as one of the primary barriers affecting time-sensitive applications typical of industry 4.0 use cases. The authors propose that combining 5G networks (which can provide the context information) and MEC technologies (capable of utilizing this context information) is the ultimate solution for realizing and improving time-sensitive applications. On a similar note, the authors in [19] evaluate the performance gains of context information availability in enhancing edge robotics. They analyze the perks of context information provided via a WiFi network to an edge application in improving robot motion through experimental analysis. However, as commented by the authors, the proposed control algorithm is straightforward and unrealistic, i.e., It only depends on the signal level of the received WiFi signal to decide the robot speeds. On the other hand, this paper employs a practical controller algorithm to evaluate the efficacy of the MEC-based approach (i.e., NDE-MRNI approach) in improving path tracking performance for a 5G remote-controlled AGV use case affected by time-varying network delays. Lastly, the authors in [20] apply reinforcement learning to control an autonomous guided vehicle from an edge cloud through a wireless communication network affected by fading. The authors find the optimal AGV speed that maintains system stability with reinforcement learning while reducing the mission time to complete tasks. However, the authors only consider the DL channel error-and

delay-prone and assume that the UL channel is error- and delay-free. Thus, the AGV commands arrive at the remote controller instantaneously. However, in our paper, we consider both the DL and UL delays and compensate for them at the remote controller end. In addition, apart from the time taken KPI considered in this paper, we also examine two other critical use case KPIs, i.e., MAE and Control Effort, when evaluating the performance of the NDE-MRNI approach.

3. Use Case Modeling

This section presents an overview of the MEC framework, focusing on the relevant MEC functionalities for our scenario. Furthermore, we provide the kinematic model and controller algorithm for the considered AGV.

3.1. MEC Framework

Fig. 2 presents the block diagram of the functional entities of the MEC framework [7] utilized in our scenario. From Fig. 2, we can see that this framework mainly comprises two levels, i.e., the MEC system and host levels. The MEC system level is primarily composed of the MEC orchestrator, maintaining an overall view of the MEC system. This MEC orchestrator is in charge of selecting the appropriate MEC hosts to instantiate the MEC applications based on the required network, compute, and storage resources. Furthermore, the MEC orchestrator is responsible for instantiating and terminating these MEC applications by contacting the requisite virtualized infrastructure manager (VIM). On the other hand, the MEC host-level consists of the MEC hosts and their managers. In particular, the MEC host comprises (i) Virtualization infrastructure; controlled by the VIM and is in charge of hosting the virtualized instances of the MEC applications, (ii) MEC platform; responsible for providing the required functionalities for MEC applications to discover, advertise, provide and consume the MEC services' data, and (iii) MEC applications; to consume and provide MEC services' data from/to the MEC platform. The MEC platform and the MEC application interact via the Mp1 reference point. Besides providing specific service functionalities to authorized MEC applications, this Mp1 reference point is also used for service registration, service discovery, and communication support. Moreover, the MEC platform contacts the data plane of the virtualization infrastructure through the Mp2 reference point to route traffic between the different MEC services and applications.

The MEC platform is controlled by the MEC platform manager, which is responsible for monitoring the lifecycle of the MEC applications and communicating with the VIM to receive performance statistics & fault records about the instantiated MEC applications and acts accordingly. Moreover, the MEC platform provides the following services [7, 21]:

- **MEC Radio Network Information (MRNI) Service:** This service provides updated information about (i) The radio network conditions (e.g., delay, throughput, and packet discard rate), (ii) User plane measurements and statistics, and (iii) UE context and their radio access bearers, to authorized MEC applications [8]. This radio network information can be used by authorized MEC applications to optimize their operations dynamically. Besides, this MRNI service data can be provided at a specified granularity (i.e., per data radio bearer (DRB) per UE, per DRB per cell, and at every specified period depending on the mobile network operator). The authorized MEC applications can obtain the information about the prevailing radio network conditions using the RNI application programming interface (API) [8]. For this paper, the focus radio channel measurement is network delay; since network delay is the most critical factor affecting remote-controlled use cases that leads

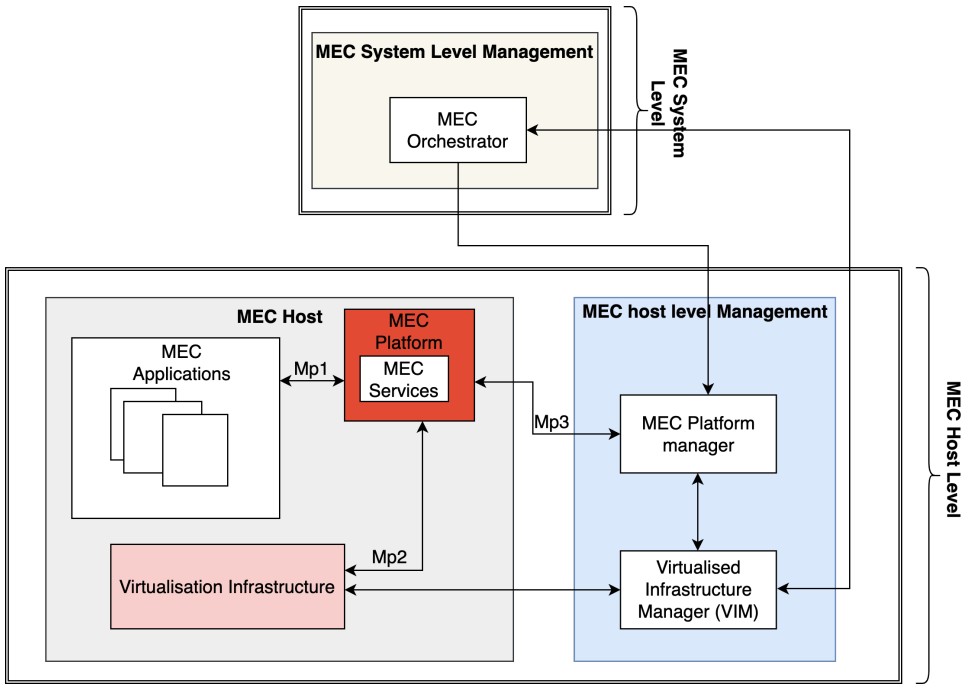


Figure 2: Block diagram of the MEC framework focusing on the entities required for the considered remote-controlled AGV use case

to system instability and performance degradation [11]. This MRNI service can provide regular estimates of the average uplink (UL) and downlink (DL) delays experienced by the remote-controlled AGV in a specified period. This period and the frequency at which these delay measurements are provided is determined by the mobile network operator, depending on the agreed-upon service level agreement (SLA) with the use case vertical provider. More information about how the MRNI service estimates the UL and DL delays can be found in [21].

- **MEC Location Service:** This MEC service provides location-related information to authorized MEC applications. For a specified MEC host, the location information can include: (i) location of all the served UEs by the radio node, and (ii) location of all the associated radio nodes.

- **MEC Bandwidth manager Service:** With this service, specific traffic from and to authorized MEC applications can be prioritized and allocated specific bandwidth.

Moreover, in this paper, we are only utilizing the MRNI service and not the other two MEC services; since its the most relevant for the considered remote-controlled AGV use case. However, for more complex industry 4.0 use cases, the location and bandwidth manager MEC services would also be very useful in improving performance. In addition, only the network delay information was extracted from the MRNI service and not the packet loss or other radio network conditions due to our work in [10]. In that article, we evaluated a remote-controlled AGV use

case over 5G in the presence of random delays and packet losses. From those experiments, we concluded that network delay has a notable impact on the use case performance, whereas for packet losses $\leq$ 35%, the performance impact is negligible. Consequently, in this article, we focus on assessing the improvement in path tracking due to utilizing the up-to-date network delay information provided by the MRNI service in remote-controlled AGV use cases affected by random network delays.

3.2. AGV MODELING

3.2.1. AGV Kinematic Model

The AGV considered in these experiments is a two-wheeled differential drive robot with a caster wheel at the front for maneuverability purposes, as illustrated in Fig. 3. From Fig. 3, the robot coordinate frame is defined by X_R and Y_R, whereas X_G and Y_G represent the global or world coordinate frame and θ is the angle between the robot coordinate frame and the global coordinate frame. The AGV position (pose) in the global coordinate frame is given by $pose = (X, Y, \theta)$. Assuming that the robot is moving in the forward direction along the X_R direction with a linear velocity (v) and angular velocity (ω); the velocity components ($\dot{x}$, $\dot{y}$, and $\dot{\theta}$) of the robot in the global coordinate frame in reference to Fig. 3 are given as follows [22]:

$$\dot{x} = vcos(\theta) \tag{1}$$

$$\dot{y} = vsin(\theta) \tag{2}$$

$$\dot{\theta} = \omega \tag{3}$$

The AGV velocities, i.e., v and w, are found through measurements and are related to the rotational wheel speeds of the AGV as follows:

$$v = \frac{r}{2}(\omega_R + \omega_L) \tag{4}$$

$$\omega = \frac{r}{l}(\omega_R - \omega_L) \tag{5}$$

Where l is the wheelbase, i.e., the distance between the left (L) and right(R) AGV wheels, and r represents the wheel radius as indicated in Fig. 3. ω_L and ω_R are the angular velocities of the left and right wheel respectively.

Using matrix notation, we can re-write (1), (2), and (3) as (6), which represents the forward kinematic model of a two-wheeled differential drive robot. It is crucial to note that this model assumes that the velocity of the robot in the Y_R direction in the robot coordinate frame is zero, and the robot is only moving forward in the X_R direction.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6}$$

At any time instant, we can keep track of the robot's position by applying numerical integration techniques to the kinematic model of the robot given in (6). In this case, the chosen numerical integration technique is the second-order Runge-Kutta integration technique[23]. Accordingly, at a discrete time instant k, given the current pose $pose_k = (x_k, y_k, \theta_k)$ and the robot

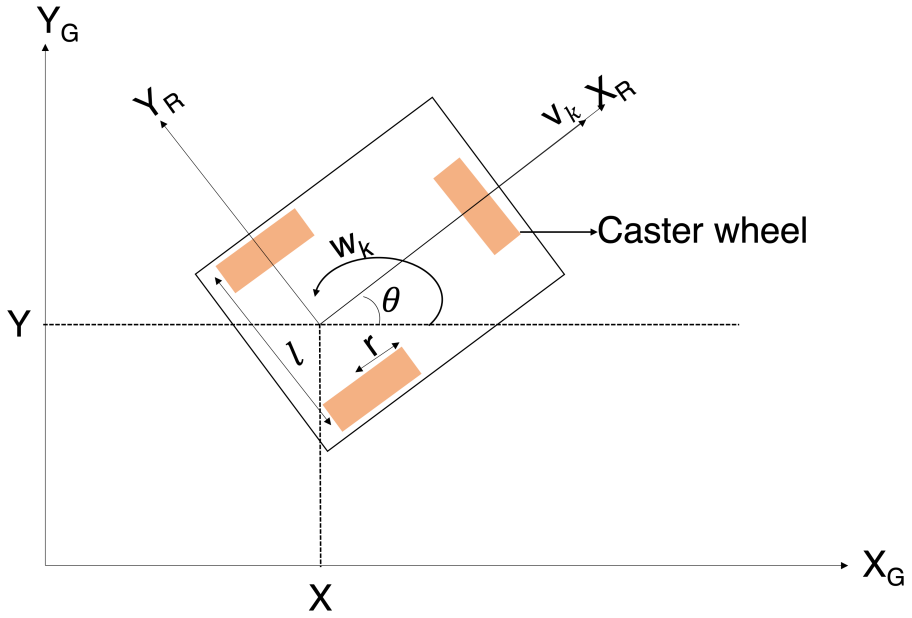


Figure 3: AGV Schema

control inputs (i.e., v_k and ω_k), after the elapse of a sampling period equal to ΔT, we can approximate the AGV pose at time instant $k+1$, i.e., $pose_{k+1} = (x_{k+1}, y_{k+1}, \theta_{k+1})$ in the global coordinate frame as:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} cos(\theta_k + \omega_k * \Delta T/2) & 0 \\ sin(\theta_k + \omega_k * \Delta T/2) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} \Delta T \tag{7}$$

This process of continuous integration of the kinematic model of the robot based on the knowledge of a fixed reference point and the issued control commands to localize the AGV position is called odometric localization or dead-reckoning. This dead-reckoning process is subject to errors over time, which become significant over long distances. These errors are caused by inaccurate calibration of the AGV parameters (for example, the wheel radius), wheel slippage, and numerical integration errors. However, these errors can be significantly reduced by keeping the AGV velocities low and periodically restarting the dead reckoning process based on a known reference point [22].

3.2.2. AGV controller algorithm

In order to test the potency of the NDE-MRNI approach, for the AGV controller algorithm, we chose the pure-pursuit algorithm due to its satisfactory performance, ease of implementation, and widespread use [24, 25]. In addition, this controller was used by vehicles in the DARPA Grand challenge [26] and the DARPA Urban challenge [27]. This geometric path tracking algorithm works by calculating the curve's curvature needed to move the AGV from its current position to a defined future position along the desired path trajectory. This defined future position is also known as the look-ahead distance and is one of the main parameters of this con-

troller. Hence, when provided with the current pose information and the appropriate look-ahead distance, this controller computes the proper angular velocity commands to move the AGV along the desired trajectory in the next sampling period.

4. LEVERAGING THE NDE-MRNI APPROACH IN REMOTE-CONTROLLED AGVs

This section presents the use case architecture when augmented with the relevant MEC service (i.e., the MNRI service) to provide the NDE-MRNI approach. Furthermore, this section details how this NDE-MRNI approach is designed and implemented in our scenario. In addition, the message flows between the different use case components is also provided.

4.1. Augmented Use Case architecture with the NDE-MRNI Approach

The remote-controlled AGV use case design presented in Fig. 1 can be augmented with the relevant MEC platform functionalities presented in Fig. 2 to improve its path tracking performance in the presence of variable delays (i.e., jitter). Accordingly, the use case design and architecture, when enabled with the MRNI service to avail the NDE-MRNI approach, is presented in Fig. 4. On the one hand, the design comprises the physical AGV at the factory site. On the other hand, the design comprises the MEC host at the network's edge. The MEC host contains: (i) the MEC platform, providing the MRNI service that offers the NDE-MRNI approach, and (ii) the relevant MEC applications for the use case (i.e., NDE-MRNI consumer, Position predictor, and the Master PLC vController). From Fig. 4, we notice that the virtualization infrastructure component of the MEC host; in charge of hosting the virtualized instances of the relevant MEC applications for the use case, has been omitted; this is because we are considering the simulated use case architecture. Otherwise, this component is pivotal to the use case in the real scenario.

The 5G-RAN block represented by the DL and UL channels is responsible for the communication between the remote Master PLC vController and the physical AGV. The MRNI service receives the UL and DL channel delay information from the 5G RAN block at the MEC host. The MRNI service uses this delay information from the 5G RAN to provide the most up-to-date estimates of the current UL and DL delay values (i.e., the NDE-MRNI approach). It is important to note that the MEC platform is co-located with the 5G-RAN, so these network delay values from the RAN are available almost instantaneously to the MEC platform and thus to the MRNI service. On the other hand, the "NDE-MRNI consumer" MEC application registers and subscribes to the MEC platform via the Mp1 reference point as a consumer of the data provided by the NDE-MRNI approach. Once authorized by the MEC platform, this MEC application consumes the NDE-MRNI data at every sampling instant and sends these data to the position predictor MEC application. This position predictor application uses these data to compute an update of the AGV pose (since the received AGV pose information is outdated due to the presence of UL delays). After that, this component forwards this new estimated pose to the "Master PLC vController" application. This MEC application employs this updated pose data to compute the new AGV velocity commands for the current sampling period. These new velocity commands are sent to the Slave PLC through the 5G network. However, these new velocity commands will also experience another random delay (i.e., DL delay) before reaching the physical AGV. Finally, the delayed velocity commands are received by the AGV and are in turn applied by the AGV actuators.

Moreover, in the Master PLC vController component, the desired AGV path trajectory is provided as a set of waypoints in the x and y directions. This controller component uses the

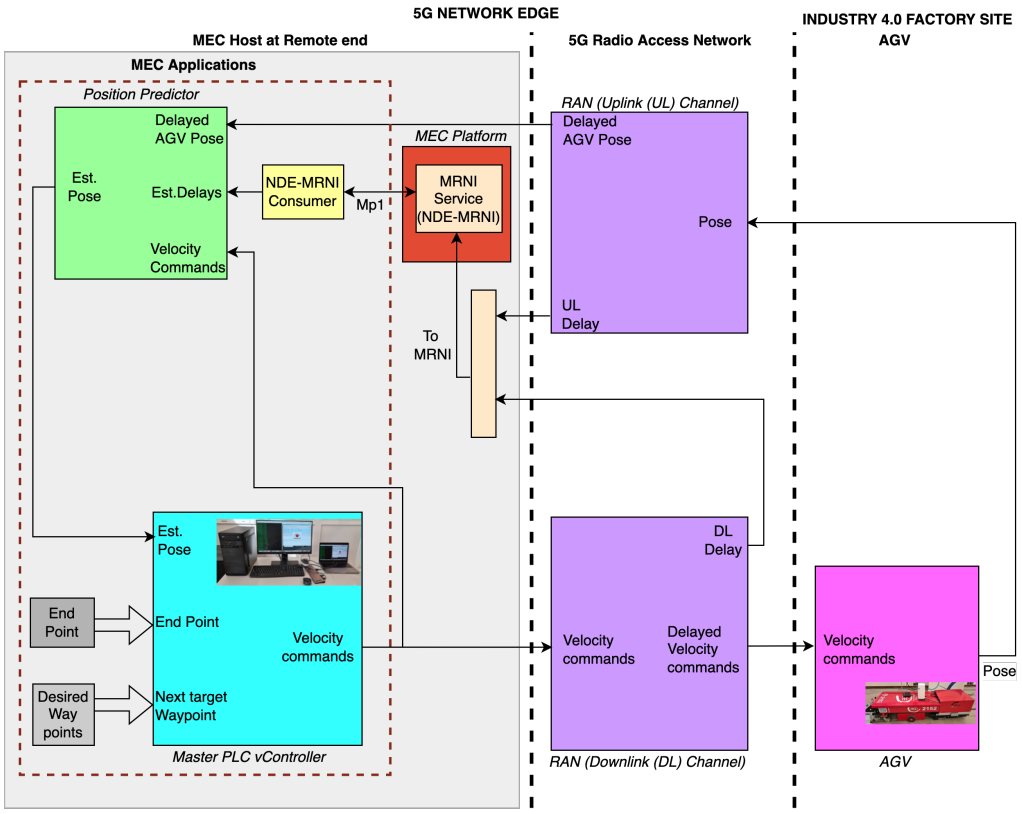


Figure 4: Design and architecture of the Remote-Controlled AGV Use Case augmented with the NDE-MRNI approach

next target waypoint and the estimated pose (from the Position predictor component) to generate the velocity commands of the AGV. Finally, to ensure that the AGV stops at the final waypoint, we keep track of this "*End-point.*" Each time the distance to this point is less than a specified threshold, the AGV stops.

Besides, this kind of design architecture has been proposed and leveraged for several other network-controlled use cases [28, 29]. Our main contribution to this architecture is augmenting the delay estimation with the NDE-MRNI approach of the MEC platform. Moreover, we use simulations to test the potency of this NDE-MRNI approach, i.e., improving the path tracking of remote-controlled AGV use cases experiencing random network delays; due to the ease of testing different networking conditions with simulations. However, we plan to examine the presented use case scenario using physical hardware and software in our future work.

4.2. Implementation of the NDE-MRNI approach and updated pose estimation for the considered AGV Use Case

This section explains how the NDE-MRNI approach is implemented in our scenario. In addition, we indicate how these data are utilized to update the pose estimation at the remote end, hence improving the path tracking performance of the considered remote-controlled AGV

use case. Besides, the message flow between the different use case architecture components presented in Fig. 4 is illustrated in Fig. 5.

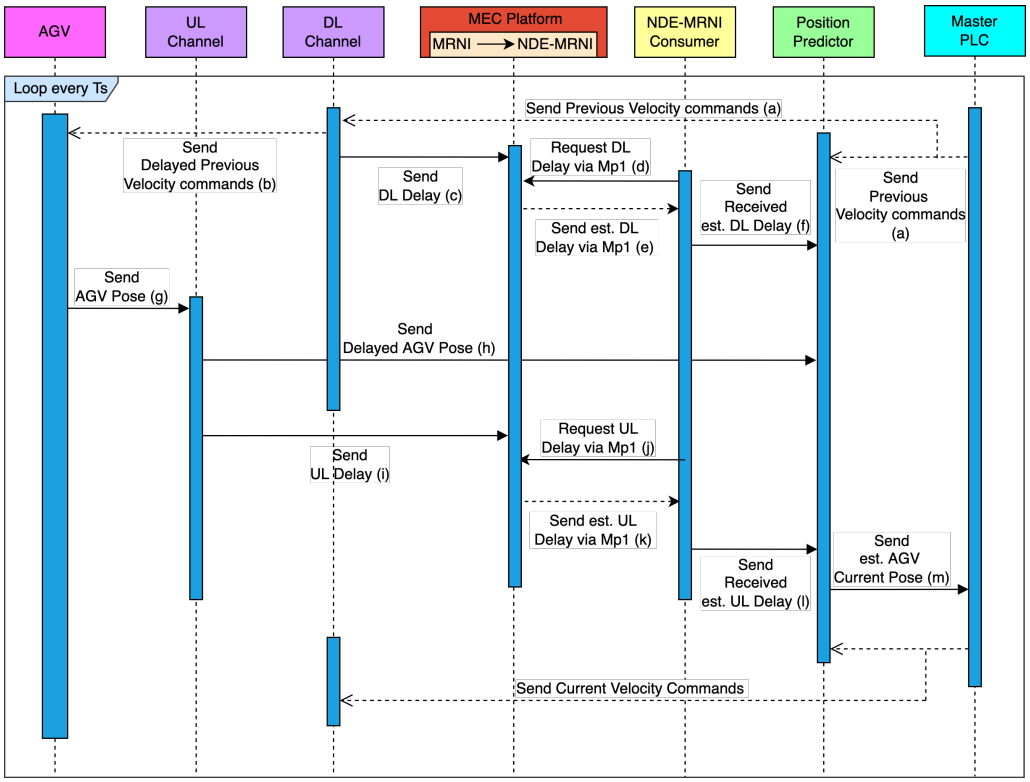


Figure 5: Message sequence flow in a remote-controlled AGV use case leveraging the NDE-MRNI approach

From Fig. 5, at the start, the Master PLC sends the previous velocity commands (i.e., from the previous sampling instant) to (i) the position predictor component and (ii) the AGV through the DL channel. The DL channel adds a random DL delay to the previous velocity commands. This DL delay value is sent to the MRNI service by the RAN. Subsequently, the "NDE-MRNI consumer" application requests for the DL delay value from the MRNI service of the MEC platform via the Mp1 reference point. Upon receiving this request from the "NDE-MRNI consumer", the MRNI service computes an estimate of the current DL delay value. In our implementation scenario, the MRNI service estimates the current DL delay value by utilizing the previously received DL delay values as follows:

$$\widehat{T}_{k,DL} = mean\{T_{k-1,DL}, T_{k-2,DL}, ..., T_{k-n,DL}\} \tag{8}$$

where $\widehat{T}_{k,DL}$ represents the estimated DL delay at the current sampling instant k and $T_{k-n,DL}$ represents the received DL delay value at the previous nth sampling instant. The optimal value of n i.e., the number of previously received delay samples to use to estimate the current delay is determined through experimental trials. Finally, the MRNI service forwards this estimated DL delay (i.e., $\widehat{T}_{k,DL}$) to the "NDE-MRNI consumer" MEC application. This MEC application forwards the estimated DL delay (i.e., $\widehat{T}_{k,DL}$) to the position predictor component.

As all these processes are ongoing, depending on the DL delay value, at every sampling period, the AGV has either received the velocity commands that the Master PLC vController sent, or it continues moving using the last received velocity commands. The AGV utilizes either of these velocities at every sampling period to compute its current pose.

The AGV proceeds to send this pose information to the UL channel. The channel adds a random UL delay and sends the delayed pose to the position predictor component after the delay period has expired. Secondly, the previous UL delay data is sent to the MRNI service by the RAN. The MRNI service receives this UL delay value and estimates the current UL delay (i.e., $\widehat{T}_{k,UL}$); in our scenario, $\widehat{T}_{k,UL}$ was estimated using (9). This estimated UL delay is dispatched upon request to the "NDE-MRNI consumer" application through the Mp1 reference point. This MEC application forwards the received $\widehat{T}_{k,UL}$ to the position predictor component.

$$\widehat{T}_{k,UL} = mean\left\{T_{k-1,UL}, T_{k-2,UL}, ..., T_{k-n,UL}\right\} \tag{9}$$

where $\widehat{T}_{k,UL}$ represents the estimated UL delay at the current sampling instant k and $T_{k-n,UL}$ represents the received UL delay value at the previous *nth* sampling instant.

Furthermore, at the position predictor component, the previously received $\widehat{T}_{k,DL}$ is used to estimate the current velocities of the AGV. These velocities are estimated by maintaining two velocity buffers, i.e., v_{buff} and ω_{buff} at the position predictor component for the linear and angular velocities, respectively. At every sampling instant, these buffers receive two inputs, i.e., the velocities ($v(t)$ and $\omega(t)$) sent by the Master PLC controller to the AGV and the previously received $\widehat{T}_{k,DL}$. In addition, these buffers also store the simulation time (t) at which these two inputs are received. Henceforth, the AGV velocities at the current sampling instant are estimated using (10) and (11):

$$\widehat{v}_k = v_{buff}(\widehat{T}_{k,DL}) : R \rightarrow R \tag{10}$$

$$\widehat{\omega}_k = \omega_{buff}(\widehat{T}_{k,DL}) : R \rightarrow R \tag{11}$$

where $\widehat{v}_k$ and $\widehat{\omega}_k$ are the estimated linear and angular velocities, respectively. Whereas v_{buff} and ω_{buff} are the buffers for the linear and angular velocities, respectively. Furthermore, $v_{buff}(\widehat{T}_{k,DL}) : R \rightarrow R$ and $\omega_{buff}(\widehat{T}_{k,DL}) : R \rightarrow R$ represent functions to estimate the velocities at the current sampling instant k, by accessing the velocity buffers with the estimated current DL delay (i.e., $\widehat{T}_{k,DL}$) as the input. In our case, these buffers implement the following functions to estimate $\widehat{v}_k$ and $\widehat{\omega}_k$ respectively:

$$v_{buff}(\widehat{T}_{k,DL}) : R \rightarrow R = v(t - \widehat{T}_{k,DL}) \tag{12}$$

$$\omega_{buff}(\widehat{T}_{k,DL}) : R \rightarrow R = \omega(t - \widehat{T}_{k,DL}) \tag{13}$$

where t is the current simulation time.

Upon receiving the delayed AGV pose information, the position predictor component uses (i) this delayed pose information (x_d, y_d, θ_d), (ii) the estimated UL delay value ($\widehat{T}_{k,UL}$) from (9), and (iii) the estimated velocity commands (from (10) and (11)) to estimate the current AGV position using (14). It is important to note that (14) is the equivalent to (7) when all the estimated values have been substituted correctly. Accordingly, this new estimated AGV position is sent to the Master PLC vController. This component utilizes this value to compute the AGV velocity

commands for the next sampling period, and the loop starts all over again. However, it is crucial to note that these velocity commands sent by the remote controller to the AGV are not applied as-is by the AGV but rather are limited by the maximum acceleration and deceleration in the AGV. Additionally, with the NDE-MRNI approach, the AGV only needs to send the pose information variable.

$$\begin{bmatrix} \widehat{x_k} \\ \widehat{y_k} \\ \widehat{\theta_k} \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix} + \begin{bmatrix} cos(\theta_d + \widehat{\omega}_k * \widehat{T}_{k,UL}/2) & 0 \\ sin(\theta_d + \widehat{\omega}_k * \widehat{T}_{k,UL}/2) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \widehat{v_k} \\ \widehat{\omega_k} \end{bmatrix} \widehat{T}_{k,UL} \tag{14}$$

where (x_d, y_d, θ_d) is the delayed AGV pose, and the rest of the symbols are the same as earlier defined.

Furthermore, we approximate the size of the messages exchanged between the different components presented in Fig. 5, and the results are provided in Table 1. For messages (a), (b), (g), and (h), the size was approximated in our previous work [10], where we captured the network packets (utilizing the tcpdump[30] tool) sent from the Master PLC to the AGV and vice versa leveraging a real-world 5G channel. The size of messages (d), (e), (j), and (k) is approximated by employing the ETSI MEC sandbox environment [31]. By using the MRNI API service endpoints [8], we send requests for layer 2 measurements (in this case, network delay) to this sandbox environment and receive responses. Accordingly, we analyze the sizes of the request and response messages. The sizes of the remaining messages, i.e., (c), (f), (i), (l), and (m), were duly deduced.

Table 1: Approximate message sizes between the use case components presented in Fig. 5

Message number	Headers size (Bytes)	Body size (Bytes)	Total size (Bytes)
(a) -> (b) (Master PLC -> DL channel -> AGV)	42	20	62
(g) -> (h) (AGV -> UL channel -> Position predictor)	42	20	62
(d), (j)	358	0	358
(e), (k)	213	1380	1593
(c), (i)	<213	<1380	<1593
(f), (l)	>213	1380	>1593
(m)	42	20	62

5. EXPERIMENT PREPARATION

This section discusses the assessed use case KPIs, and the additional simulation approaches used to compare against the NDE-MRNI approach presented in this paper. Besides, the AGV path trajectories used for the analysis are also presented. It is important to note that the design and simulations of our experiments were implemented with MATLAB (version 2020b) and Simulink software [32].

5.1. Evaluated Use Case KPIs

The following KPIs were examined to evaluate the performance of the NDE-MRNI approach:

5.1.1. Mean Absolute Error (MAE)

This KPI was computed as the mean of the shortest distance between the received AGV pose information and the desired AGV trajectory. In particular, for each received AGV pose information, we compute the shortest distance between that point to each of the lines on the desired AGV trajectory; and the minimum of these distances is the absolute error. In order to make the best use of space, AGVs must travel as close as possible to the desired AGV trajectory, i.e., the MAE KPI must be as low as possible. This allows optimizing the routes and the available space when AGVs share the workspace with other AGVs, manned transport vehicles, or humans.

5.1.2. Time taken (t_t)

This KPI is the total time taken by the AGV to complete the path trajectory in seconds(s). It is computed as:

$$t_t = T_{end} - T_{start} \tag{15}$$

where T_{end} is the time when the last AGV pose information variable is sent to the Master PLC, and T_{start} is the time when the first AGV pose variable is sent. In logistics, it is vital to reduce the time taken KPI, i.e., the time needed to transport assets. If we reduce this KPI, we will transport more assets in the same period. Thus, smaller values of the time taken KPI provides enormous profitability and a shorter investment return.

5.1.3. Control Effort (C_{eff})

This KPI is the measure of how much energy the controller expends to move the AGV along the desired path [33]. We computed this KPI using the controller output (i.e., the angular velocities output by the controller), and the average control effort was calculated as:

$$C_{eff} = \sqrt{1/(T_{end} - T_{start}) * \int_{T_{start}}^{T_{end}} \Omega_i^2 \, dt} \tag{16}$$

where C_{eff} is the control effort, Ω_i is the output angular velocity by the controller at the sampling instant i in rad/s. Whereas $T_{end} - T_{start}$ is the total time taken by the AGV to traverse the entire AGV path trajectory in seconds(s). This total time is split into equal sampling instants, and each sampling instant i has a corresponding output angular velocity. This KPI provides a measurement of the mechanical components' degradation speed. If we want to increase the lifecycle of the AGVs, we need to keep the control effort KPI as low as possible. Furthermore, this KPI also gives us information on the energy consumption by the AGV, i.e., the bigger the control effort KPI, the larger the energy consumption by the AGV.

5.2. Additional Simulation Approaches

In our analysis, to evaluate the performance of the NDE-MRNI approach in improving path tracking performance of a remote-controlled AGV use case, we also considered the following simulation approaches:

5.2.1. Baseline approach (NDE-Kalman filter)

At every sampling instant, the AGV calculates the total delay experienced in the loop in the previous sampling instant (i.e., from the AGV to the remote controller and back to the AGV) on top of the pose variable. This total delay is also known as the previous round trip time (RTT), is computed by the AGV as:

$$\widehat{\tau}_{k-1} = T_{k-1,Ack} - T_{k-1,Tx} \tag{17}$$

Where $\widehat{\tau}_{k-1}$ is the previous RTT, $T_{k-1,Tx}$ is the time the AGV sends the pose information variable, and $T_{k-1,Ack}$ is the time the AGV receives a response from the remote controller. Moreover, with this information about the previous RTT, several approaches have been proposed to estimate the current RTT. For this article, we focus on one such baseline approach, i.e., the Kalman filter NDE approach, which has been shown to yield the most promising results compared to the other baseline NDE approaches [14]. In addition, for these baseline approaches, to estimate the current UL delay ($\widehat{T}_{k,UL}$) and DL delay ($\widehat{T}_{k,DL}$), as it is not possible to know the contribution of the UL and DL delays to the RTT, a typical approach is to divide the estimated RTT by 2; which produces less accurate results compared to the NDE-MRNI approach.

The Kalman filter approach is capable of providing optimal estimates when provided with noisy input measurements [14]. Thus to employ the Kalman filter approach to estimate the current RTT, we have to analyze the RTT signal of the 5G channel. To perform this analysis, we set up several experiments to observe the actual RTT signals of the channel over a real private 5G network. From these tests, we noticed that the RTT signal of the 5G channel is smooth with some high-frequency components with a shape approaching the Weibull and Gamma probability distributions. As a result, the RTT signal of the 5G network can be modeled as a noisy signal, and the Kalman filter can be used to provide optimal RTT estimates [34, 14]. Accordingly, the current RTT ($\widehat{\tau}_k$) was estimated by utilizing the Kalman filter algorithm implementation in Matlab [35].

Moreover, the Kalman filter parameters were chosen optimally through experiments. The parameters that yielded the best results for our scenario were selected: the initial guess of the a posteriori RTT estimate was set to 0 and the initial error covariance to 1. The measurement noise covariance was fixed to 400, and the process noise covariance was set to 0.0001. It is important to note here that the accuracy of the Kalman filter approach depends highly on these filter parameters; conversely, this is not a concern for the NDE-MRNI approach. Besides, with the Kalman filter approach, the AGV needs to send another information variable, i.e., the previous RTT ($\widehat{\tau}_{k-1}$) on top of the pose variable, in contrast, for the NDE-MRNI approach, the AGV needs to send only the pose information variable.

Subsequently, this estimated RTT ($\widehat{\tau}_k$) by the Kalman filter approach is used by the position predictor component to estimate the current UL delay ($\widehat{T}_{k,UL}$) and DL delay (($\widehat{T}_{k,DL}$)) as follows:

$$\widehat{T}_{k,DL} = \frac{\widehat{\tau}_k}{2}, \quad \widehat{T}_{k,UL} = \frac{\widehat{\tau}_k}{2} \tag{18}$$

where $\widehat{\tau}_k$ is the estimated RTT at the current sampling instant k. Next, the estimated DL delay ($\widehat{T}_{k,DL}$) is used to estimate the current linear ($\widehat{v}_k$) and angular velocities ($\widehat{\omega}_k$) using (10) and (11), respectively. Finally, the estimated $\widehat{T}_{k,UL}$, and velocities ($\widehat{v}_k$ and $\widehat{\omega}_k$) are employed to estimate the current AGV position using (14). Finally, this new estimated pose variable is sent to the "Master PLC vController" component, which uses it to compute the AGV velocity commands for the next sampling period. This simulation scenario is the optimal state-of-the-art NDE approach used in our experiments to compare with the NDE-MRNI approach.

5.2.2. No network delay (Ideal-No delay)

This simulation scenario is the ideal case with no channel delays between the AGV and the remote Master PLC vController in the MEC host. This simulation scenario is used as the benchmark for all the other experiments.

5.2.3. No Network delay estimation (No-NDE)

In this case, there are channel delays (i.e., UL and DL delays) between the AGV and the remote Master PLC vController; however, there is no network delay estimation carried out at the MEC host. This scenario is used as the worst-case scenario, i.e., there are network delays, but no network delay estimation approach is applied at the remote controller end.

5.2.4. Ideal Network delay estimation (NDE-Ideal)

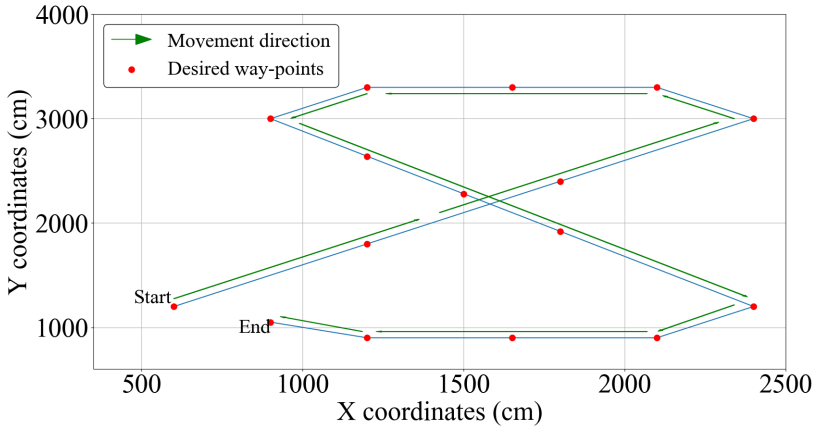
For this setting, there are also channel delays between the AGV and the remote Master PLC vController in the MEC host. However, in this scenario, the precise network delays (i.e., UL and DL) experienced in the loop between the AGV and the Master PLC vController are used to estimate the current pose of the AGV. This setting corresponds to the ideal or best-case NDE scenario where the delay estimation is accurate at the remote controller end. Nonetheless, these accurate network delay values are only attainable in simulation scenarios.

5.3. Assessed AGV Path Trajectories

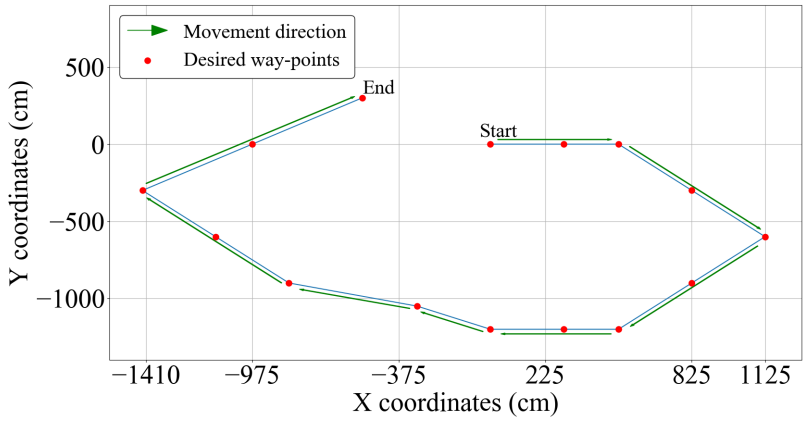
To analyze the path tracking performance of the remote-controlled AGV use case with the NDE-MRNI approach, we utilized two commonly used AGV path trajectories, i.e., 8-Shape and the Irregular hexagon shape (Irr-hex-Shape) [14]. These two trajectories were chosen to test the AGV maneuverability in a wide range of possible movement scenarios, i.e., straight lines of different lengths, cyclic rotations, curves with variable bending radius, unexpected bends of varying magnitudes in either direction (i.e., right and left). Besides, the 8-Shape trajectory is one of the most used trajectories in industry factory settings [10, 36]. Conversely, the Irr-hex-Shape (one of the many variations of the hexagon path trajectory) has been tested and widely applied in path planning experimentation due to the hexagon shape being more representative of natural curves typical of industry settings than square trajectories [14, 37, 38] . These two path trajectories are presented in Fig. 6a and Fig. 6b, respectively. Moreover, the start and endpoints of the AGV together with the movement direction have been indicated.

6. RESULTS AND DISCUSSION

This section presents and discusses the results from our experiments, comparing the path tracking performance of the NDE-MRNI approach with the baseline NDE approach for remote-controlled AGV use cases. In addition, to model the 5G channel delays, we considered two probability distributions, i.e., Weibull [39] and Gamma [40] distributions, due to their widespread use in modeling communication channel delays in remote-controlled systems [41, 42]. Moreover, from our experiments, the results from the two-channel delay distributions were comparable; henceforth, we only present the results from the Weibull distribution in this section. Furthermore, for each distribution, we considered five network loads with mean delay values ranging from 50 ms to 150 ms in steps of 25 ms and standard deviations equal to 20% of the mean values. This resulted in highly variable delays for each network load. It is crucial to note that the industrial AGVs considered in these experiments are sensitive to much higher network delay values, i.e.,



(a) 8-Shape trajectory



(b) Irr-hex-Shape trajectory

Figure 6: Considered AGV path trajectories, i.e., the 8-Shape and the Irr-hex-Shape, respectively

up to delays $\leq 50ms$, these AGVs perform acceptably; hence, we begin our tests considering network delay values $\geq 50\,ms$. However, some remote-controlled use cases are sensitive to much lower network delay values. So to cater for such use cases, we have also tested the potency of our proposed NDE-MRNI approach considering network delay values $\leq 50\,ms$ and these results are provided in section 6.3. Besides, we set the AGV linear velocity (v) to 1 m/s, the maximum angular velocity (ω) as 1.3 rad/s, and the controller look-ahead distance parameter to 1.2 m. These speeds are the standard in industrial applications [43].

In addition, for each of the KPI results presented henceforth, we carried out the simulations 30 times, and at each time, we varied the channel conditions. Thus, we computed the mean value over the 30 simulations, and these are the results presented in our graphs. Besides, the 95%

confidence interval of the presented mean value over the 30 values for each KPI is also indicated. Moreover, to evaluate the NDE-MRNI approach, we carried out two categories of experiments. On the one hand, we conducted experiments to find the optimal value of n to provide the optimal estimate of the current delays (i.e., $\widehat{T}_{k,UL}$ and $\widehat{T}_{k,DL}$) for the NDE-MRNI approach. On the other hand, given the most optimal value of n, we simulated the NDE-MRNI approach and compared its path-tracking improvement for the use case (considering the relevant use case KPIs) against other simulation approaches, mainly the baseline approach. The results from these two kinds of tests are presented henceforth.

6.1. Optimal n parameter for the NDE-MRNI approach analysis results

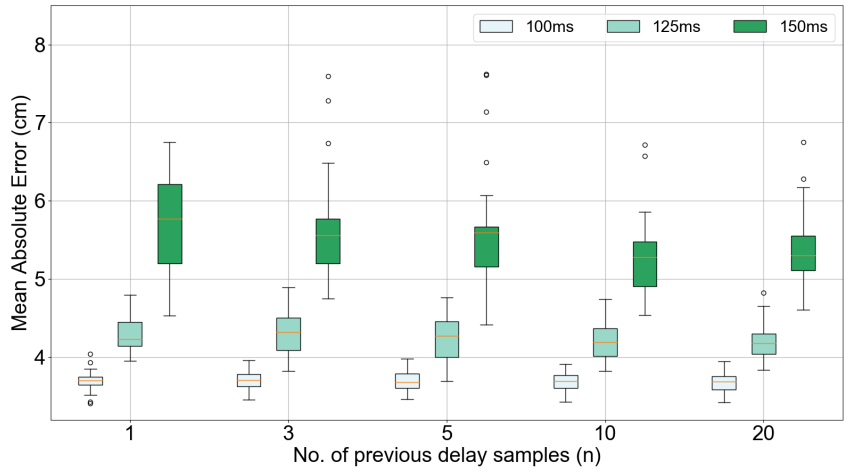
To determine the optimal value of n (i.e., the number of previous delay samples; where each previous delay sample value is the average delay in the last specified time interval (in our case, this interval was set to $50ms$)) to use in (8) and (9) for the estimation of $\widehat{T}_{k,DL}$ and $\widehat{T}_{k,UL}$, respectively, we performed the following tests:

- For each AGV path trajectory, considering the different delay values for a given channel distribution, we implemented the NDE-MRNI approach by varying the value of n.
- We considered $n = \{1, 3, 5, 10, 20\}$, where each value of n represents the number of previous delay sample values used to estimate the current delays (i.e., $\widehat{T}_{k,UL}$ and $\widehat{T}_{k,DL}$). From our tests, these are the values of n that yield a noticeable difference in performance from the previous n value.
- Thirdly, for each value of n, we ran the simulation 30 times and computed the MAE KPI for each simulation. The results from these tests for each path trajectory are presented in 7a and 7b.

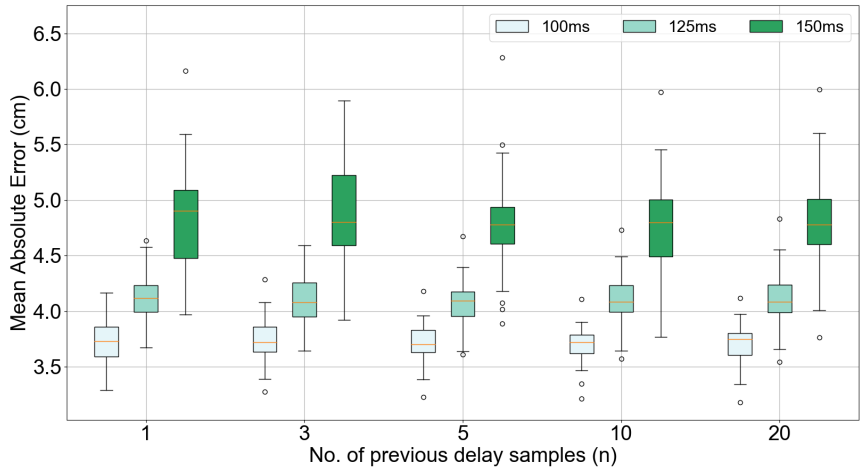
From 7, first, we notice that only results corresponding to $100ms$, $125ms$ and $150ms$ delay values have been presented; this is because for delay values $< 100ms$, for the NDE-MRNI approach, all values of n give similarly good performance. So, for comparison purposes, we only present the results for delays $\geq 100ms$. On the one hand, for the 8-Shape trajectory, we notice a slight decrease in the MAE as n increases until $n = 10$. After $n > 10$, we do not notice any further decrease in the MAE KPI. On the other hand, for the Irr-hex-Shape trajectory, we notice a similar trend, i.e., a decrease in the MAE KPI as n increases, but this time, this decrease is up to $n = 5$. In this case, for $n \geq 5$, we have very similar performance for the MAE KPI, with $n = 5$ and $n = 10$ giving slightly better results compared to $n = 20$. These results showed us that using a higher number of previous delay samples to estimate the current network delay yields better MAE KPI results; however, after $n = 10$ samples, there is no noticeable improvement in the MAE KPI. Consequently, we chose the optimal value of n for both trajectories as 10; thus, the KPI results presented for the NDE-MRNI approach are based on $n = 10$.

6.2. KPI results of the NDE-MRNI approach versus the other considered simulation approaches

The obtained KPI results for the NDE-MRNI approach compared to the different simulation approaches presented in Section 5.2 are presented hereafter:



(a) 8-Shape trajectory



(b) Irr-hex-Shape trajectory

Figure 7: Number of previous delay samples n used in the NDE-MRNI approach

6.2.1. Mean Absolute Error

The average MAE for each delay value is presented in Fig. 8, where we notice the following:

- Firstly, the MAE increases as the delay values increase, with the no NDE approach giving the worst results.
- Secondly, we observe lower MAE values with the Kalman filter, NDE-MRNI, and Ideal NDE approaches, with the latter two giving better reductions in the MAE for delay values

$\geq 100ms$.

- Thirdly, the NDE-MRNI approach provides very similar results to the case with ideal NDE.
- Fourthly, the MAE at 50 *ms* delay is close to the ideal case with no delay. Therefore, we can conclude that up to mean delays of 50 *ms*, all approaches give comparable results, and the MAE is relatively low; hence, the remote-controlled AGV can move without any problem.

However, as the mean delay values increase to 75 *ms*, the Kalman filter, NDE-MRNI approach, and Ideal NDE approaches give similar results and reduce the MAE by at least 6% compared to the case with no NDE. At 100 *ms*, the NDE-MRNI performs better than the Kalman filter approach with a net reduction in the MAE of 15% and 16% for the 8-Shape and Irr-hex-Shape trajectories, respectively.

As the mean delay increases to $125ms$, the decrease in the MAE by the NDE-MRNI approach is much higher, i.e., at least 33% & 28% for the 8-Shape and Irr-hex-Shape trajectories, respectively, compared to the Kalman filter approach. Finally, at mean delays of 150 *ms*, this improvement in the MAE by the NDE-MRNI approach is 59% and 55% for the 8-Shape and Irr-hex-Shape trajectories, respectively compared to the Kalman filter approach.

6.2.2. Time taken

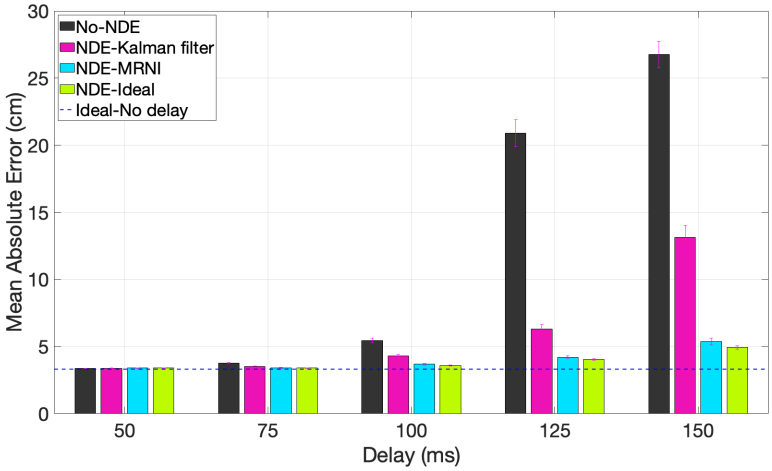
The average time taken by the AGV at each mean delay value is provided in Fig. 9. From Fig. 9, we make the following remarks about this KPI:

- The KPI increases with the mean delay value, with the no NDE approach giving the worst results.
- The mean time taken by the AGV is much less with the Kalman filter, NDE-MRNI approach, and Ideal NDE approaches, with the latter two giving slightly better results at delay values ≥ 100 *ms*.
- The NDE-MRNI approach provides relatively similar results to the Ideal NDE approach.
- Furthermore, at a mean delay of 50 *ms*, the time taken by the AGV to traverse the path trajectory is very close to the ideal case with no delay, with all approaches giving similar results.

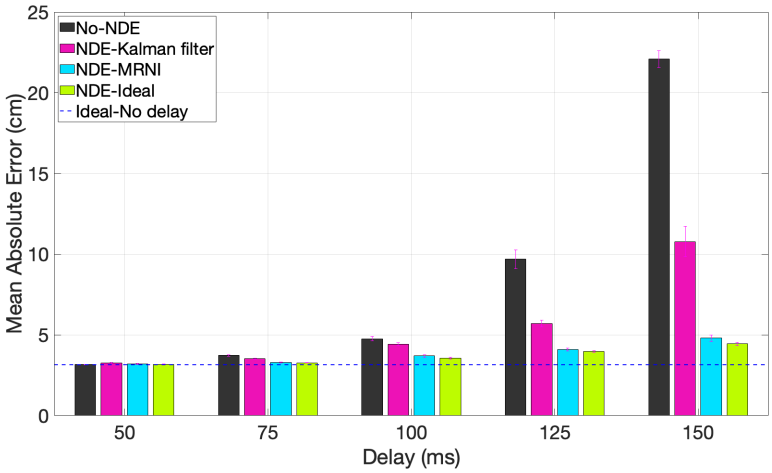
Till mean delays of 75 *ms*, all approaches give similar results. However, as the mean delay increases to 100 *ms*, the Kalman filter, NDE-MRNI, and ideal NDE approaches give a slight improvement in the time taken KPI of at least 1% relative to the no NDE approach. On the other hand, at mean delays of 125 *ms*, the NDE-MRNI approach improves the time taken KPI by at least 1.4% & 1% for the 8-Shape and Irr-hex-Shape trajectories, respectively, relative to the Kalman filter NDE approach. Lastly, at mean delays of 150 *ms*, the improvement in this KPI by the NDE-MRNI approach is more significant than the Kalman filter approach by 6% and 4% for the 8-Shape and Irr-hex-Shape trajectories, respectively.

6.2.3. Control Effort

The mean control effort KPI is shown in Fig. 10, and we can make the following observations:



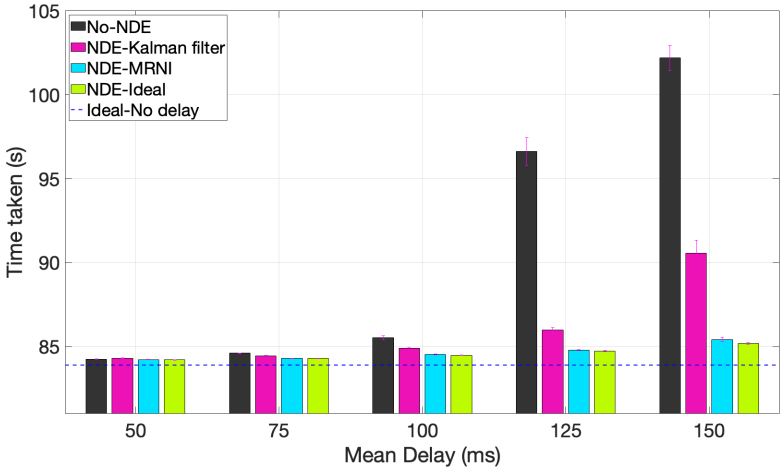
(a) 8-Shape trajectory



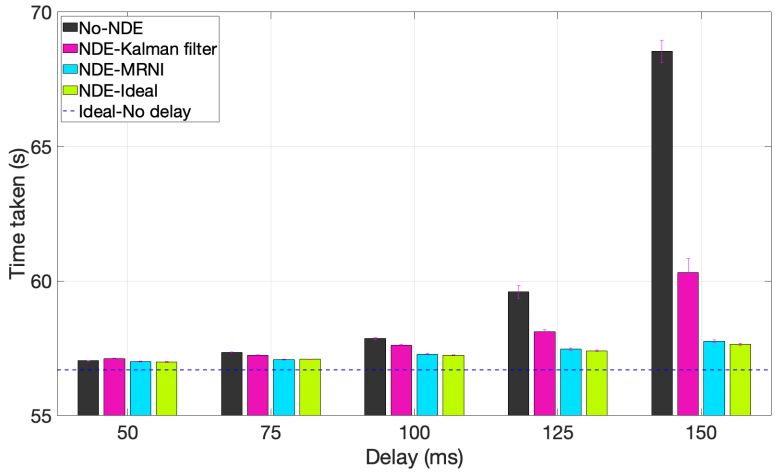
(b) Irr-hex-Shape trajectory

Figure 8: Mean Absolute Error of the AGV's movement when traversing the 8-Shape and Irr-hex-Shape trajectories

- This KPI grows linearly with the mean delay values for the no NDE and Kalman filter approaches after delays $\geq 100ms$.
- On the other hand, till mean delays of $100ms$, this KPI is almost constant with the increase in the mean delays for the NDE-MRNI and the ideal NDE approaches. In addition, these two approaches give the best and yet comparable results, outperforming the Kalman filter approach.
- The mean control effort at 50 ms is close to the ideal case, with all approaches providing almost equivalent results. However, it is essential to note some slight decline in performance for the NDE-Kalman filter approach at this delay value compared to the No-NDE



(a) 8-Shape trajectory



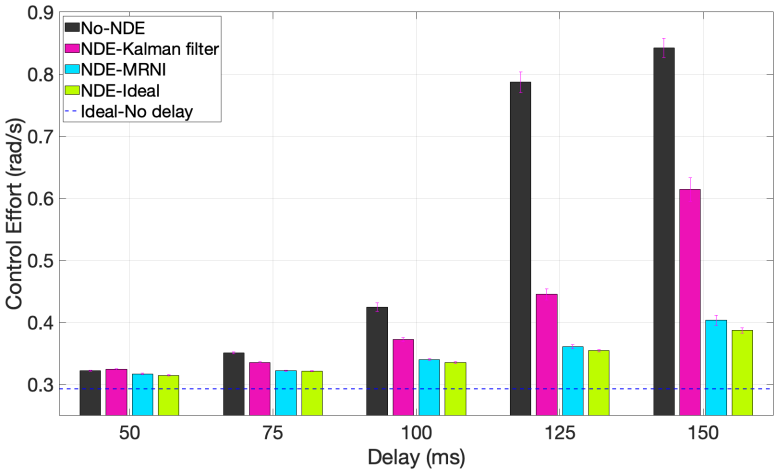
(b) Irr-hex-Shape trajectory

Figure 9: Average time taken by the AGV when traversing the 8-Shape and Irr-hex-Shape trajectories

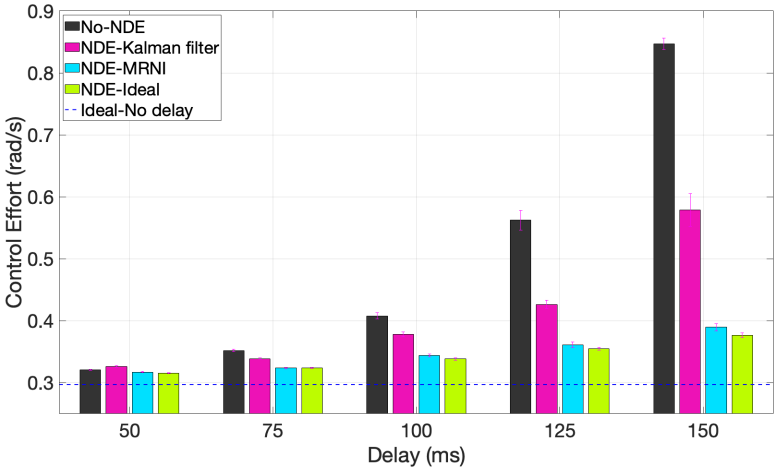
approach. This is due to the initial guesses of the Kalman filter's error covariance and RTT parameters. These initial guesses are utilized to determine the filter outputs at the initial iteration and are updated in the successive iterations hence the improved performance as the delay values increase. Nevertheless, our initial guesses are not so far off since the performance decline is just 0.7% and 1.6% for the 8-Shape and the Irr-hex-Shape trajectories compared to the No-NDE approach.

As the mean delay increases, the KPI improvement by the NDE-MRNI approach relative to the Kalman filter approach considering the 8-Shape trajectory can be summarized as: (i) 4% at 75 *ms*, (ii) 9% at 100 *ms*, (iii) 19% at 125 *ms*, and (iv) 34% at 150 *ms*. Conversely, taking into account the Irr-hex-Shape trajectory, this improvement can be quantified as follows: (i) 5% at

75 ms, (ii) 9% at 100 ms, (iii) 15% at 125 ms, and (iv) 33% at 150 ms.



(a) 8-Shape trajectory



(b) Irr-hex-Shape trajectory

Figure 10: Average Control effort considering the 8-Shape and Irr-hex-Shape trajectories

6.3. Additional KPI results for the NDE-MRNI approach considering much lower network delays

As seen from the results presented in section 6.2, the remote-controlled AGV use case considered in this paper is sensitive to much higher network delays, i.e., for delays $\leq$ 50 ms, the AGV performs acceptably. However, in this section, we wanted to test the efficacy of the proposed NDE-MRNI approach when applied to use cases that are sensitive to much lower delays, i.e., delays $\leq$ 50 ms. To emulate this scenario in our experiments, we increased the speed of the AGV from 1 m/s to 2.5 m/s; this way, the effect of a slight increase in network delays has a

higher impact on the path tracking performance of the AGV. Moreover, we employed a genetic algorithm [44] to find the new optimal controller parameters for each considered path trajectory. For the 8-Shape trajectory, the maximum angular velocity was tuned to 2.111346 rad/s, whereas the look-ahead distance was tuned to 1.355904 m. Conversely, for the Irr-hex-Shape, the maximum angular velocity was set to 2.284065 rad/s, whereas the look-ahead distance was set to 1.471513 m.

In addition, similar to section 6.2, for each probability distribution, we examined five network loads with mean delay values varying from 10 ms up to 50 ms in steps of 10 ms and standard deviations equal to 20% of the mean delay values. This resulted into highly variable delays for each network load. It is important to note that we did not evaluate average delay values < 10 ms because, in AGV applications, we would need to go faster than 2.5 m/s to test the impact of such low delay values. However, these speeds are generally not recommended for such applications due to safety regulations. Accordingly, the results from these experiments are presented in the following section.

6.3.1. Mean Absolute Error

From Fig. 11, on the one hand, we observe that even at lower delay values, the proposed NDE-MRNI approach provides some performance improvement over the No-NDE approach for delay values ≥ 20 ms. In particular, for the 8-Shape trajectory, this improvement can be quantified as:(i) 1.1% at 20 ms, (ii) 2.85% at 30 ms, (iii) 4% at 40 ms and (iv) 9% at 50 ms. Whereas, for the Irr-hex-Shape trajectory, this improvement is slightly higher and can be approximated as:(i) 4% at 20 ms, (ii) 7% at 30 ms, (iii) 10% at 40 ms and (iv) 9% at 50 ms. On the other hand, both the proposed NDE-MRNI and NDE-Kalman filter approaches have similar performance for the 8-Shape trajectory. However, for the Irr-hex-Shape trajectory, the proposed NDE-MRNI approach provides slight improvements over the NDE-Kalman filter approach for mean delay values ≥ 30 ms of (i) 2.3% at 30 ms, (ii) 3.5% at 40 ms and (iii) 4% at 50 ms.

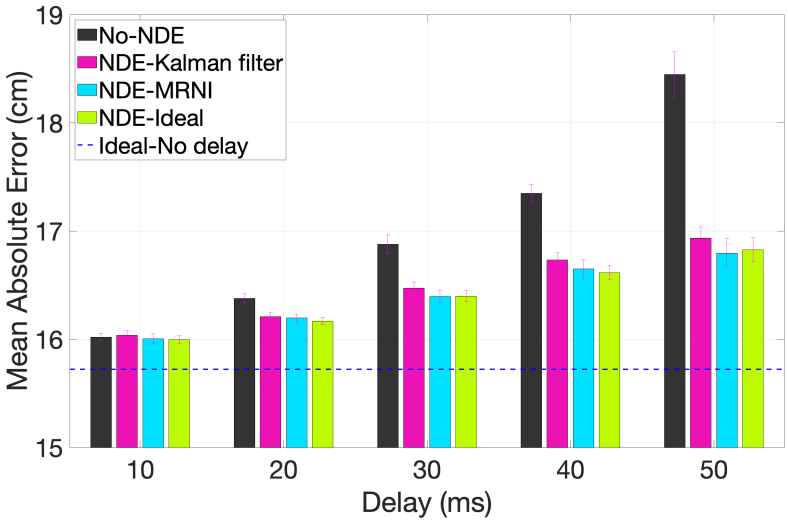
6.3.2. Time taken

The average time taken by the AGV in this scenario is provided in Fig. 12, and we can notice the following: for the 8-Shape trajectory, the improvement by the proposed NDE-MRNI approach compared to the other approaches is minuscule. Conversely, for the Irr-hex-Shape trajectory, this improvement compared to the No-NDE approach is more noticeable for mean delay values ≥ 30 ms, i.e., 1% for 30 ms, 2% for 40 ms and 3% for 50 ms. Besides, the proposed NDE-MRNI approach performs very similarly to the NDE-Kalman filter approach.

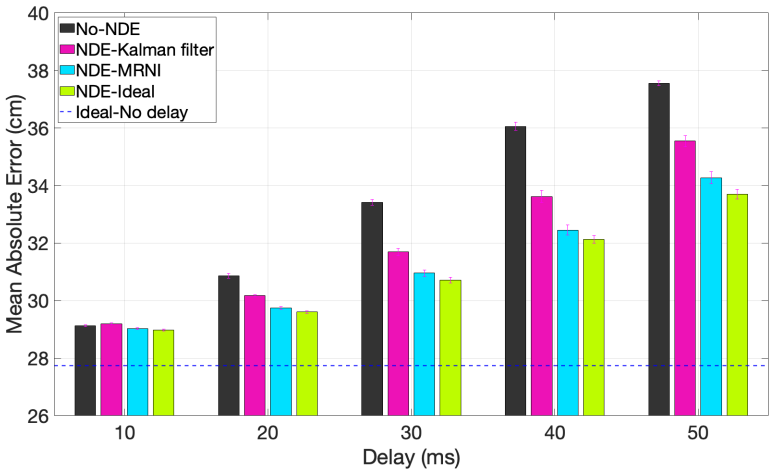
6.3.3. Control Effort

From Fig. 13, we notice that at mean delay values of 10 ms, all approaches give similar performance. Furthermore, for the 8-Shape trajectory, we can observe some minimal performance improvements over the No-NDE for delay values ≥ 30 ms of: (i) 1% for 30 ms, (ii) 2% for 40 ms and (iii) 3.4% for 50 ms. On the other hand, for the Irr-hex-Shape, this performance improvement is doubled compared to the 8-Shape trajectory. This improvement can be quantified as: (i) 2% for 20 ms, (ii) 4% for 30 ms, (iii) 5% for 40 ms and (iv) 6% for 50 ms.

Accordingly, the proposed NDE-MRNI approach can also be leveraged for use cases sensitive to lower network delays and provide some performance improvements, especially regarding the



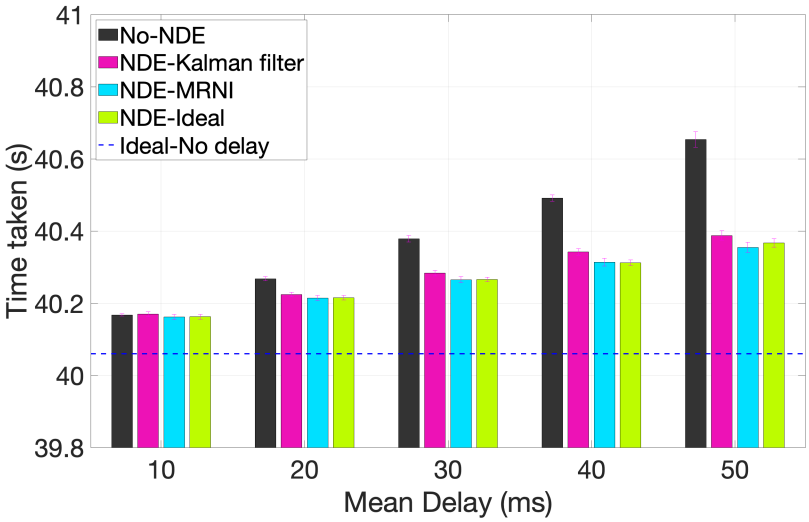
(a) 8-Shape trajectory



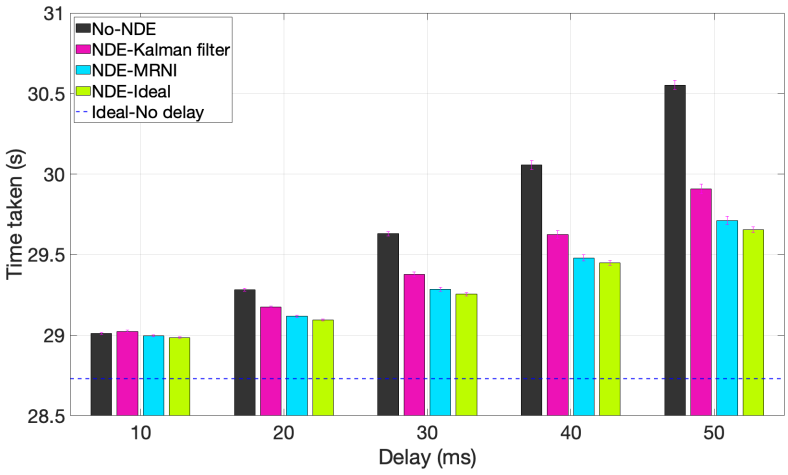
(b) Irr-hex-Shape trajectory

Figure 11: Mean Absolute Error of the AGV's movement when traversing the 8-Shape and Irr-hex-Shape trajectories

MAE and Control Effort KPIs. Moreover, even though the proposed NDE-MRNI approach provides very similar performance to the NDE-Kalman filter approach for such delay values, it is crucial to note that the proposed NDE-MRNI approach does not require any modification to the remote-controlled device. In contrast, the latter requires that the remote-controlled device periodically sends the round-trip time variable depending on the use case configuration. Besides, the accuracy of the NDE-Kalman filter approach depends heavily on the proper choice of the filter parameters, whereas this is not a concern for the proposed NDE-MRNI approach.



(a) 8-Shape trajectory

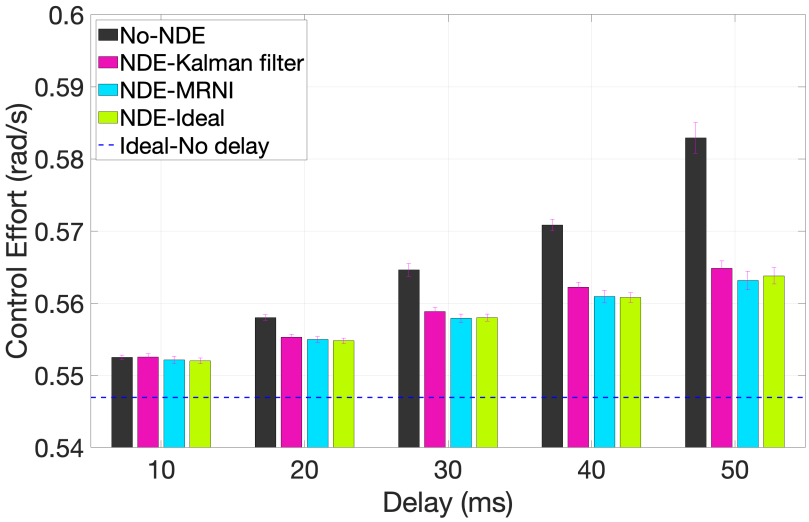


(b) Irr-hex-Shape trajectory

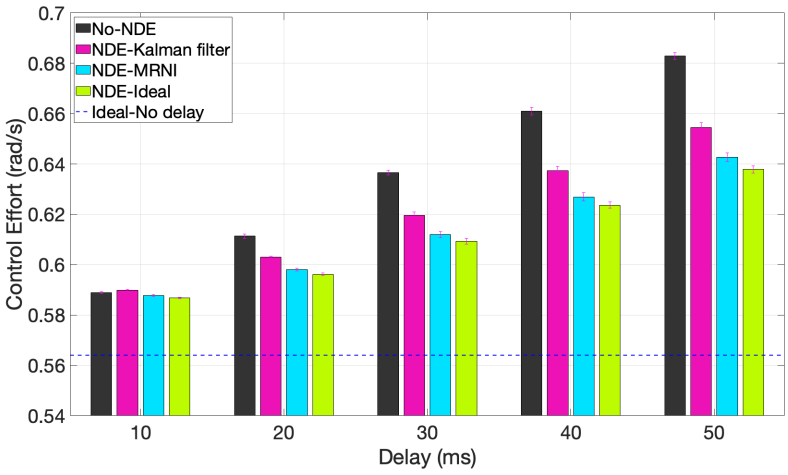
Figure 12: Average time taken by the AGV when traversing the 8-Shape and Irr-hex-Shape trajectories

7. Conclusion

This paper has augmented a remote-controlled AGV use case with the relevant MEC platform service, i.e., the MRNI service, to provide the radio network delay information (i.e., the NDE-MRNI approach). Subsequently, we have shown how this radio network delay information can be leveraged by the considered use case to improve the path tracking performance in the presence of random delays. Moreover, we have compared this performance improvement against the most optimal baseline NDE approach considering the relevant use case KPIs. Our results show that the NDE-MRNI approach outperforms the optimal baseline approach for delays $\geq$ 75 *ms*. In



(a) 8-Shape trajectory



(b) Irr-hex-Shape trajectory

Figure 13: Average Control effort considering the 8-Shape and Irr-hex-Shape trajectories

addition, our experiments indicate that the proposed NDE-MRNI approach can be leveraged by use cases sensitive to much lower delays, i.e., ≤ 50 *ms*, and provide performance enhancements compared to the case without any delay compensation. Besides, our results indicate that, as the delay values increase, the use case performance deteriorates rapidly without efficient network delay estimation techniques like the NDE-MRNI approach and subsequent delay compensation methods at the remote controller end.

In our future work, we plan to test the NDE-MRNI approach with the actual scenario composed of a physical AGV, a MEC framework providing the relevant MEC service and applica-

tions, and a private MEC-based 5G network. Furthermore, for this paper, only the pose information was extracted from the sensor data; we plan to extract more information from the sensor data. Accordingly, we plan to leverage these data to evaluate how the NDE-MRNI approach can be employed for obstacle avoidance in the remote-controlled AGV use case. Moreover, we also plan on exploiting machine Learning algorithms to estimate the network delays in the NDE-MRNI approach and compare it with the approach presented in this paper. In addition, to evaluate the applicability of the proposed NDE-MRNI approach for ultra-low latency applications, we plan to test this approach considering a robotic arms use case.

Acknowledgment

This work was partly funded by the European Commission under the European Union's Horizon 2020 program - grant agreement number 815074 (5G EVE project). The paper solely reflects the views of the authors. The Commission is not responsible for the contents of this paper or any use made thereof.

References

[1] S. K. Rao, R. Prasad, Impact of 5g technologies on industry 4.0, Wireless personal communications 100 (1) (2018) 145–159.

[2] 3GPP, Technical Specification Group Services and System Aspects; Study on Communication for Automation in Vertical Domains;(Release 16), Technical report (tr) 22.804, 3GPP (Jul. 2020).

[3] G. Hampel, C. Li, J. Li, 5g ultra-reliable low-latency communications in factory automation leveraging licensed and unlicensed bands, IEEE Communications Magazine 57 (5) (2019) 117–123. doi:10.1109/MCOM.2019.1601220.

[4] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1657–1681.

[5] I. Sittón-Candanedo, R. S. Alonso, S. Rodríguez-González, J. A. G. Coria, F. De La Prieta, Edge computing architectures in industry 4.0: A general survey and comparison, in: International Workshop on Soft Computing Models in Industrial and Environmental Applications, Springer, 2019, pp. 121–131.

[6] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, D. O. Wu, Edge computing in industrial internet of things: Architecture, advances and challenges, IEEE Communications Surveys & Tutorials 22 (4) (2020) 2462–2488.

[7] ETSI MEC, Multi-access edge computing (MEC); Framework and reference architecture, ETSI GS MEC 003 (2020) V2.2.1.

[8] ETSI MEC, Multi-access Edge Computing (MEC); Radio Network Information API, ETSI GS MEC 12 (Dec. 2019).

[9] P. Mach, Z. Becvar, Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys & Tutorials 19 (3) (2017) 1628–1656.

[10] W. Nakimuli, J. Garcia-Reinoso, J. E. Sierra-Garcia, P. Serrano, I. Q. Fernández, Deployment and evaluation of an industry 4.0 use case over 5g, IEEE Communications Magazine 59 (7) (2021) 14–20. doi:10.1109/MCOM.001.2001104.

[11] P. M. Kebria, H. Abdi, M. M. Dalvand, A. Khosravi, S. Nahavandi, Control methods for internet-based teleoperation systems: A review, IEEE Transactions on Human-Machine Systems 49 (1) (2018) 32–46.

[12] H. Lu, Y. Hu, C. Guo, W. Zhou, New stability criteria for event-triggered nonlinear networked control system with time delay, Complexity 2019 (2019).

[13] Y. Zhang, S. Xie, L. Ren, L. Zhang, A new predictive sliding mode control approach for networked control systems with time delay and packet dropout, IEEE Access 7 (2019) 134280–134292.

[14] C. Lozoya, P. Martí, M. Velasco, J. M. Fuertes, E. X. Martín, Simulation study of a remote wireless path tracking control with delay estimation for an autonomous guided vehicle, The International Journal of Advanced Manufacturing Technology 52 (5-8) (2011) 751–761.

[15] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, V. Young, Mobile edge computing—a key technology towards 5g, ETSI white paper 11 (11) (2015) 1–16.

[16] Y. Mao, C. You, J. Zhang, K. Huang, K. B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Communications Surveys & Tutorials 19 (4) (2017) 2322–2358.
[17] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, et al., Mec in 5g networks, ETSI white paper 28 (2018) 1–28.
[18] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet of Things Journal 5 (1) (2017) 450–465.
[19] K. Antevski, M. Groshev, L. Cominardi, C. Bernardos, A. Mourad, R. Gazda, Enhancing edge robotics through the use of context information, in: Proceedings of the Workshop on Experimentation and Measurements in 5G, 2018, pp. 7–12.
[20] P. M. de Sant Ana, N. Marchenko, P. Popovski, B. Soret, Wireless control of autonomous guided vehicle using reinforcement learning, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–7.
[21] 3GPP TS 38.314, 5G; NR; Layer 2 measurements, 3GPP (Release 16) (Apr. 2021).
[22] G. Oriolo, A. De Luca, M. Vendittelli, Wmr control via dynamic feedback linearization: design, implementation, and experimental validation, IEEE Transactions on Control Systems Technology 10 (6) (2002) 835–852. doi:10.1109/TCST.2002.804116.
[23] A. L. Dontchev, W. W. Hager, V. M. Veliov, Second-order runge–kutta approximations in control constrained optimal control, SIAM journal on numerical analysis 38 (1) (2000) 202–226.
[24] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, E. Frazzoli, A survey of motion planning and control techniques for self-driving urban vehicles, IEEE Transactions on intelligent vehicles 1 (1) (2016) 33–55.
[25] D. S. Lal, A. Vivek, G. Selvaraj, Lateral control of an autonomous vehicle based on pure pursuit algorithm, in: 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy), IEEE, 2017, pp. 1–8.
[26] M. Buehler, K. Iagnemma, S. Singh, The 2005 DARPA grand challenge: the great robot race, Vol. 36, Springer, 2007.
[27] M. Buehler, K. Iagnemma, S. Singh, The DARPA urban challenge: autonomous vehicles in city traffic, Vol. 56, springer, 2009.
[28] W. Ren, J. Xiong, H-infinity control of linear networked and quantized control systems with communication delays and random packet losses, IEEE Transactions on Systems, Man, and Cybernetics: Systems (2021).
[29] G. Pin, G. Fenu, V. Casagrande, D. Zorzenon, T. Parisini, Robust stabilization of a class of nonlinear systems controlled over communication networks, IEEE Transactions on Automatic Control (2020).
[30] TCPDUMP, TCPDUMP & LIBPCAP (04 2022).
URL https://www.tcpdump.org/
[31] ETSI MEC, "MEC Sandbox - Experience MEC APIs" (Accessed on: Jul. 23, 2021).
URL https://try-mec.etsi.org/
[32] MathWorks, "MATLAB - Maths.Graphics.Programming" (Accessed on: Jul. 23, 2021).
URL https://es.mathworks.com/products/matlab.html
[33] A. Abbaspour, K. Alipour, H. Zare Jafari, S. A. A. Moosavian, Optimal formation and control of cooperative wheeled mobile robots, Comptes Rendus Mécanique 343 (5) (2015) 307–321. doi:https://doi.org/10.1016/j.crme.2015.04.003.
URL https://www.sciencedirect.com/science/article/pii/S1631072115000455
[34] K. Jacobsson, H. Hjalmarsson, N. Möller, K. H. Johansson, Round trip time estimation in communication networks using adaptive kalman filtering, in: Reglermöte, Gothenburg, Sweden, 2004.
[35] MathWorks, "Design and use Kalman filters in MATLAB and Simulink" (Accessed on: Dec. 16, 2021).
URL https://www.mathworks.com/discovery/kalman-filter.html
[36] J. E. Sierra-García, M. Santos, Mechatronic modelling of industrial agvs: A complex system architecture, Complexity 2020 (2020).
[37] K. Fransen, J. van Eekelen, Efficient path planning for automated guided vehicles using a*(astar) algorithm incorporating turning costs in search heuristic, International Journal of Production Research (2021) 1–19.
[38] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino, C.-C. Peng, Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges, Sensors 18 (9) (2018) 3170.
[39] F. Jameel, M. A. A. Haider, A. A. Butt, et al., Performance analysis of vanets under rayleigh, rician, nakagami-m and weibull fading, in: 2017 International Conference on Communication, Computing and Digital Systems (C-CODE), IEEE, 2017, pp. 127–132.
[40] P. S. Chauhan, S. Kumar, V. K. Upaddhyay, S. K. Soni, Unified approach to effective capacity for generalised fading channels, Physical Communication 45 (2021) 101278.
[41] I. Ivan, P. Besnier, X. Bunion, L. Le Danvic, M. Drissi, On the simulation of weibull fading for v2x communications, in: 2011 11th International Conference on ITS Telecommunications, 2011, pp. 86–91. doi:10.1109/ITST.2011.6060167.

[42] H. Chen, P. Huang, Z. Liu, Z. Ma, Time delay prediction for space telerobot system with a modified sparse multivariate linear regression method, Acta Astronautica 166 (2020) 330–341.
[43] ASTI Mobile Robotics, "ASTI Mobile Robotics - Leading the Mobile Robotics era" (Accessed on: Dec. 23, 2021). URL https://www.astimobilerobotics.com/
[44] S. Katoch, S. S. Chauhan, V. Kumar, A review on genetic algorithm: past, present, and future, Multimedia Tools and Applications 80 (5) (2021) 8091–8126.