www.cad-journal.net
Computer-Aided Design AND APPLICATIONS™

A Real-time Dynamic Model in VR Environments: A Case Study

Franca Giannini[1], Katia Lupinetti [1], Marina Monti[1], Luca Mantelli[2], Marco Ferrando[2], Alberto Traverso[2], Sara Anastasi[3], Augugliaro Giuseppe[3] and Luigi Monica[3]

[1]IMATI, Consiglio Nazionale delle Ricerche, franca.giannini@ge.imati.cnr.it,
[2]TPG-DIME, Università degli Studi di Genova, luca.mantelli@edu.unige.it,
[3] DIT, Istituto Nazionale Assicurazione contro gli Infortuni sul Lavoro, s.anastasi@inail.it

Corresponding author: Franca Giannini, franca.giannini@ge.imati.cnr.it

Abstract. The use of virtual simulators in training has been gaining attention in recent years. From this perspective, advances and cost reductions in technology make immersive virtual reality a more engaging platform to experiment with the activities to be learned, facilitating concepts and operation learning. Anyhow, to guarantee that the virtual experience can be transferred to real-world activities, it is necessary that the training system ensures a realistic behavior in terms of reaction to the learner's actions; this can only be achieved through a faithful simulation of the phenomena and of the functioning of the equipment suitably integrated with the Virtual Reality environment. To this aim, this paper describes a methodology for the integration of a dynamic model of industrial equipment with an immersive Virtual Reality simulator for the training of operators.

Keywords: VR training system, industrial equipment simulation, Natural system interaction
DOI: https://doi.org/10.14733/cadaps.2024.1076-1088

1 INTRODUCTION

The ability to represent and emulate real operations of different processes in real-time is becoming of utmost importance in Virtual Reality (VR) applications. At first, VR environments were exploited basically to visualize 3D digital content in a more immersive manner, where the user was just a watcher with limited interactions. Now, with the ongoing technological improvement, the use of VR environments aims to improve the user experience, making the user an active subject in applications. For this reason, innovative interaction methods and novel simulation solutions are finding a large room for applications in medicine [10], education [12], industry [20], and cultural heritage [4]. Considering the industrial domain, VR applications are often used for training purposes providing a replica of the real world and mimicking real interactions [3] [8] [2].

In recent years, the interest in the development and use of Virtual Reality simulators for training purposes in industrial contexts has increased due to the possibility of experimenting with potentially dangerous situations in safe and controlled conditions [18]. To be effective, such training systems

should provide realistic and immersive environments where learners can experience activities in an engaging way; gesture naturalism makes their use easy even for novices, and the digital equipment behavior reflects the real one, thus ensuring the transfer of the learnt concepts to the real-world activities [5]. Such a realist equipment behavior can be achieved only through the use of suitable dynamic models simulating the equipment functioning according to the operator actions integrated with the virtual environment.

The capabilities offered by simulation tools make possible to also consider the impact of users' actions in the VR environment from a physical point of view; thanks to these technologies, the trainees can interact naturally within a simulated environment as they do in reality. Pinheiro et al. in [21] offer a first example of such an integration, developing a 3D mechanical maintenance training simulator for F-16 aircraft engines by using the OpenSimulator open-source platform that allows to create virtual environments supporting both the creation of 3D contents and some simple physical simulations in real-time such as collision computation and vehicles velocity. In this case, the physical emulations are limited to simply action-reaction, and a more interesting approach expects the combination of VR applications with more complete existing simulation tools whose capability of modeling the reality and predicting real physical behaviors is already well established. Indeed, tools as Matlab-Simulink or Modelica are able to model cyber-physical systems whose evolution over time is determined by mathematical equations that need to be solved in real-time to get the status of the system.

In this context, this paper proposes a framework to facilitate the integration of the most recent VR technologies with simulation tools in order to develop immersive simulators. In particular, we focus on the definition and implementation of an architecture for the communication between two models, a Matlab-Simulink model and a virtual environment model. Thanks to this connection, a change of state derived by user actions on virtual elements is given as input to the Matlab-Simulink model, which, once the data has been processed, provides information about the physical change that has occurred. These changes will be consequently displayed in the virtual environment providing the users with feedback on the effect of their actions and thus improving their virtual experience.

The rest of this paper is organized as follows. Section 2 reports some related works adopting different modalities for data exchange. Section 3 proposes a method for implementing a communication between processes and, in particular, it focuses on the requirements of the virtual environment to track user's actions and display their feedback, and then on the data streaming flow. Finally, a use case is illustrated in section 4.

2 RELATED WORKS

There exist several ways to implement communication between processes [1], typically the most used methods foreseen the use of IP sockets or shared memory.

The first method uses the network for the communication, in this way it allows to work locally as well remotely, i.e. the two applications can run on different computers and communicate across the network. Differently, the shared memory allows to create segments of memory to be accessed by multiple processes, but it limits the communication to applications that run on the same computer. Therefore, the first solution seems more flexible. In this perspective, the User Datagram Protocol (UDP) is used in several applications with different purposes (e.g., training, simulation, or virtual prototyping) for data transmission, where typically users play in the VR environment interacting with different components and the result of their actions is sent to an external system that is responsible of analyzing users' behavior and actions and foreseeing physical behavior of the environment even in complex configurations.

Yamaura et al. [26] propose a simulation-based solution by using OpenMETA to integrate Simulink, Modelica and Unity. This necessity is raised by the fact that Matlab-Simulink and Modelica are currently the state-of-the-art tools for analyzing automotive models and simulating vehicle dynamics, while Unity3D is suitable for modeling complex environmental conditions.

With the aim of predicting and correcting speed-tracking errors, Wang et al. [24] model driver behaviors feeding a neural network. In their framework, Unity3D has been used to create a traffic scenario where the user (the driver) can interact by sending multiple inputs through UDP sockets to the neural network, which computes the predicted speed error and sends the prediction back through the UDP socket.

A building energy simulation has been realized as an immersive interactive experience in VR [17], where Unity3D and the Modelica building model exchange information in real-time. The user interactions in the VR environment are sent as input to the thermal building model, which computes the simulation and sends back the results as serialized data via UDP.

An immersive simulated experience where the user can perceive the indoor climate of a room has been proposed by Nytsch-Geusen et al. [18]. In this application, the user interacts in a 3D virtual room through a VR headset, and their actions are sent via UDP to a Modelica room model that computes its internal temperature. Differently from the previously discussed applications, the authors also connect the Modelica room model to a real climate chamber that is able to reproduce in real time the temperature variations simulated by the room model and make it possible for the user to actually perceive them.

In a different scenario but still exploiting UDP communication protocol, Munoz et al. [16] integrate physiological signals measured through wearable devices (such as heart rate, electroencephalography, and electromyography signals) in mobile VR applications. In their framework, physiological sensors are connected to a smartphone, and communication with external client applications is done via UDP. UDP communications are also exploited to monitor robots' actions in industrial processes by linking a 3D virtual representation of robots and the workpieces involved in their processes with a Robot Operating System that allows hardware communications for robots or sensors. An example is proposed by Sita et al. [23] that aims at simulating the monitoring strategy with an industrial robot for heavy welding and grinding tasks. To guarantee to receive in real-time the welding robot configuration also, in this case, the communication is achieved through UDP protocol.

Another implementation using a different communication protocol is proposed by Chaos et al. [6]; in this paper, authors define a system to allow users working from their home, tele-operating a real robot exchanging information by using FTP and TCP protocols to communicate.

Finally, Andaluz et al. [1] propose a real-time communication between Unity3D and Matlab software by exploiting the shared memory between these software. As already mentioned, this kind of approach assumes that both the applications (the virtual reality application and the Matlab simulator) work on the same computer, assumption that in some cases can represent a huge limitation.

The final goal of the proposed approach is to enhance the training experience by integrating a dynamic model into the virtual world. In particular, this paper aims at providing a sort of guiding framework for the creation of such an integrated environment where user's actions performed in the virtual environment result not only into 3D model changes but also into input parameters for the dynamic model execution whose outcomes become input for the user also by transforming 3D model elements, in terms of position and visual attributes, giving the users the feeling of a truly realistic interaction.

In the following section they will be introduced together with the methodology that has been applied for the development of the immersive simulator for the training of operators and verifiers of industrial equipment.

3 METHODOLOGY

To facilitate the inclusion of simulation tools in VR applications, in this section, we illustrate the requirements and the creation process of the two single applications (the VR application and the MatLab simulator) such that a proper integration and an efficient exchange of information are possible. In particular, we focus on the creation of the two involved models, i.e. the 3D digital model and the simulated physical model, and on the data streaming by using a homogeneous structure that encodes the states of the various elements present in both the applications.

3.1 Virtualization

As previously said, for some specific applications, it is important that the layout and behavior of the 3D virtual environment strongly reflect the real-world physical counterpart. Virtualizing a real environment for VR applications is a process that includes the 3D shape reconstruction of the environment and of some relevant objects as well as the identification of the interaction modalities and of the outcomes of the user's actions.

Here the focus is on the creation of 3D digital objects included in VR applications on which users can interact. Modelers usually create digital content for VR applications directly by using authoring software for computer animation and 3D rendering, such as 3DS Max. Independent of the adopted methods for content creation, it is crucial to know which elements a user can interact with so that these components are modeled as single elements of the 3D scene. With the aim of facilitating the creation of training applications in VR environments, limiting the intervention of game designers, here we report a workflow for the creation of 3D digital elements suitable for VR applications based on models obtained by using engineering CAD (Computer-Aided Design) systems adopted in industrial contexts, such as SolidWorks or CATIA to name some. Indeed, it is expected that there exist digital copies in CAD formats of most of the standard equipment used for the training on industrial applications and, in case they are not available, it is reasonable that they can be developed easier than other specific formats.

Figure 1 shows the workflow to obtain 3D models suitable for VR applications, starting from CAD assembly models. To allow the replicability of the workflow independently from the adopted CAD software, it involves non-proprietary file formats for the file exchange. In a first phase, with the help of a CAD software, 3D assembly models representing real/realistic objects are defined. During this design phase, in addition to create CAD models with the same measures of the physical counterparts, the elements on which users can interact are modelled as single components of the assembly. This operation is necessary to implement the response of these elements once users perform some actions on them. In the first block of Figure 1 an example is reported where buttons, handles and switches are single objects in the hierarchy of the model. Finally, in this phase, materials and colors are added to the model to achieve a realistic representation.

Independently from the system used, the representation adopted in CAD software (boundary representation) is not suitable for VR applications that require tessellated models. Thus, it is necessary to process these models in order to obtain a tessellation of the boundary surfaces of the single parts present in the assembly model. This operation is possible by exporting all the parts involved in the CAD assembly model in a STL format keeping their original position in the global reference frame of the CAD system. This ensures the correct position relative to each other of all affected parts, avoiding further adjustments in the VR application. The STL format, conceived for 3D printing, is the standard supported by all engineering CAD systems but, even if it represents parts as triangles, it is not accepted by VR applications. Therefore, 3D parts in STL format must be converted to an accepted VR format, such as the OBJ format, to be included in the VR application and, if necessary, simplified in vertex number to improve rendering performance. Both of these operations can be accomplished using modeling libraries such as the CinoLib library [14], which allows the processing of polygonal meshes in C++ applications, or open-source systems such as MeshLab [9] which provides a set of tools for editing and converting meshes via a GUI or batch script.

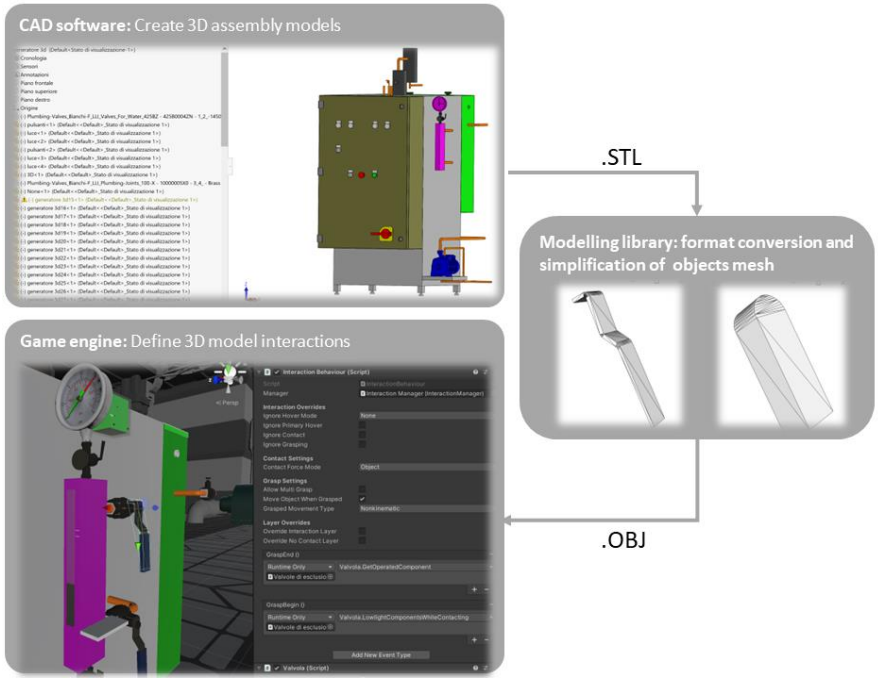


Figure 1: Workflow for the creation of 3D interactable elements in VR environments.

Finally, the created parts can be imported in a game engine, such as Unity3D, where the interaction behaviors are defined, i.e. how the user can interact with that particular component and how it reacts. To reduce repetitive behavior specification on 3D instances of same functional elements (e.g. buttons), in literature some proposals have been presented exploiting the semantics of the parts [22], [7], [24], [11]. In particular, in [11] authors introduce a methodology for the automatic association of interaction behavior to 3D shapes based on their type. Namely, the concepts of *actuators* and *control elements* have been introduced. On the one hand, actuators correspond to those components of an equipment on which the operator is acting to command the equipment. On the other hand, control elements are those components which communicate to the operator the status of indicators that are significant for the functioning of the equipment. Examples of actuators are switches and buttons, while alarm led or gauges are types of control elements. A hierarchical classification of these types of components has been defined facilitating the reuse of characteristics and behavior. This also includes the specification of the gestures for acting on them and the object changes, in terms of appearance and position, resulting from the user's action. To each element a status is associated that reflects the element actual condition. The automatic association of 3D elements to the corresponding class simply requires the specification of the correspondence class with some salient parameters.

Concerning the simulated physical model development, Matlab-Simulink has been chosen, being a software widely used in academia and industry that allows structuring complex dynamic models in a simple and intuitive way, thanks to a block display of the various functions used. The dynamic model should include all the elements with which the operator can interact, i.e. the actuators, as input variables as well as the output indicators (i.e. the control elements), present in the real system. When the dynamic model is connected to the VR environment, all the results of the actions of the users on the actuators, e.g. a specific valve is opened/closed, are sent to Matlab-Simulink, while the

control logic of the system is always reproduced inside the dynamic model. At any time, based on these inputs, the dynamic model simulates the operation of the system and sends to the VR environment all the data necessary to provide feedback to the operator. The exchange of information between the dynamic model and the VR environment is carried out by means of UDP blocks as described in the next subsection.

3.2 Data Streaming

In designing the communication between the two environments, it was taken into consideration that:

- the two applications can run on different computers, so their communication must take place via the network;
- the sharing of information between the two applications must have latency times low enough to ensure interaction in the virtual environment as realistic as possible.

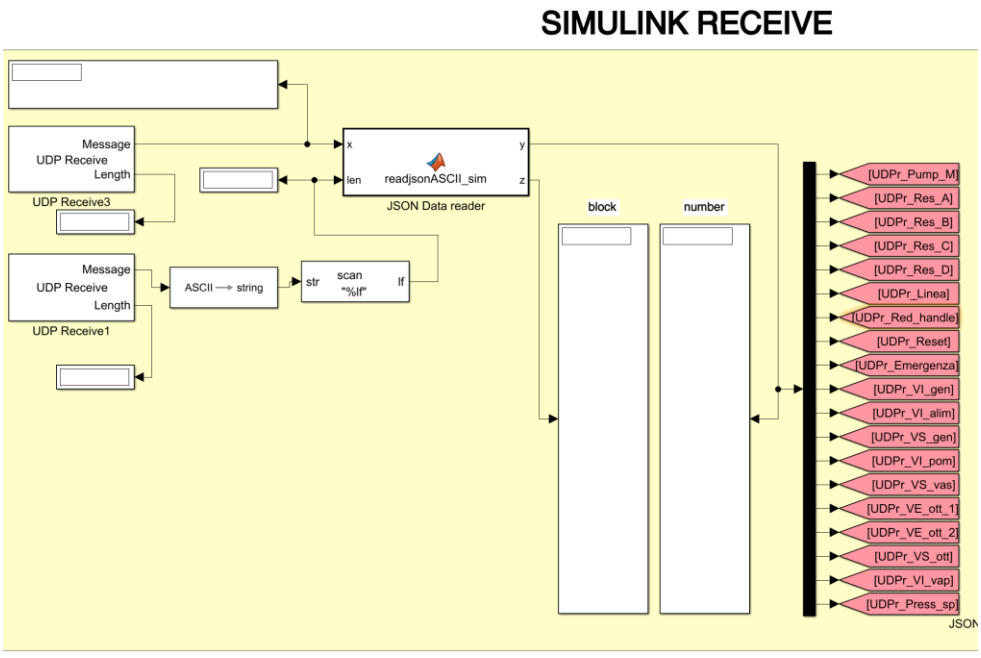
Based on these considerations, the two systems exchange data within an IP network where the User Datagram Protocol (UDP) was chosen for the transmission of data between the two applications. In fact, the UDP protocol is very fast as it does not manage the reordering of packets or the retransmission of lost ones, a feature that makes it interesting for the transmission of audio-video content in real-time or for online multiplayer games. A priori, the UDP protocol does not guarantee either message delivery or message ordering; this problem is not affecting the overall user experience considering that 3D components not updated with the latest values in a few frames are anyhow corrected the frame after. For this reason, we decided to favor the faster transmission approach (to guarantee real-time data streaming).

To allow communication between the two systems, a server needs to be configured to host the Matlab-Simulink application. In addition, the dynamic model needs to be enriched with two blocks, one to receive and the other to send the data, as depicted in Figure 2. The *receiving data block* (see Figure 2 (a)) reads two messages at two different ports, a message corresponding to the status of the operable components structured in JSON (JavaScript Object Notation) format [13] and the length of the message representing the number of bytes of the sent string. This last operation was necessary due to the rigidity of Simulink in the management of variable length inputs, while the JSON structure has been chosen to guarantee a certain flexibility in case of variation or addition of information.

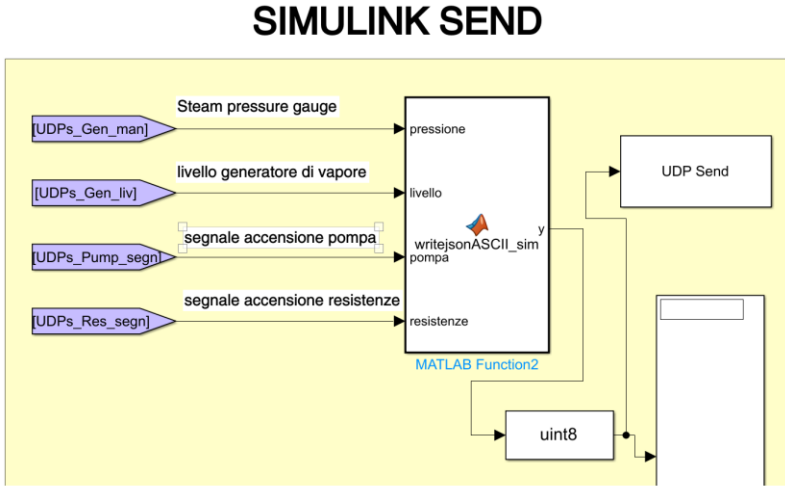
In the transmission of the data, it is important to notice that the input data of the message read by the Simulink UDP block is encoded in ASCII format; thus, to process the content of the message, it is necessary to convert the ASCII input in characters data type. Once the message is converted into text format, it presents a set of nodes representing the operable components, where each of them can be characterized by several attributes (an example of a case study is illustrated in the next section).

These nodes are then used as input of the dynamic model to compute physical simulations of the system. In the *sending data block* (see Figure 2(b)), the set of outputs is encoded in JSON with the Matlab JSONencode function, converted to string format and sent by a Simulink UDP send block.

From the VR application side, each time the operator acts on an interactable component, the datagram corresponding to the JSON (more precisely a text string in ASCII format) is sent to the network node running the Matlab-Simulink application. The reverse process is asynchronously performed: the VR application collects the received datagrams in a queue on the main thread, then, at each frame, the VR environment is updated using the last received datagram. In this way, the receiving module does not need to wait for the complete delivery of the data, avoiding the interruption of the VR application, thus ensuring a real-time updating of the VR environment with the dynamic model results.



(a) Receiving data block



(b) Sending data block

Figure 2: Architecture of the Matlab-Simulink model for receiving (a) and sending data (b).

4 CASE STUDY

The above-described communication system between a 3D VR environment and a dynamic model [15] has been applied in the proposed immersive system for the training of conductors and verifiers of industrial equipment. As use cases, steam generators are considered. In particular, the electrical steam generator available at the Savona campus of the University of Genova has been reconstructed in terms of 3D layout and dynamic model to realistically reproduce its behavior.

Figure 3 illustrates two views of the real steam generator at the Savona campus used as case study and its digital representation embedded in a virtual environment achieved through the workflow discussed in the previous section. In particular, Figure 3(a) and 3(b) depict the control panel of the generator, while Figure 3(c) and 3(d) represent a pressure gauge (blue square) and a water measure level (green rectangle) with additional components (valves in red and yellow squares) installed on the generator. Through the visible control panel, the user can (i) change the generator pressure set-point; (ii) set the pump operating mode (manual, off, automatic); (iii) activate or deactivate the four groups of electric heaters individually; (iv) connect or disconnect the system power supply; and (v) press the reset and emergency buttons. The pressure gauge and the water measure level represent two components where the operator can read measures of the system to understand its behavior. In addition, the operator can interact with the various valves for the water management. All these components are examples of *interactable elements* whose status feeds the dynamic model. The integrated dynamic model is based on a mixed physics-based data-driven approach, which supports the resolution of physical equations (e.g., mass and energy balances), the use of semi-empirical correlations and performance maps.

Moreover, the physical properties of the water are calculated by integrating Matlab-Simulink with the instrument opensource CoolProp. In this way, the model updates the properties of the internal water and of the outgoing steam by considering the heat supplied by the resistances, the pressure, the temperature, and the latent heat of the water, as well as the internal level of the liquid phase. These kinds of computations are hardly encodable in VR applications by simply using a game engine. Therefore, the integration with dedicated dynamic simulation tools allows us to achieve a more realistic training experience.

(a) Real view (b) Digital view (c) Real view (d) Digital view

Figure 3: Real and digital reconstructed views of the control panel - (a) and (b) - of the steam generator used as case study and of the pressure gauge - (c) and (d) – component.

Figure 4 shows a portion of the more complex dynamic model developed for this case study. The represented elements simulate the steam generator, the optical level meter with its exclusion and discharge valves, the liquid and vapor discharge valve, and the lamination valve. The lamination valve is the last component of the plant, and its opening influences the amount of steam delivered through the pipes. The optical level meter computes the measurement of internal liquid level that is communicated to the VR application as feedback to the learner, while the statuses of the valves are inputs of the dynamic model which depend on the learner's actions in the virtual environment.

The development of the VR environment is based on the game engine Unity 3D, and the user can access the virtual world by wearing a HTC Vive viewer and interact in the VR environment by using their bare hands that are tracked by a Leap Motion Controller. These actions are used as inputs of the dynamic model to simulate the response of the system, after which the results of the model are sent to the VR environment. These results include information on the operation of the generator,

such as pressure and water level, which the operator can read on the appropriate measuring instruments installed on the system.

Considering this application scenario, the learners can act on the system actuators (buttons, switches, etc.) with bare hand gestures within the VR environment. Once they act on any interactable component, the state of the concerned actuator is modified accordingly, and the value is communicated to the Matlab-Simulink dynamic model changing the related input values. The dynamic model is then updated, and the result of the simulation is returned to the VR application that updates the status and consequently their visual appearance of the control elements. In this way, it is possible to faithfully simulate the response of the physical system to the various actions performed by the operators, who can experience the consequences of their maneuvers as if they were acting on the real equipment. This is fundamental in understanding their mistakes in accomplishing the assigned task in a safe environment. Thus, dynamic simulation allows to overcome the limits of traditional training methodologies that are hardly sufficient to instruct the operators for seldom-occurring perilous situations [18].

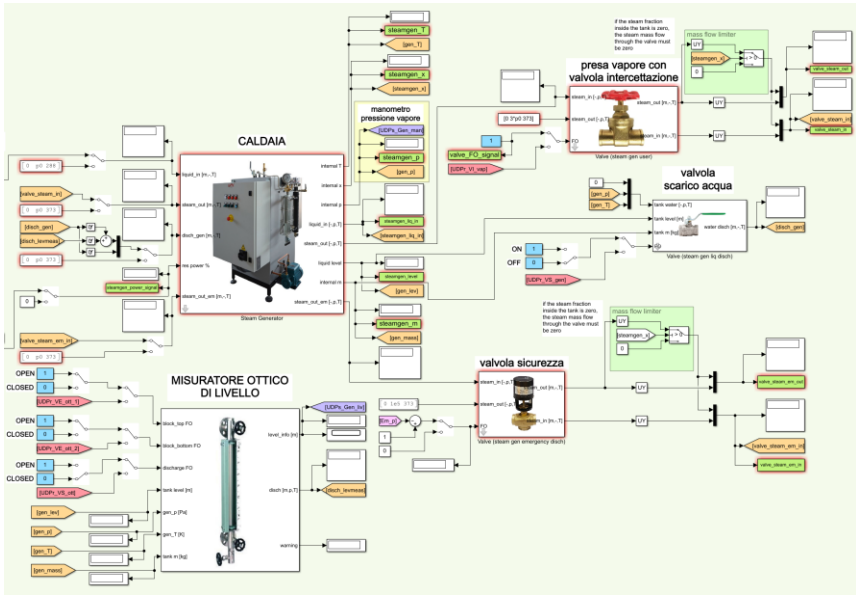


Figure 4: Portion of the dynamic model of the steam generator developed in Matlab-Simulink.

In this case study, the input/output components are characterized by three fields:

- **Name** identifies a component of the generator.
- **Status** encodes the status of a component in numerical format, which can be a discrete value 0-1 (e.g. for a group of resistors representing the off-on status) or continuous (e.g. the pressure setpoint set by the user).
- **Block** represents additional information in numerical format. For example, if a group of resistors is activated by the user but is blocked by the control logics, the value will be 0, otherwise it will be 1 (default value).

Figure 5 shows a portion of the text message in JSON format used for the information encoding that will be converted in ASCII format for the data transmission.

```
[
 {
  "status": 70,
  "block": -1,
  "name": "Optical level meter"
 },
 {
  "status": 903093.59253524442,
  "block": -1,
  "name": "Steam pressure gauge"
 },
 {
  "status": 33,
  "block": -1,
  "name": "Tank water level"
 },
 {
  "status": 0,
  "block": 1,
  "name": "Pump ignition mod"
 },
...
]
```


Figure 5: Portion of the datagram used for the data transmission in JSON format.

To confirm the correctness of the applied methodology for the integration of the dynamic model in the virtual environment, the VR application was finally validated, comparing its results with the ones provided by the Matlab-Simulink application performing different actions on the system. For this purpose, the developed dynamic model can be used also in standalone modality where it is possible to define the status of the input components directly within Simulink. In particular, the tests considered the steam generator activation and the block of the system. In the first case, it is necessary to close the drain valves (the water drain valve, the water supply tank drain valve and the optical level drain valve), open the valves to pump water inside the generator (the optical level valves, the pump suction shut-off valve, the water supply shut-off valve and the water interception valve), turn the generator on by rotating the power line lever, select a certain pressure set-point (in a range between 5 and 10 bar) and set the pump operation mode to automatic. Finally, it is necessary to turn on some of the resistances from the control panel. In this way, the resistances heat the water and eventually start the steam production. The number of activated resistances influences the time necessary to heat up the water and generate steam. During all these user's activities, the behavior of the virtual generator has been compared with the data simulated by the dynamic model to verify the correctness of the data streaming. In the second tested case, after the generator is activated and a stable working activity is achieved, the drain valves are opened to empty the generator of water. When the water level is too low the resistances are blocked by the emergency control logics of the generator. Also, this behavior is simulated by the dynamic model, which sets the "block" attribute associated with the resistors to 0 in the returned JSON data. When this condition happens, the lights of the resistances in the VR application are turned off to indicate their shutdown. If the water level is brought back to an acceptable value, the system can be restored pressing the "reset" button present in the control panel. During the validation process, the pressure trend was in line with the expected one simulated by the dynamic model. In both the tested cases, the data conveyed in the VR environment are updated with the ones in the dynamic model immediately providing the user no sense of delay in the interaction. This aspect has been evaluated

qualitatively by engineers who regularly use the real generator, and they have found that the virtual generator faithfully replicates the behavior of the real one. This ensures that, in this context, the instability in the UDP protocol with respect to other protocols like the TCP, for example, does not require mitigation actions at this level.

Finally, Figure 6 shows an example of the pressure gauge updating after receiving the dynamic model data. In this example, it is possible to read part of the JSON received in the Unity editor console (bottom left red square), where the name and status of a control element appear. Subsequently, the status of the component associated with that command element is updated, as can be observed from the inspection panel on the right, and the pointer of the pressure gauge in the VR environment displays the information received.

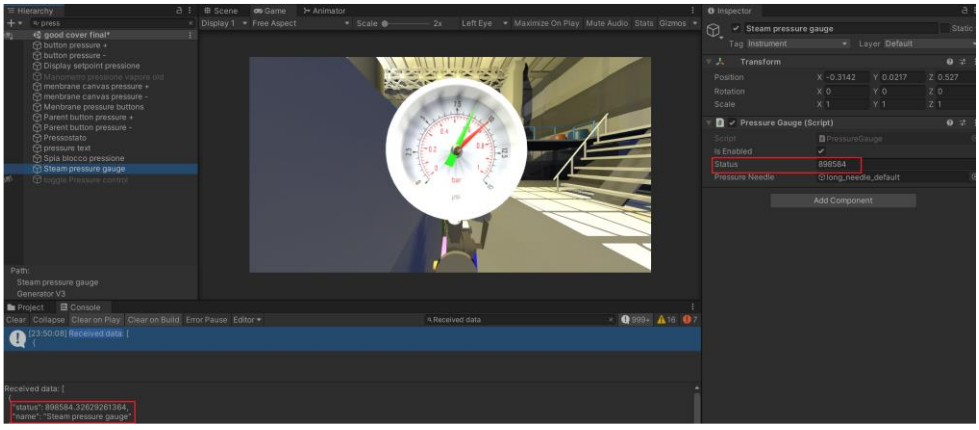


Figure 6: Example of a control element updating, where the measure shown by the pressure gauge component is determined by the dynamic model. Once the system receives the data structured in JSON format, the relative components are updated accordingly.

5 CONCLUSIONS

This paper described a methodology applied to integrate a dynamic model simulating the functional behavior of industrial equipment created in a Virtual Reality scene, where the user's actions are used as inputs of the simulation. The article described the main issues and solutions adopted for the development of a training system for conductors and verifiers of steam generators. Integrating VR simulators with dynamic models is of utmost importance for creating systems that provide a sense of realism that can overcome traditional training methodologies. It can allow the learners to experience operations by getting equipment behavior closer to that of the real world but in a safe environment. Clearly, this strongly depends not only on the ability of both the dynamic model and the VR effects to simulate real physical behavior but also on the ability to make the two environments communicate in real-time.

The qualitative evaluation of this case study shows the possibility of transmitting data via UDP in real-time between a virtual environment and a simulation tool developed in Simulink, while the flexibility of the transmission structure provided by the JSON format guarantees that it can be applied to other similar simulators. To this aim, the application and testing of this methodology to a more complex steam generator are in progress. The new case study is the auxiliary steam generator for the Savona thermoelectric plant: its dynamic model and virtualization of the plant are currently under development.

In addition, from our evaluation, this implementation does not require to manage data fragmentation since the exchanged messages' size (less than 4500 bytes in the worst case) is quite far from the UDP limit.

Finally, the usage of different networks can be further investigated. Indeed, the conducted tests access the net by cable. It would be interesting to test the system with wireless connections such as a 5G or WI-FI networks.

ACKNOWLEGEMENTS

The presented work has been partially supported by the project "PITSTOP: Piattaforma Immersiva per il Training STrutturato dell'OPeratore", funded by INAIL (Italian Workers' Compensation Authority), Bando BRIC 2019.

Franca Giannini, http://orcid.org/0000-0002-3608-6737
Katia Lupinetti, http://orcid.org/0000-0002-0202-4909
Marina Monti, http://orcid.org/0000-0002-1627-3551
Yuanju Zhu, http://orcid.org/0000-0002-3913-5557
Luca Mantelli, http://orcid.org/ 0000-0002-4866-0302
Marco Ferrando, https://orcid.org/0000-0003-2082-6208
Alberto Traverso, https://orcid.org/0000-0001-5934-3452
Sara Anastasi, http://orcid.org/0000-0003-2163-6684
Giuseppe Augugliaro, http://orcid.org/0000-0001-9948-8058
Luigi Monica, http://orcid.org/0000-0001-5702-6912

REFERENCES

[1] Andaluz, V. H., Chicaiza, F. A., Gallardo, C., Quevedo, W. X., Varela, J., Sánchez, J. S., Arteaga, O.: Unity3D-matlab simulator in real time for robotics applications, in International Conference on Augmented Reality, Virtual Reality and Computer Graphics, Springer, 2016, 246–263.

[2] Andaluz, V. H., Pazmiño, A. M., Pérez, J. A., Carvajal, C. P., Lozada, F., Lascano, J., Carvajal, J.: Training of tannery processes through virtual reality, in International Conference on Augmented Reality, Virtual Reality and Computer Graphics, Springer, 2017, 75–93.

[3] Andaluz, V. H.; Amaquina, J. L.; Quevedo, W.X.; Aguilar, J.M.; Castillo-Carion, D.; Miranda, R.J.; Perez, M.G.: Oil processes vr training, in International Symposium on Visual Computing, Springer, 2018, 712–724. http://doi.org/10.1007/978-3-030-03801-4_62

[4] Bekele, R.; Pierdicca, E.; Frontoni, E.; Malinverni, S.; Gain, J.: A survey of augmented, virtual, and mixed reality for cultural heritage, Journal on Computing and Cultural Heritage (JOCCH), 11(2), 2018,1–36.

[5] Bowman, D. A.; McMahan, R. P.; Ragan, E. D.: Questioning naturalism in 3D user interfaces, Communications of the ACM, 55(9) 2012,78–88. https://doi.org/10.1145/2330667.2330687.

[6] Chaos, D.; Chacon, J.; J. Lopez-Orozco, A.; Dormido, S.: Virtual and remote robotic laboratory using ejs, matlab and labview, Sensors, 13(2), 2013, 2595–2612.

[7] Chevaillier, P.; Trinh, T.-H.; Barange, M.; De Loor, P.; Devillers, F.; Soler, J.; Querrec, R.: Semantic modeling of virtual environments using mascaret, in 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS), 2012, 1–8. https://doi.org/10.1109/SEARIS.2012.6231174

[8] Chiluisa, M. G.; Mullo, R. D.; Andaluz, and V. H.: Training in virtual environments for hybrid power plant, in International Symposium on Visual Computing, Springer, 2018, 193–204.

[9] Cignoni, P.; Callieri, M.; Corsini, M.; Dellepiane, M.; Ganovelli, F; Ranzuglia, G.: MeshLab: an Open-Source Mesh Processing Tool, in Eurographics Italian Chapter Conference, V. Scarano, R. D. Chiara, and U. Erra, Eds., The Eurographics Association, 2008, https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[10] Fairclough, S. H.; Gilleade, K.: Advances in physiological computing. Springer, 2014.
[11] Giannini, F.; Lupinetti, K.; Monti, M.; Zhu, Y.; Mantelli, L.; Anastasi, S.; Augugliaro, G.; Monica L.: A customizable VR system for industrial equipment operator training, Computer-Aided Design & Applications, 20(4), 2023, 716-730, http://doi.org/10.14733/cadaps.2023.716-730.
[12] Hamilton, D.; McKechnie, J.; Edgerton, E.; Wilson, C.: Immersive virtual reality as a pedagogical tool in education: a systematic literature review of quantitative learning outcomes and experimental design. J. Comput. Educ. 8, 2021, 1–32, https://doi.org/10.1007/s40692-020-00169-2.
[13] "Introducing json." [Online]. Available: https://www.json.org/json-en.html (visited on 24/03/2023).
[14] Livesu, M.: Cinolib: A generic programming header only c++ library for processing polygonal and polyhedral meshes, Transactions on Computational Science XXXIV, Lecture Notes in Computer Science, Springer, Ed., 2019, https://doi.org/10.1007/978-3-662-59958-7_4
[15] Mantelli, L.; Ferrando, M.; Traverso, A.; Giannini, F.; Lupinetti, K.; Monti, M.; Anastasi, S.; Augugliaro, G.; Monica, L.: Integration of dynamic models and virtual reality for the training of steam generator operators, J. Energy Resour. Technol, 2023, 145(6): 061701 (11 pages), https://doi.org/10.1115/1.4056561
[16] Munoz, J. E.; Paulino, T.; Vasanth, H.; Baras, K.: Physiovr: A novel mobile virtual reality framework for physiological computing, in 2016 IEEE 18th international conference on e-Health Networking, Applications and Services (Healthcom), IEEE, 2016, 1–6.
[17] Nytsch-Geusen, C.; Ayubi, T.; Mockel, J.; Radler, J.; Thorade, M: Building systems VR – a new approach for immersive and interactive building energy simulation, in BS2017: 15th Conference of Internation Building Performance Simulation Association, 2017, 628– 634.
[18] Nytsch-Geusen, C.; Mathur, K.; Westermann, L: Development of a real-time test bed for indoor climate simulation in a vr environment using a digital twin, in Modelica Conferences, 2021, 263–269.
[19] Patle, D.S.; Manca, D.; Nazir, S.; Sharma, S: Operator training simulators in virtual reality environment for process operators: A review, Virtual Reality, Augmented Reality and Commerce, 23 (3), 2019, https://doi.org/10.1007/s10055-018-0354-3
[20] Phuyal, S.; Bista, D.; Bista, R.: Challenges, opportunities and future directions of smart manufacturing: A state of art review, Sustainable Futures, 2, 2020, 100 023. https://doi.org/10.1016/j.sftr.2020.100023
[21] Morgado L., Paredes H., Fonseca B., Pedrosa D., Pinheiro A., Fernandes P., Maia A., Cruz G., Martins P.: Development of a mechanical maintenance training simulator in opensimulator for f-16 aircraft engines, Procedia Computer Science, 15, 2012, 248–255.
[22] Repetto A.; Catalano, C.E.: A semantic layer for knowledge-based game design in edutainment applications, EAI Endorsed Transactions on Future Intelligent Educational Environments, 1 (5), 2015.
[23] Sita, E.; Horvath, C. M.; Thomessen, T.; Korondi, P.; Pipe, A. G.: Ros-unity3d based system for monitoring of an industrial robotic process, in 2017 IEEE/SICE International Symposium on System Integration (SII), IEEE, 2017, 1047–1052.
[24] Walczak K., Flotynski J., Strugala D., Strykowski S., Sobocinski P., Galazkiewicz A., Gorski F., Bun P., Zawadzki P., Wielgus M., Wojciechowski R.: Semantic modeling of virtual reality training scenarios, Virtual Reality and Augmented Reality, P. Bourdot, V. Interrante, R. Kopper, A.-H. Olivier, H. Saito, and G. Zachmann, Eds., Cham: Springer International Publishing, 2020, 128–148.
[25] Wang Z., Liao X., Wang C., Oswald D., Wu G., Boriboonsomsin K., Barth M.J., Han K., Kim B., Tiwari P.: Driver behavior modeling using game engine and real vehicle: A learning-based approach, IEEE Transactions on Intelligent Vehicles, 5(4), 2020,738–749.
[26] Yamaura M., Arechiga N., Shiraishi S., Eisele S., Hite J., Neema S., Scott J., Bapty T.: Adas virtual prototyping using modelica and unity co-simulation via openmeta, in The First Japanese Modelica Conferences, 2016, 43–49.