Heliyon 10 (2024) e34456

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Beyond simulation: Unlocking the frontiers of humanoid robot capability and intelligence with Pepper's open-source digital twin

Hiba Sekkat[a,*], Oumaima Moutik[a], Badr El Kari[a], Yassine Chaibi[b], Taha Ait Tchakoucht[a], Ahmed El Hilali Alaoui[a]

[a] *Euromed University of Fes, UEMF, Morocco*
[b] *National School of Applied Sciences, University Sidi Mohamed Ben Abdellah, Fez, Morocco*

A R T I C L E I N F O

Keywords:
Pepper Robot
Digital twin
ROS 2
Simulation fidelity
Machine learning
Open-source software

A B S T R A C T

This research paper presents a high-fidelity, open-source digital-twin of the Pepper robot developed within the framework of the Robot Operating System 2 (ROS 2) for better simulation realism in complex tasks of machine learning. We developed a dedicated, custom ROS 2 package with modern simulation tools, such as Gazebo Sim, MoveIt 2, Rviz2, that brings complete, realistic environments in line with the exact behaviors and interactions of robots in reality. Better accuracy of the physical movement of Pepper robot's simulation was shown on the digital twin, validated by the Choregraphe software and real robot performance, to be a strong platform of collaboration and further research by the community. This development greatly pushes the envelope of human-like humanoid robotics further by offering a scaled, flexible, and plausible training environment conducive to integrating complex algorithms of robot learning and interaction capabilities.

1. Introduction

In recent years, the use of smart robots operating in dynamic, complex, and unstructured situations has increased dramatically [1]. As a result, researchers have developed instructional robotics simulators that feature 3D models of real robots [2]. Additionally, significant research has been conducted on the use of randomized simulations to train robots, with many studies offering a thorough review of the sim-to-real strategy in robotics [3]. Humanoid robots like Pepper have the potential to transform the way we live, work, and interact with technology [4]. These robots are designed to replicate human form and movement, enabling them to navigate and operate in human environments. However, developing and integrating complex machine learning (ML) algorithms for these robots presents significant challenges, primarily due to the limitations of current simulation environments [5]. The evaluation of these interactions in real-world scenarios is crucial, as seen in the deployment and assessment of the Pepper robot within the AMIRO social robotics framework [6].

Pepper has been used in various applications, such as customer service [7], healthcare [8], education [9], and entertainment [10]. Staying current with the latest developments in the field and incorporating advanced Artificial Intelligence (AI) techniques for training on the Pepper robot can be a significant contribution to robotics research, especially given that the Pepper robot has been used in a variety of research projects [11,12]. The cost of the physical Pepper robot is well known to be prohibitive [13], and even if it is

* Corresponding author.
E-mail address: h.sekkat@ueuromed.org (H. Sekkat).

https://doi.org/10.1016/j.heliyon.2024.e34456
Received 29 February 2024; Received in revised form 4 July 2024; Accepted 9 July 2024
Available online 10 July 2024

affordable, it is critical to test and validate the training's safety and effectiveness before deploying it on the real robot. Therefore, testing new AI approaches on a simulated Pepper robot is recommended [14]. Nevertheless, current simulation platforms for the robot Pepper present significant gaps, including a lack of high fidelity, limited integration of complex ML frameworks, and insufficient open-source scalability, which hinder large-scale research and community involvement [14].

This paper addresses the critical gaps in existing simulation platforms by developing an open-source, high-fidelity simulation environment for the Pepper robot. Rather than describing and integrating particular ML techniques, the goal here is to build a solid digital twin of Pepper that is built with the characteristics and adaptability needed to facilitate the integration of complicated ML algorithms in the future. Humanoid robots such as Pepper are hard and expensive to build and test [13] but simulation environments are critical for testing fresh concepts and increasing robot performance in a safe and effective manner [3,15]. In addition to these simulation capabilities, implementing advanced navigation systems, such as those based on SLAM technology specifically developed for social robots like Pepper, is critical for realistic interactions within these environments [16]. Advances in computer hardware and graphics have substantially improved the realism and accuracy of these simulations [17], which are critical for training and evaluating advanced machine learning algorithms [18]. Such surroundings are especially important for Pepper, because its physical equivalent cannot withstand intense training sessions. The mentioned simulation environment [18], while not open source, allows for the effective transfer of ML procedures between the digital twin and the physical robot, increasing speed and minimizing the impact on the actual robot.

In light of these improvements, this work addresses the task of creating a high-fidelity, open-source simulation environment for the Pepper robot based on the Robot Operating System 2 (ROS 2). This simulation platform is primarily intended to assist future integration of complex machine learning algorithms, rather than designing and integrating these algorithms within the scope of this research. The existing limits of simulations leveraging ROS 1 middleware [19,20] together with the growing demand for greater ML capabilities, drove the development of a ROS 2 package. This package, which combines Gazebo Sim,[1,2] and MoveIt 2, intends to provide precise simulation and assist sophisticated ML training, bridging the gap between the constraints of current simulations using ROS 1 and the growing demand for ML capabilities in ROS 2.

The main contribution of this work is the creation of an exclusive open-source ROS 2 module that considerably improves the realism and usefulness of the simulation environment. This package makes use of cutting-edge technologies including ROS 2, MoveIt 2, and the integration of Gazebo Sim as a simulator with Rviz2 visualization capabilities, resulting in a highly precise representation of the robot's behavior. By developing a powerful and adaptable digital twin of the Pepper robot, we aim to provide a foundational platform that can support advanced ML research and development.

This study evaluates the accuracy of the digital robot twin generated in the simulated environment by comparing it with both the Choreograph environment and a real Pepper robot. This comparison is crucial for validating the precision and dependability of the simulation models, underscoring our simulation's capability to closely mimic real-world dynamics and behaviors. The simulation environment's purpose is to facilitate the implementation of advanced machine-learning algorithms for training and testing the robot. Although the ML algorithms are not the focus of this paper, the study explores the potential of simulation environments in improving humanoid robots. Specifically, it highlights the environment's ability to accurately replicate the robot's behavior in real-world scenarios and validate the precision and dependability of the simulation models.

The rest of the paper is organized as follows: Section 2 delves into the Literature Review, providing insights into existing knowledge to establish a contextual backdrop. The focal point of the paper unfolds in Section 3, meticulously detailing the Technical Approach and Open-Source Implementation of the Pepper Robot Simulation leveraging ROS 2. Section 4 rigorously examines the simulation's fidelity through meticulous validation processes. Finally, the paper concludes in Section 5, where key findings are summarized, implications and limitations are discussed, and avenues for future exploration are presented.

2. Literature review

Social robots have gained widespread attention in recent years for their potential to provide engaging and entertaining experiences to the general public, as well as assist with healthcare and other tasks [21–23]. The MultiModal Mall Entertainment Robot (MM-MER) project, funded by the EU and led by Foster et al., aimed to create a humanoid robot that could interact naturally with people in a shopping mall using SoftBank Robotics' Pepper robot as the primary platform [24]. Niemelä et al. evaluated the MM-MER project through an interview study with mall stakeholders and found that the robot was generally well-received [25]. Gardecki et al. highlighted the challenges of ensuring safe and effective interaction with people in a public space while operating Pepper [26]. By developing the simulation environment presented in this paper, it is possible to improve the capabilities of social robots like Pepper that possess humanoid features.

Simulation environments have always been important for robotics research, allowing for the modeling and testing of concepts and algorithms [27]. Over the years, there has been a gradual improvement in the performance of robot perception, cognition, and decision-making algorithms, enabling robotic systems to understand and interact with their environments autonomously [17]. Modern simulation environments have evolved to meet the demand for simulating complex robotic systems and collecting diverse, large-scale, and realistic data for training and testing deep learning algorithms [28]. These environments aim to provide visually realistic and physically accurate simulations, simulating the complexity of the real world [29]. They also allow for the management of realistic

[1] https://community.gazebosim.org/t/a-new-era-for-gazebo/1356.
[2] https://www.blackcoffeerobotics.com/blog/migration-from-gazebo-classic-to-ignition-with-ros-2.

virtual environments where robotic simulations can be performed, without being biased by synthetic data [29]. The use of accurate simulations in robotics helps in the development, programming, testing, and validation of robotic projects, reducing development time and cost.

While simulation environments provide a cost-effective and safe means to test and design robotic solutions, they also present significant challenges, particularly in accurately simulating the complex dynamics of real-world interactions [1]. This limitation underscores the ongoing need for advancements in simulation technologies that can bridge the gap between virtual testing and real-world applications. Recent comparative studies further highlight the evolution and differentiation among leading simulation platforms. Lenka Pitonakova et al. provided a feature and performance comparison of V-REP, Gazebo, and ARGoS, underscoring Gazebo's superior performance in handling complex scenes crucial for simulating detailed and dynamic interactions in humanoid robotics [30]. Similarly, Angel Ayala et al. conducted a quantitative analysis comparing Gazebo with Webots and V-REP, noting that although Webots uses fewer resources, Gazebo provides better support for complex simulations, aligning with the needs for simulating advanced robotics systems like humanoid robots [31]. Moreover, Nathan P. Koenig and A. Howard discussed the design and usage paradigms for Gazebo, an open-source multi-robot simulator, emphasizing its ability to recreate complex worlds encountered by the next generation of mobile robots [32]. This capability uniquely positions Gazebo in offering high fidelity and fine-grained control over simulation parameters, distinguishing it from other simulators. Mirella Santos Pessoa de Melo et al. echo this idea, offering a comprehensive comparison of Gazebo with V-REP and Unity and highlighting Gazebo's superior performance in complex simulation scenarios, particularly in its integration with the Robot Operating System (ROS), which is critical for developing sophisticated robotics applications [33].

Building upon the strengths highlighted in recent comparative studies, the integration with advanced frameworks has significantly enhanced the capabilities of simulation environments, particularly in fostering highly realistic and dynamic simulation settings essential for humanoid robotics research. Notably, the introduction of Gym-Ignition by Diego Ferigo et al. has leveraged Gazebo's flexibility to craft reproducible robotic environments optimized for reinforcement learning, integrating over 100 modeling and simulation tools to boost the real-world applicability of simulations [34]. Further enhancing interoperability, Lange et al. have implemented the Functional Mock-up Interface (FMI), enriching the ROS and Gazebo community by bridging Gazebo with other modeling tools [35]. Additionally, the synergy between ROS and MoveIt within Gazebo, as discussed by M. Marian et al., has improved functionality in dynamic interaction and manipulation tasks, facilitating complex robotic behaviors necessary for different applications [36]. The integration efforts extend to the work of Christoffer Brohus Kristensen et al., who developed a Gazebo-based framework for robotic unscrewing tasks using reinforcement learning, thereby creating environments that accurately mimic real-world complexities [37]. Finally, the introduction of PIC4rl-gym by Mauro Martini et al., utilizing ROS 2 and Gazebo, underscores the platform's advanced integration with deep learning techniques to enhance autonomous navigation [38]. These developments collectively underscore evolutions in robotics simulation, supporting cutting-edge demands in modern robotics research and advancing the capabilities of simulated environments to mirror complex real-world applications.

By using these simulation environments and tools, researchers can test their code, algorithms, and hardware designs more convenient and can verify their codes before launching them to the robot. One of the most prominent robot for simulation environment is the Robot Pepper designed by SoftBank Robotics [39,40]. Formed with multiple sensors, actuators, and communication capacities, the Robot Pepper is highly adaptable and can perform a variety of tasks. Its capacity can be increased using simulation environments and real-world problems. DTPAAL project harnessed VPepper, the virtual replica of the robot, for anomaly detection and remote support, linking real and virtual testing settings [18]. Lier et al. tackled the lack of experience in this area by developing a simulation environment for the robot pepper to test new algorithms [14]. Furthermore, researchers like Silva et al. developed an online navigation framework that will enable robots like Pepper to interact with people in crowded indoor spaces in a socially acceptable manner [41]. The PePUT toolkit, integrating Unity and Python, allows for enhanced control and adaptability in programming the Pepper robot, offering developers substantial flexibility in robot behavior modeling and functional regression testing [42]. Meanwhile, initiatives like the RoboCup competition challenged Pepper in real-world settings, enhancing its skills and interaction capabilities through the development of sophisticated software systems and components [43]. By integrating technologies like ROS and cloud services, Pepper's autonomy, environmental awareness, and user interaction capabilities can be significantly improved [44]. Each of these projects and tools play a crucial role in both simulating and directly enhancing the functionalities of service robots like Pepper, providing valuable platforms for both development and practical application.

Current simulation environments for the Robot Pepper, while useful, exhibit several limitations that extend beyond the integration of complex machine learning algorithms. They aren't fully capable of faithfully simulating Pepper's complex behaviors and interactions, which is necessary to perform robot testing and development in a controlled manner. Furthermore, current platforms are closed-source, which means that their capabilities are limited and the robotics and scientific communities imply little input in their development [44]. Existing Pepper simulations that use ROS1 middleware lack both the fidelity and the modularity required for integrating and testing advanced machine learning algorithms [45]. Moreover, the machine learning tools available in ROS1 are suited only for simpler tasks, often requiring substantial custom modifications or the use of external frameworks like PyTorch or TensorFlow to handle more sophisticated algorithms [46]. These issues underscore critical gaps in performance, community support, and modularity within ROS1, which impede real-time execution and the seamless integration of complex models. The absence of ROS 2 middleware in available Pepper simulations further highlights the need for updated solutions that can fully leverage advancements in machine learning and address the growing demand for the robot Pepper with advanced capabilities. Moreover, the evolving landscape of ROS distributions reinforces the transition to ROS 2, marked by major up-dates and enhancements since 2017. While the last ROS1 distribution, Noetic, is officially supported until May 2025, the community is expected to shift to ROS 2 for its improved architecture, signaling a forward-looking choice for robotic applications [47]. For a detailed comparison of the specific limitations encountered in these various environments, refer to Table 1.

Our proposed solution to the identified challenges-limited integration of complex machine learning, inaccurate simulation of Pepper's behaviors, closed-source nature, limitations of ROS 1 middleware, limited machine learning tools in ROS 1- is to develop a ROS 2 package for the Pepper robot. This package will integrate MoveIt 2 and Gazebo Sim, creating a high-fidelity simulation environment that most accurately reflects Pepper's real-world behavior. As an alternative to ROS 1 constrained environments, ROS 2 guarantees realistic training for the demanding machine learning application and platform robustness, as it offers flexibility and modularity. The integrated machine learning algorithms and frameworks can be used and extended through our system. So, in comparison to the state of the art at the moment, our approach offers several novel elements, as demonstrated by our thorough study in this section.

- High-Fidelity Simulation: With Gazebo Sim and MoveIt 2, the simulation environment uses sophisticated physics engines and high-resolution 3D models to simulate Pepper's behaviors with an unprecedented level of realism. This configuration surpasses the physical accuracy and complexity restrictions of earlier platforms, ensuring accurate physical interactions and sophisticated behavior modeling.
- Integration with ROS 2: Unlike existing simulations that rely on the outdated ROS 1 middleware, this system leverages ROS 2, offering improved architecture, real-time performance, and enhanced modularity. This integration ensures robust and scalable simulation environments that can handle advanced robotic applications.
- Open-Source Accessibility: In contrast to many other closed-source systems now in use, the platform is completely open-source. This transparency encourages community involvement and ongoing development, allowing academics and programmers to add to and enhance the simulation environment's capabilities.
- Future Integration Capability: While the current focus is on creating a high-fidelity digital twin, the simulation environment is designed with the modular architecture of ROS 2, allowing for the flexibility to support the future integration of complex machine learning algorithms. This design choice ensures that the platform remains relevant and adaptable to emerging research needs.

It is important to note that, in this paper, we focus on the creation of the open-source digital twin of the Pepper robot, and we do not include specific ML algorithms. The intention is to present the identified gap and provide a foundation for future work, where the integration of ML algorithms into the developed platform is envisioned. This emphasis on future work is driven by our commitment to advancing research and fostering community collaboration. By making our digital twin accessible to the community, we aim to catalyze further research and development in the field of robot learning.

3. Development of a High-Fidelity ROS 2 Simulation Environment for the Pepper Robot

2.1. *Choosing the ideal simulator: evaluating Gazebo Sim for enhanced robotics simulation*

Multiple studies have systematically compared simulation software for robotic arm manipulation using ROS 2, focusing on benchmarking under similar parameters, tasks, and scenarios [17,54]. Evaluation criteria include long-term operations, task completion success, repeatability, and resource usage. While no overall best software exists, Gazebo Sim and Webots consistently demonstrate higher stability. Regarding resource usage, PyBullet and CoppeliaSim outperform competitors in terms of efficiency.

It's important to highlight that Table 2 accompanies these findings of Audonnet et al. [54], presenting an overview of simulation software and their capabilities. This table serves as a comparative overview of simulation platforms based on the review made by Audonnet et al. [54] enhanced by other relevant sources.[3] This table is set for some major simulation platforms: Gazebo Classic, Gazebo Sim, Webots, Isaac Sim, Unity, Pybullet, CoppeliaSim (Vrep), and Mujoco, comparing the features of the physics engine, headless support, open source, ROS 2 support, and machine learning support. It underlines the diversity of the Physics Engine, Headless level of support, Open-source availability, and ROS 2 compatibility from highlighting which robotic and machine learning applications it best suits. This comparison, therefore, does not only serve to help platform users in the application of specific technical needs and corresponding software compatibilities but also provide insight into the larger trend of a more open and versatile trend in development environments of simulation technology. That helped us determine which simulation software is best suited for the creation of the open-source digital twin of the robot Pepper with ROS 2. Based on the needs of our ROS 2 open-source package, on the comparison in Table 2 and on the results depicted by Audonnet et al. earlier in the paragraph [54], we narrowed our choice between Gazebo Sim and Webots since they demonstrate the higher stability.

In pursuit of this goal, we compared Webots and Gazebo Sim in Table 3, both open-source simulation environments in order to provide insights into our decision-making process, explaining why Gazebo Sim was chosen over Webots for our particular use case based on the exhaustive comparison made by Audonnet et al. [54].

In our comparative analysis of robotic simulation platforms, we found that both Webots and Gazebo Sim offer robust features suitable for various educational and research applications. However, for our specific aim to produce an open-source ROS 2 digital twin of the Pepper robot, Gazebo Sim is the most appropriate choice. This decision was influenced by Gazebo Sim's efficient resource usage

[3] https://gazebosim.org/api/gazebo/3.3/physics.html.

Table 1
Comparison of pepper Robot simulation environments: Focus, software, and limitations.

Paper Title	Application	Focus	Simulation Software	Limitations	Supporting Evidence from Related Works
DTPAAL: Digital Twinning Pepper and Ambient Assisted Living [18].	Anomalous Situation Detection and Remote Support.	Development of VPepper, a virtual replica, for anomaly detection and remote support.	- Unity 3D. - Physics simulation environment.	- Limited Physical Accuracy. - High-Dimensionality Challenges. - Not open source.	Unity requires more setup for robotics tasks due to its limited focus on the field [33].
Towards an Open Simulation Environment for the Pepper Robot [14].	Navigation in Simulated Apartment and People Detection.	Addressing the need for a simulation environment for testing new algorithms.	- Morse. - Blender. - ROS 1. - NAOqi.	- ROS 1 Preference. - Interface compatibility issues with existing NAOqi systems.	- Compatibility limitations of NAOqi [48]. - MORSE based simulation focuses on human-robot interactions and not on reliable physics [49].
Online Social Robot Navigation in Indoor, Large and Crowded Environments [41]	Enhancing robot navigation in indoor social spaces.	Developing an online navigation framework for robots like Pepper to behave socially appropriately in crowded indoor environments.	- Gazebo. - PedsimROS.	- PedsimROS is integrated with ROS1.	- Teleoperating humanoid robots using ROS for complex tasks creates data gathering and control challenges, hindering data collection for robot learning [53]. - PedsimROS is a simulator integrated on deprecated versions of ROS1 [50].
PePUT: A Unity Toolkit for the Social Robot Pepper [42].	Virtual Testbed for Social Interactions.	Introducing PePUT, a toolkit for controlling Pepper through Unity and Python.	- Unity UI. - Python. - NAOqi.	- Rendering issues in Animation Editor when querying real robot pose. - Restricted to Unity environment, which is not open-source	The software environment of NAOqi is not fit for advanced algorithms and the like. Instead, ROS is designed for these kinds of implementations and has many advantages [51].
Collision Avoidance for Indoor Service Robots Through Multimodal Deep Reinforcement Learning [52]	Indoor Collision Avoidance.	Implementing collision avoidance using Deep Reinforcement Learning.	- Gazebo. - ROS 1.	- ROS 1 Preference. - Reality Gap.	- Traditional ROS-based DRL frameworks lack features for efficient training needed for advanced learning techniques [52]. - Reliance on accurate sensory observations can be a limitation in dynamic environments [53].

Table 2
Overview of the simulation software and their capabilities.

Name	Physics Engine	Headless Support	Open Source	Ros 2 Support	ML support
Gazebo Classic	Bullet, DART, ODE, Simbody	Full	Yes	Yes	External
Gazebo Sim	DART	Full	Yes	Yes	External
Webots	ODE	Partial	Yes	Yes	External
Isaac Sim	PhysX	Full	No	Yes	Integrated
Unity	Havok, PhysX, RaiSim	Full	No	No	External
Pybullet	Bullet	Full	Yes	No	External
CoppeliaSim (Vrep)	Bullet, Newton, ODE, Vortex Dynamics	Full	No	Yes	External
Mujoco	Mujoco	Full	Yes	No	External

Table 3
Webots vs. Gazebo Sim: Choosing the Advanced Robotics Simulation Platform.

Feature	Webots	Gazebo Sim
Open Source	Yes	Yes
GUI	Comprehensive	Less extensive
Documentation	Extensive	Good
Physics Engines	ODE	DART
Headless Mode	Supported	Supported
Hardware Load Acquisition	Not ideal if execution time and resource usage are critical	Efficient in resource usage, suitable for ML
Accuracy & Stability	performs well in long-term operations and task repeatability	Well-suited for constant and slow-moving tasks

and its suitability for machine learning applications, as it performs well in tasks that require consistent and slow-moving dynamics. Furthermore, Gazebo Sim's compatibility with ROS 2 and its ability to handle complex simulations with fewer computational resources align perfectly with the demands of developing a high-fidelity digital twin, ensuring that our project remains scalable and performant under varied simulation scenarios. Moreover, featured with Rviz2 through ROS 2, Gazebo Sim ensures that changes in Gazebo Sim are published to ROS 2 topics and then visualized in RViz2, allowing for coordinated updates rather than immediate reflections in visualization which contributes to a precise monitoring and debugging during development.

2.2. *Overview of ROS 2 simulation architecture for pepper Robot*

Building upon the challenges identified in the literature review section regarding Pepper's limitations in manipulation within existing simulations, we introduce a carefully crafted ROS 2 package architecture that not only facilitates detailed modeling and testing of the robot behaviors in a robust 3D simulation environment but also supports iterative testing, development, and refinement of the robot functionalities, ensuring that it can perform its intended functions safely and effectively before any real-world deployment.

In this architecture, ROS 2 middleware acts as the central node for communication and data flow coordination among all components. The system starts with defining the robot's structure using the Pepper_robot_description module, which contains all the URDF (Unified Robot Description Format) and SDF (Simulation Description Format) files necessary to describe the physical and visual properties of the Pepper robot. Once the robot's structure is defined, several processes run in parallel. Gazebo Sim provides real-time physics-based simulation of the robot's interaction with the environment, while RViz2 represents non-physical data, such as navigation paths and robot states, for comprehensive debugging and development. ROS 2 enables communication between RViz 2 and Gazebo Sim, where Gazebo Sim sends highly detailed simulation data on many different ROS topics. In this case, RViz 2 subscribes to those topics, receiving visual information for visual simulation experience to be both in synchrony and to present a coherent simulation experience.[4]

Fig. 1 summarizes the architecture of the Pepper robot simulation package. The central part is the Pepper_robot metapackage, where ROS 2 middleware coordinates communication and data flow among all system components. The system flow is as follows.

1. The process begins with the Pepper_robot_description module, defining the robot's physical and visual structure.
2. The Pepper_robot_ign module configures Gazebo Sim for real-world physics and dynamics simulation.
3. Simultaneously, the Pepper_robot_moveit_config component, integrated with MoveIt 2, handles higher-level motion planning.
4. ROS 2 middleware processes simulation commands sent to the Pepper_robot_ign module to ensure the execution of actions in Gazebo Sim.
5. Motion planning data from the Pepper_robot_moveit_config component is sent through the ROS 2 middleware to execute planned movements in Gazebo Sim.
6. RViz2 provides real-time visual feedback by subscribing to simulation data from Gazebo Sim.
7. The system iterates back to the ROS 2 middleware for synchronization and dynamic adjustments based on performance data, ensuring that the simulation remains accurate and responsive.

In such a configuration, on the one hand, RViz2 represents real-time visual monitoring of what the robot is doing—movements and behaviors out of the simulation data. On the other hand, the Pepper_robot_ign module sets up Gazebo Sim for simulating the robot with accurate real-world physics and dynamics. Higher-level motion planning is under the responsibility of the Pepper_robot_moveit_config component, which, when tied with MoveIt 2, calculates and performs movements of the robot according to the current state and wanted tasks. The motion planning data and simulation commands are processed through ROS 2 middleware, ensuring execution in Gazebo Sim. Iterations with the ROS 2 middleware synchronize the system, providing dynamic adjustments to the robot's motion planning in real-time based on performance data. This cohesive flow, illustrated in Fig. 1, outlines how different modules under the pepper_robot metapackage work to simulate, visualize, and control the Pepper robot, all within one ROS 2 environment. The source code for the entire ROS 2 package, including detailed implementations and configuration files, is available on our GitHub repository.[5]

2.3. *Comprehensive development of the ROS 2-based simulation system for pepper*

This section provides a comprehensive exploration of the ROS 2 package architecture, elucidating how each component contributes to simulating key aspects of Pepper's manipulation in simulated environments. While Fig. 1 provided an overview of the system flow within the developed package and a brief description of the sub-packages, Fig. 2 provides finer-grain detail of each of the sub-packages of the *pepper_robot metapackage: pepper_robot_description, pepper_robot_ign, pepper_robot_moveit_config.* This is going to consider core functionalities of these components, without redundantly covering the system flow already illustrated in Fig. 1. This in-depth representation lays the foundation for a closer examination of each core component.

2.3.1. *Detailed architecture analysis of ROS2-Based packages for Pepper robot Simulation*

Following the overview of the system flow, this section takes a closer look at the architecture and the most important packages in

[4] https://gazebosim.org/docs/fortress/ros2_interop.
[5] https://github.com/HibaSekkat/pepper_ign_moveit2.git.

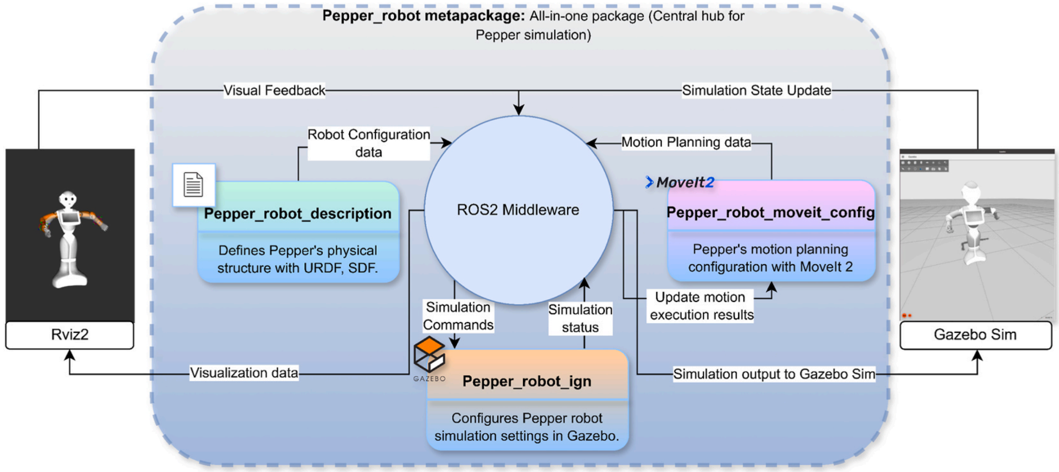

Pepper_robot metapackage: All-in-one package (Central hub for Pepper simulation)
Visual Feedback
Simulation State Update
Robot Configuration data
Motion Planning data
ROS2 Middleware
Pepper_robot_description
Defines Pepper's physical structure with URDF, SDF.
MoveIt2
Pepper_robot_moveit_config
Pepper's motion planning configuration with MoveIt 2
Update motion execution results
Simulation Commands
Simulation status
Rviz2
Gazebo Sim
Visualization data
GAZEBO
Pepper_robot_ign
Configures Pepper robot simulation settings in Gazebo.
Simulation output to Gazebo Sim

Fig. 1. ROS2 system architecture for Pepper robot simulation.

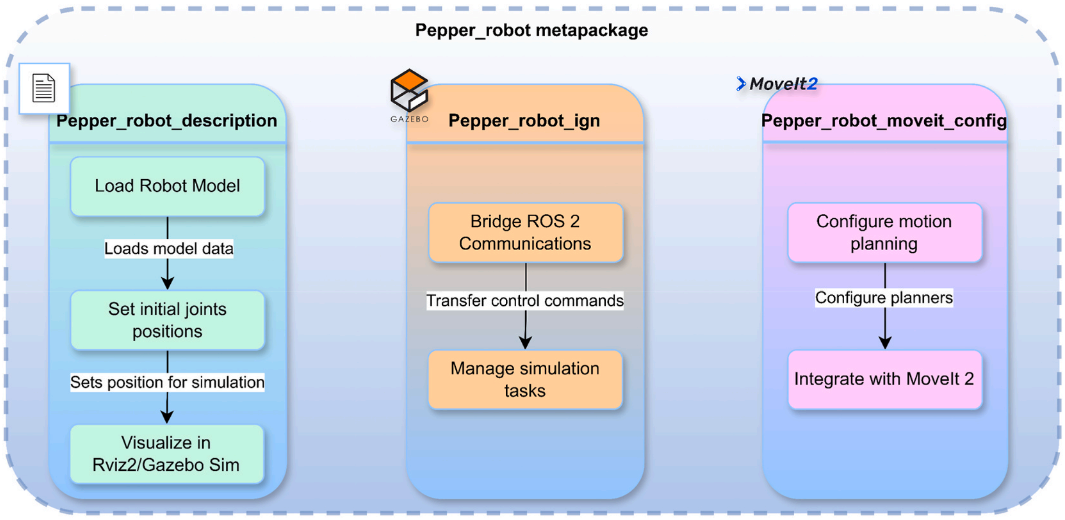

Pepper_robot metapackage
Pepper_robot_description
Load Robot Model
Loads model data
Set initial joints positions
Sets position for simulation
Visualize in Rviz2/Gazebo Sim
GAZEBO
Pepper_robot_ign
Bridge ROS 2 Communications
Transfer control commands
Manage simulation tasks
MoveIt2
Pepper_robot_moveit_config
Configure motion planning
Configure planners
Integrate with MoveIt 2

Fig. 2. Component-specific workflows within the Pepper robot simulation system.

the simulation framework. It explains how each of the elements will contribute to constituting a strong and at the same time flexible simulation environment suited for our cutting-edge robotics research.

a. Pepper_robot Metapackage: Central Coordination of Robotic Descriptions

This metapackage of the pepper robot is at the heart of our simulation architecture. This guarantees that robot sub-packages can be used seamlessly in all the simulation scenarios studied. This package is very important as it mechanizes the transformation and validation of robot descriptions throughout the build process, hence improving the level of efficiency in development and eliminating the chances of errors due to manual processes. This package enables the automatic transforms to ensure that robot descriptions always represent current specifications correctly without human intervention.

The metapackage pepper_robot handles all dependencies of the respective package so that the code base stays clearly organized and clearly navigable. This very kind of centralized management approach, therefore, makes the maintenance and updating of the system pretty easy, since all the changes made across modules synchronize smoothly. It allows the development cycle to be shortened by one hand through automation and centralization and increases the overall reliability and maintainability of the complete simulation system. This is an indispensable feature of our high-level robotic simulation framework.

b.Pepper_robot_description: Crafting a Digital Twin for Simulation

The Pepper_robot_description package accurately and with a high level of detail describes the digital twin of the Pepper robot needed to perform the simulation work with accuracy and effectiveness. The package fully specifies the kinematic structure of the robot, including all links and joints, as well as its visual appearance based on meshes. These make the following definitions applicable to different simulation environments, even in human-like applications in research and development, through enabling their application as in Fig. 3a,b.

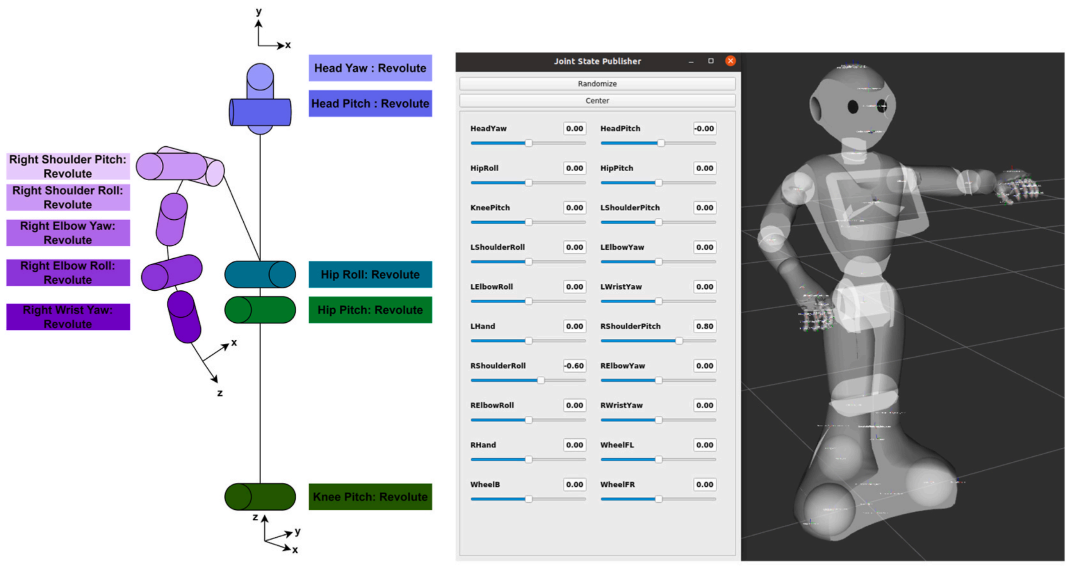
This package is very critical in starting simulations in Gazebo Sim. The package loads model data from URDF (Unified Robot Description Format) and SDF (Simulation Description Format) files, which specify the physical dimensions and properties of the Pepper robot. These joint positions' initial conditions are of importance to the robot in the simulation, whereby it starts under consistent and controlled conditions. The Pepper_robot_description further allows the configuration of the visual parameters in RViz2 using middleware ROS 2, thus providing real-time updating of the visualizations with data processed from Gazebo Sim. This dynamic updating is a feature critical to the real-time monitor of the robot's state and robot's movements during simulations, providing instant visual feedback, something that is absolutely vital during testing phases and iterative development.

It doesn't just describe the kinematic and aesthetic details but also brings in essential physical properties such as mass distribution, which will be required for calculation of inertia tensor, expressed as a 3x3 matrix $= \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix}$. This tensor has important roles in dynamic simulations. It is used by Gazebo Sim in simulating the robot response to applied forces and torques using Newton-Euler dynamics:

$$F = ma + I\alpha, \tau = I\alpha + \omega \times I\omega \qquad [1]$$

Where, F denotes force, τ signifies torque, m represents mass, a represents linear acceleration, ω denotes angular velocity, α denotes angular acceleration, I is the inertia tensor, and $\omega \times I\omega$ represents the cross product of angular velocity and the product of inertia tensor and angular velocity, describing the rotational effect influenced by mass distribution. The force equation describes how the force applied to an object results in its linear acceleration and accounts for rotational effects via the inertia tensor and angular acceleration. The torque equation explains how the torque applied to an object generates angular acceleration, factoring in the object's resistance to change in rotation (inertia) and the rotational effects influenced by the distribution of mass (cross product term). The accurate representation of these dynamics is key to the simulation of real-world physics interactions and will directly impact issues like momentum, stability, and impact dynamics.

It is made easy for setting up and testing the simulation by including launching scripts in the package, which automatically visualizes in both RViz2 and Gazebo Sim, hence efficient and effective simulations. This capability of model representation through a dual-format model (URDF and SDF) promises compatibility among different simulation platforms and opens the way for integration with different tools and environments such as Gazebo and Fuel. Ultimately, it's detailed and accurate model descriptions form the necessary element for simulating complex interactions, motion planning, and control strategies in a virtual setting, making it a cornerstone of our advanced robotics simulation framework.



HeadYaw	0.00	HeadPitch	-0.00
HipRoll	0.00	HipPitch	0.00
KneePitch	0.00	LShoulderPitch	0.00
LShoulderRoll	0.00	LElbowYaw	0.00
LElbowRoll	0.00	LWristYaw	0.00
LHand	0.00	RShoulderPitch	0.80
RShoulderRoll	-0.60	RElbowYaw	0.00
RElbowRoll	0.00	RWristYaw	0.00
RHand	0.00	WheelFL	0.00
WheelB	0.00	WheelFR	0.00

a. Head and Right Arm in the Spotlight: Modeling Pepper's Kinematics with Joint Types

b. Rviz2 Visualizations of Pepper's Kinematics and Joint Limits

Fig. 3. From code to reality: Unveiling Pepper's kinematics and joint limits.

c.Pepper_robot_ign: Facilitating Communication Between ROS 2 and Gazebo Sim

The pepper_robot_ign package is an important interface in the ROS 2 framework, which serves as the bridge between Gazebo Sim and control algorithms within the ROS 2 environment. These ensure that there is no disruption in the seamless flow of data and commands between control systems and simulation modules. This improves the fidelity of the simulation with exact control and real-time feedback. This also provides a simulation that is able to be adapted or even customized to run through the provision of a configurable bridge script as part of the simulation setup, allowing several scenarios and control strategies. It helps doing realistic simulations by bringing forth the powerful integration of ROS 2 and Gazebo Sim through *Pepper_robot_ign*. Such control algorithms within ROS 2 can directly manipulate the virtual environment in getting feedback, thereby making the simulations responsive and interactive.

This integration supports the behavior simulation of the autonomous robots and their complex dynamics of interactions with the environment; it showcases capabilities of the advanced motion planning tools like MoveIt 2. Besides movement control and managing the interaction of the robot, *Pepper_robot_ign* also manages the flow of sensor data to support real-time control and sensing that gives the simulation its realistic impression, in addition to the Pepper robot personality. Such real-time capability is very much important for those systems whose applications require immediate feedback, as in the case of the adaptive control system and the interactive training environment.

On the other hand, it enhanced the visualization capabilities in the simulation framework. It gives support to RViz2 visualization, providing insight into the robot's actions and planning processes for use in debugging and control strategy refinement. It is important to be able to make sense of how the robot is interfacing with the environment and gain insights from the visual feedback on improved design and execution of the simulation.

Overall, the package *pepper_robot_ign* acts as the communication backbone of the simulation architecture, correctly flowing both data and commands across the system. To maintain veracity and uniformity of behavior within the simulation, this coordination is required, enabling the robot to respond properly towards changing tasks of the simulation and control inputs. This is a very important package in the field of advanced research and development of robotics, which through its wide function, will raise the simulation environment to a new level.

d.Pepper_robot_moveit_config: Orchestrating Advanced Motion Planning with MoveIt 2

The *Pepper_robot_moveit_config* module is one of the key modules within the architecture of our simulation. It provides the Pepper robot with advanced motion planning capabilities using the MoveIt 2 framework.

This module has been designed to configure and integrate a number of different MoveIt 2 components into the robot manipulators, enabling the creation and execution of elaborate, detailed, and accurate motion plans. More in this context, the current module tunes the motion planning algorithms and the set of parameters in relation to the capabilities of the Pepper robot and specific demands of the simulation. It therefore selects and parametrizes the MoveIt 2 planners for operation scenarios of the robot to be able to dynamically plan and simulate complex robot motion and interaction. This careful configuration makes it possible for motion plans to be feasible and collision free, hence allowing for complex dynamics of robot manipulation.

The *Pepper_robot_moveit_config* increases its capabilities by allowing the handling of SRDF (Semantic Robot Description Format) in a flexible way, such that the simulation can adjust itself to different robot configurations and different situations that may occur. Such adaptation is necessary since many customizations and high precision are needed for such simulations. It also allows the high level of visualization and interaction within the simulated environment. It's integrated with RViz2 and therefore helps to visualize the planned motions and the state of the robot. This visualization is powerful because it gives insight into the operational dynamics of the process of motion planning in a manner that assists in understanding and debugging. The visual feedback allows researchers and developers to iteratively adapt their strategies toward improved robot performance for the simulated tasks.

In addition, the *Pepper_robot_moveit_config* allows the development of very complex behaviors for the robots, right from path planning, obstacle avoidance, to manipulation tasks. All these are highly enhanced with the development of a strong background on which these capabilities are built, hence robotic research can be easily carried out simulating high-level tasks with an ability to represent real-life implementations. In general, the module extends simulation capabilities of the advanced motion planning to make sure that these capabilities are well integrated within a broad simulation framework. The *Pepper_robot_moveit_config* allows precision and adaptability in the development and testing of advanced robotic applications within the ROS 2 environment.

3. Validation and accuracy assessment of Pepper robot digital twin

In recent years, the concept of digital twins has gained prominence for their role in analysis, prediction, and optimization [55,56]. This research takes a pivotal step in developing a precise digital twin for the Pepper robot, leveraging the robust capabilities of ROS 2, Rviz2, Gazebo Sim, and MoveIt 2. The cornerstone of our endeavor lies in a comprehensive demonstration and validation process, ensuring the accuracy and efficacy of our digital twin. This scrutiny is indispensable, providing the confidence needed to fully exploit its potential for tasks in robot learning and manipulation.

3.1. Setup and environment

The research leveraged a software stack of Rviz2, ROS 2 Galactic, Gazebo Sim (Fortress), Python 3.8.10, and Docker 20.10.21, executed on a Dell Precision-5820 workstation boasting an Intel Xeon w-2155 CPU (3.30 GHz) and 31 GB of memory. Ubuntu 20.04.4

LTS, a popular and reliable choice in robotics, served as the operating system, while Docker 20.10.21 ensured consistent and reproducible software dependencies across different machines. This powerful hardware, with its high-performance CPU and ample memory, allowed for efficient execution of simulations, handling large datasets with ease. Ultimately, this carefully chosen configuration paved the way for replicable results and smooth experimentation.

3.2. *Gazebo Sim digital twin validation*

A crucial facet of Pepper's movement capabilities centers on joint limits, defining its range of motion. Considering the feedback that both Gazebo Sim and Rviz2 rely on the same URDF/SDF model, a separate comparison with Rviz2 might be redundant. Therefore, we focused on directly comparing Choregraphe's planning instructions with Gazebo Sim's execution to pinpoint any inconsistencies within the planning stage itself. Our meticulous comparison focused exclusively on Choregraphe, acknowledged as the primary source of joint limit data for Pepper, providing a distinct benchmark for evaluating Gazebo Sim's fidelity in accurately reflecting these limits. The findings, presented in Table 4, shed light on the intricate correspondence between our Gazebo Sim-based digital twin and the bench-marks set by Choregraphe. This analysis not only underscores the precision of our simulation but also establishes a robust foundation for an impactful demonstration and validation narrative, setting the stage for further exploration into the capabilities of our digital twin.

We evaluate the average deviation of the simulated limits for each joint from predefined benchmarks using a Mean Absolute Error (MAE) technique. Specifically, we employ the MAE measure to provide an explicit interpretation of the findings and a straightforward representation of the error magnitudes. In order to provide a clear indicator of accuracy along each joint's range of motion, it specifically computes the average absolute disparities between the benchmarks from Choregraphe and the outcomes from Gazebo Sim simulation for each joint limit—minimum and maximum.

These mean absolute errors for each joint, resulting from the absolute value of the difference of the corresponding minimum and maximum joint limits between Choregraphe and Gazebo Sim, are indicative of how close the digital twin is. For instance, an error of 0.01–0.02 radians indicate an average deviation of the Gazebo Sim simulation from experimental data by such a small amount. Such very low values of MAE give confidence in the very high fidelity and high reliability of the digital twin in reproducing real-world dynamics, which becomes key in high-precision tasks including interaction and complex manipulation within different.

Furthermore, talking about how these MAE values might affect the robot's operational needs and other aspects of its design strengthens our belief in the usefulness of our digital twin. Low MAE implies that the errors are very minimal, further confirming that it will not compromise the real performance of the robot in any case [57]. The robustness of the quantitative analysis, based on the MAE, underpins very robustly our validation metrics in respect of suitability in very advanced simulations and practical applications in robotics.

3.3. *Unveiling Movement Fidelity: from planning to reality*

In this paper, the precision of the Gazebo Sim model of Pepper was measured carefully with respect to real-world movements, as the movements were recorded through a high-definition video of the robot using high-definition camera at 1080p resolution and 60 fps. With this setup, the Pepper's joint movements were recorded at particular time intervals (from T_0 to T_4). The high-definition video is a component of the system set up that was used to measure and compute the joint angles of Pepper in real-world scenarios as illustrated in Fig. 4. The system processed and computed the joint angles by using a set of specialized hardware and software components in conjunction with MPU6050 sensors that were placed on each joint of Pepper. The MPU6050 sensors, which are positioned thoughtfully on each joint of Pepper and are crucial to the capture of six-axis motion data (three axes of acceleration "A_x, A_y, and A_z" and three axes

Table 4
Pepper Robot joint limit fidelity: Gazebo sim simulation compared to choregraphe Reference values.

Joints	Choregraphe		Gazebo Sim		MAE		
(in rad)	**min**	**max**	**min**	**Max**	**MAE_min**	**MAE_max**	**Average MAE**
Head Yaw	−2.07	2.07	−2.09	2.09	0.02	0.02	0.020
Head Pitch	−0.69	0.63	−0.71	0.64	0.02	0.01	0.015
Hip Roll	−0.50	0.50	−0.51	0.51	0.01	0.01	0.010
Hip Pitch	−1.02	1.02	−1.04	1.04	0.02	0.02	0.020
Knee Pitch	−0.50	0.50	−0.51	0.51	0.01	0.01	0.010
LShoulder Pitch	−2.07	2.07	−2.09	2.09	0.02	0.02	0.020
LShoulder Roll	0	1.55	0.01	1.56	0.01	0.01	0.010
LElbow Yaw	−2.07	2.07	−2.09	2.09	0.02	0.02	0.020
LElbow Roll	−1.55	0	−1.56	−0.01	0.01	0.01	0.010
LWrist Yaw	−1.81	1.81	−1.82	1.82	0.01	0.01	0.010
LHand	0.01	0.97	0.02	0.98	0.01	0.01	0.010
RShoulder Pitch	−2.07	2.07	−2.09	2.09	0.02	0.02	0.020
RShoulder Roll	−1.55	0	−1.56	−0.01	0.01	0.01	0.010
RElbow Yaw	−2.07	2.07	−2.09	2.09	0.02	0.02	0.020
RElbow Roll	0	1.55	0.01	1.56	0.01	0.01	0.010
RWrist Yaw	−1.81	1.81	−1.82	1.82	0.01	0.01	0.010
RHand	0.01	0.97	0.02	0.98	0.01	0.01	0.010

of gyroscope data), form the basis of this measuring system. The data is crucial for figuring out each joint's alignment and dynamics of motion.

The output reading from the sensors was transferred to a NodeMCU 1.0 microcontroller, one of the central pieces of our hardware setup. The NodeMCU interfaced with the MPU6050 through the effective I2C communication protocol, which has good provisions for synchronizing data from multiple sensors. The NodeMCU received the raw sensor data and used developed algorithms to calculate the angles of each individual joint. The raw accelerometer and gyroscope inputs are converted into useful angular measurements by custom-built algorithms that compute the exact angles of each individual joint using the following mathematical equations in the mathematical models that account for both initial calibration and ongoing motion dynamics:

$$\mathit{Roll}\ (\rho) = \arctan\left(\frac{A_x}{\sqrt{A_y^2 + A_z^2}}\right) \qquad [2]$$

$$\mathit{Pitch}\ (\Phi) = \arctan\left(\frac{A_y}{\sqrt{A_x^2 + A_z^2}}\right) \qquad [3]$$

$$\mathit{Yaw}\ (\psi) = \arctan\left(\frac{A_z}{\sqrt{A_x^2 + A_y^2}}\right) \qquad [4]$$

In order to bring the real-world measurements into mathematical equivalency with the simulated data, we used the rule of three. As a result, both data sets were standardized under equivalent time frames from *T*0 to *T*4, allowing a recorded frame from the real-world setup to be directly compared with its simulated counterpart.

This rigorous comparison is further detailed in Fig. 5a, b, and 5c which showcase three distinct Pepper movements across four environments: Rviz2 (planning), Gazebo Sim (simulation), Choregraphe (control), and the real world. Each movement, captured across five key frames, reveals the nuanced differences between planned intent and actual execution. Subtle variations in posture, limb angles, and trajectory emerge as the robot transitions from the digital realm to reality.

This visual analysis, complemented by Table 5's joint angle values, offers valuable insights into our Gazebo Sim model's accuracy. The table reveals close alignment between simulated and real-world data, with average discrepancies of only 0.01–0.02 radians for most movements. Even the head's wider range of motion shows remarkable fidelity, with discrepancies largely within 0.01 radians.

Delving deeper into the intricate dance of motion, Fig. 6a, b, and 6c meticulously dissect the dynamic execution of various joint movements, comparing Gazebo Sim simulations with their real-world counterparts controlled by Choregraphe. Each subplot unveils the movement trajectory of a specific joint across all five key frames, offering a nuanced comparison. The left shoulder pitch (Fig. 6a, fourth curve), for instance, exhibits remarkable fidelity between Gazebo Sim and Choregraphe across all frames. This close alignment, with deviations mostly within 0.01 radians, reaffirms Gazebo Sim's ability to accurately replicate real-world joint movements. Similarly, Fig. 6c (bottom left) scrutinizes the head pitch joint angle, revealing an elegant mirroring of trajectories between the two systems. While a keen eye might detect a narrow gap, typically hovering within 0.01–0.02 radians, this minor discrepancy pales in comparison to the overall alignment.

Continuing the analysis, Fig. 7a, b, and 7c help complete Fig. 6 by showing error graphs plotting in detail with inconspicuous units

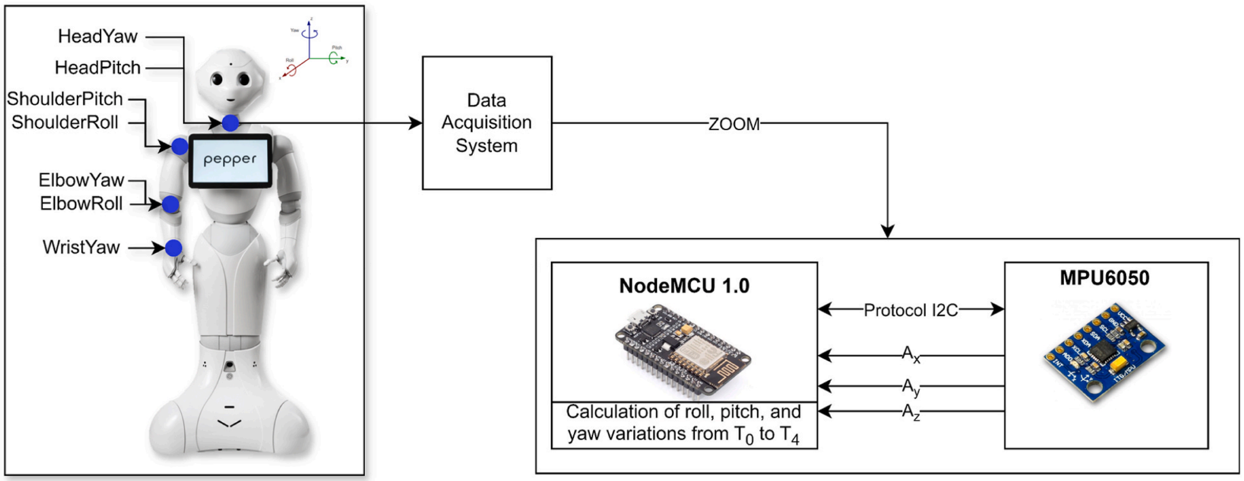
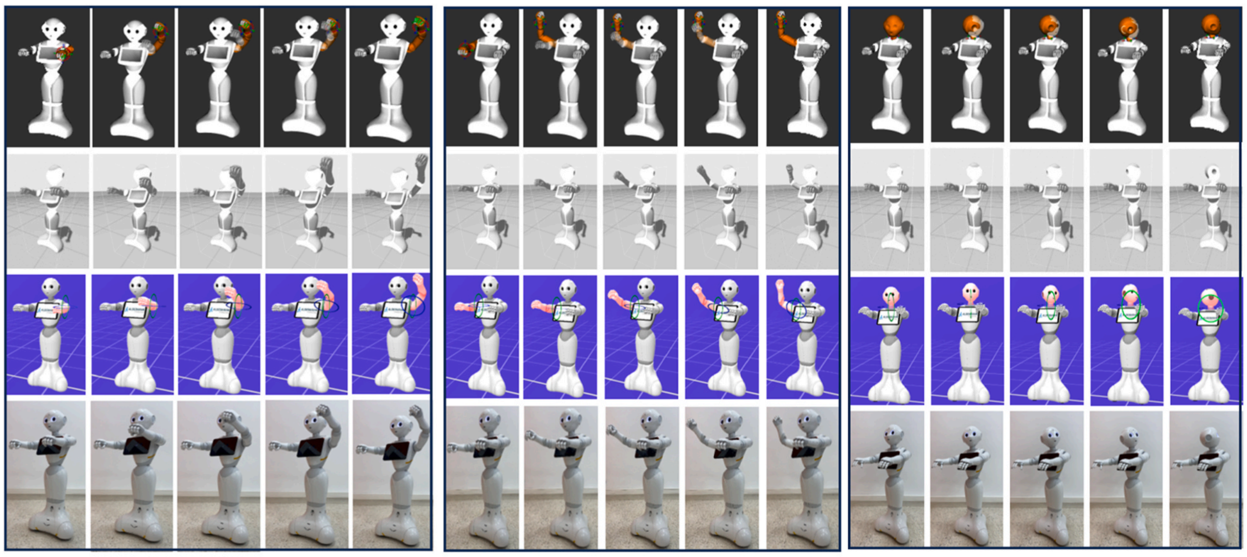


Fig. 4. Real-world calculation of joint angles in Pepper robot using NodeMCU 1.0 and MPU6050.



a. Pepper's Left Arm Movement in Rviz2, Gazebo Sim, Choregraphe, and Reality

b. Pepper's Right Arm Movement in Rviz2, Gazebo Sim, Choregraphe, and Reality

c. Pepper's Head Movement in Rviz2, Gazebo Sim, Choregraphe, and Reality

Fig. 5. From concept to control: Validating gazebo Sim's movement fidelity for left arm, right arm, and head.

in radians of robot head pitch, and yaw; and elbow, shoulder, and wrist on left and right arms, performed during 1000 iterations. The errors in the head pitch joint goes from around 0.01 radians to −0.01 radians, thus giving the impression that a transition has been made from positive error to minimally negative and then the error stabilizes. The corresponding graph below it, for the head yaw joint, begins with the error that is almost at zero, goes up to a little over 0.01 radians, and then again decreases a bit and becomes stable, which suggests a trend of initial alignment followed by small deviations occurring with adjustments. For the right arm, the 2 top graphs error starts at very near −0.01 and 0 radians, respectively, slowly rising to a peak of just over 0.00 radians–0.01 radians. This shift from a negative/null error to a positive error would imply an overshoot in the process of validation: the angle of the target would first be overcorrected in the undershoot direction and then overcorrected in the overshoot direction before it becomes stable. Such a pattern of under-correction to near-positive indicates a kind of overcompensation actually being evidenced by the simulation, but it finally stabilizes near zero. The errors are relatively high in the beginning but drastically decrease at the joints of the arm, particularly the shoulder; this seems to be due to the good modeling of dynamic behavior on these joints. The errors are low on many joints, especially on the left arm, and almost constant; this means that the simulation is accurate and stable. The detailed breakdown of error trends at the various joints shows how realistically the simulation can mimic real movements and validates the reliability of the digital twin for fine robotics applications.

Overall, Fig. 7 demonstrates that the error in each joint angle consistently falls within the narrow range of 0.01–0.02 radians. This minute deviation, further solidifying the observations from individual joint analyses, underscores the commendable fidelity of the Gazebo Sim simulations. While potential factors like sensor noise or slight control variations could contribute to these minor discrepancies, they do not diminish the overall conclusion: Gazebo Sim excels at replicating real-world robot movements with remarkable accuracy.

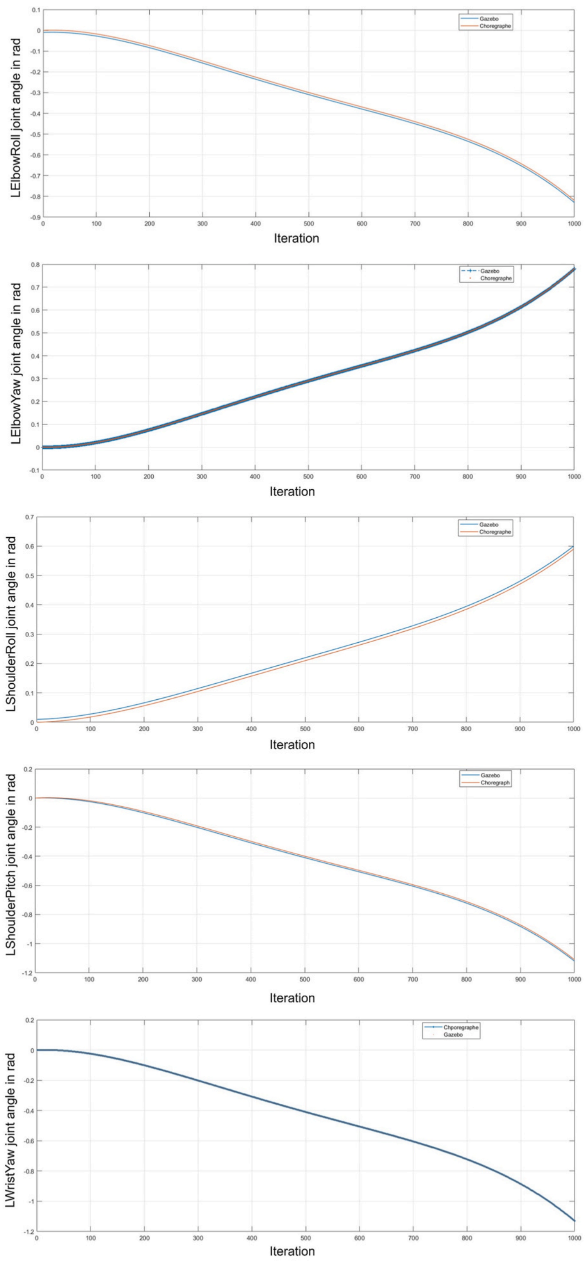
These findings highlight the power of Gazebo Sim as a robust platform for accurately replicating Pepper's movement capabilities. The close alignment between simulated and real-world data across joints and movement frames suggests that Gazebo Sim not only faithfully translates planned trajectories into virtual actions but also provides a reliable bridge between planning and real-world robot behavior. This accuracy opens doors for a multitude of applications, ranging from robot training and algorithm development in a safe simulated environment to the optimization of movement strategies for specific tasks in the real world. The remarkable fidelity between simulated and real-world Pepper movements presented here holds promising implications for advancing robotics and AI, fostering precise algorithm training and safe deployment testing.

4. Conclusion

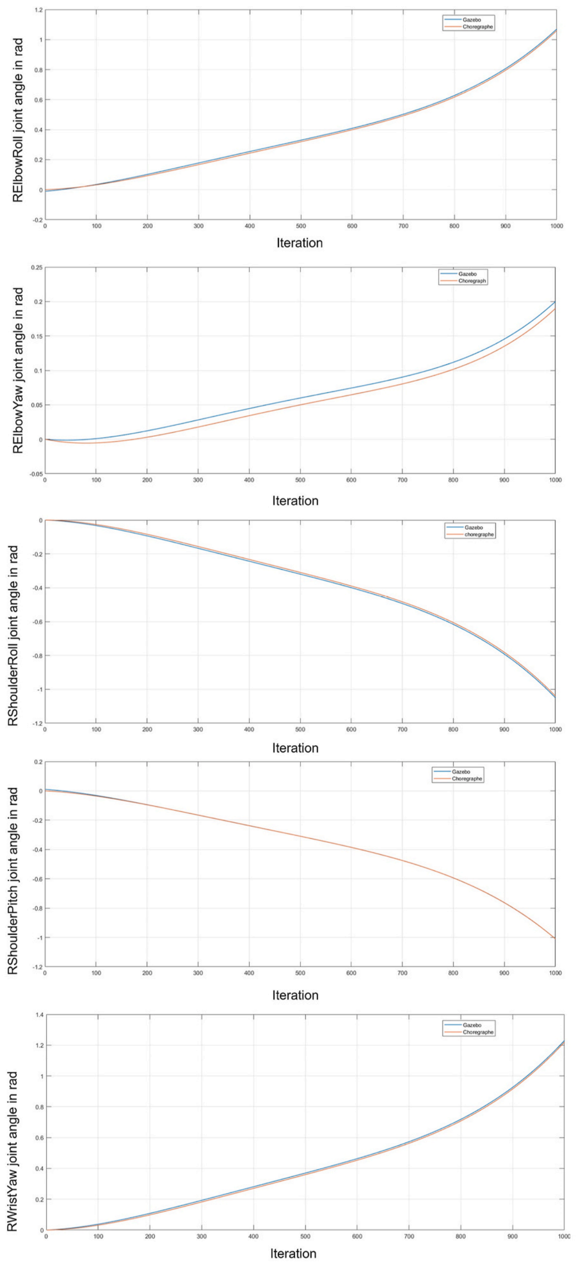
The findings presented in this paper offer compelling evidence for the transformative potential of our open-source ROS 2 package in revolutionizing humanoid robot training, specifically for dexterous robots like Pepper. The remarkable fidelity achieved between simulated and real-world Pepper movements, demonstrably exceeding the limitations of traditional ROS 1 environments, represents a significant advancement in robot simulation accuracy. The proposed system provides a high-fidelity environment essential for future training and development of machine learning algorithms. By accurately replicating real-world dynamics and enabling seamless

Table 5
Bridging the gap: Numerical validation of Pepper's joint angles across planning, simulation, and actual movement.

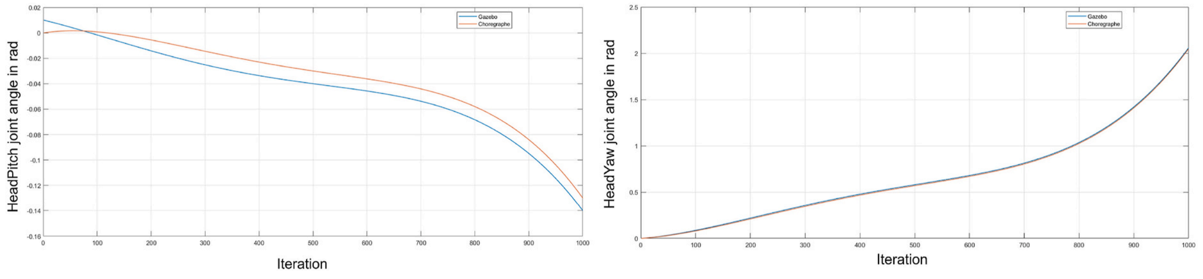
Left Arm	Rviz2					Gazebo Sim					Choregraph					Reality				
(in rad)	T_0	T_1	T_2	T_3	T_4	T_0	T_1	T_2	T_3	T_4	T_0	T_1	T_2	T_3	T_4	T_0	T_1	T_2	T_3	T_4
LshoulderP	0.00	−0.14	−0.40	−0.65	−1.11	0.00	−0.15	−0.41	−0.66	−1.12	0.00	−0.14	−0.40	−0.65	−1.11	0.00	−0.14	−0.40	−0.65	−1.11
LshoulderR	0.00	0.08	0.21	0.35	0.59	0.01	0.09	0.22	0.36	0.60	0.00	0.08	0.21	0.35	0.59	0.00	0.08	0.21	0.35	0.59
LElbowY	0.00	0.11	0.29	0.46	0.78	0.00	0.11	0.29	0.46	0.78	0.00	0.10	0.28	0.45	0.77	0.00	0.10	0.28	0.45	0.77
LElbowR	0.00	−0.11	−0.30	−0.48	−0.82	−0.01	−0.12	−0.31	−0.49	−0.83	0.00	−0.11	−0.30	−0.48	−0.82	0.00	−0.11	−0.30	−0.48	−0.82
LwristY	0.00	−0.15	−0.41	−0.66	−1.13	0.00	−0.15	−0.41	−0.66	−1.13	0.00	−0.14	−0.40	−0.66	−1.13	0.00	−0.14	−0.40	−0.66	−1.13
Right Arm																				
LshoulderP	0.00	−0.12	−0.31	−0.54	−1.04	0.00	−0.13	−0.32	−0.55	−1.05	0.00	−0.12	−0.31	−0.54	−1.04	0.00	−0.12	−0.31	−0.54	−1.04
LshoulderR	0.00	−0.13	−0.31	−0.53	−1.01	0.01	−0.13	−0.31	−0.53	−1.01	0.00	−0.12	−0.30	−0.52	−1.00	0.00	−0.12	−0.30	−0.52	−1.00
LElbowY	0.00	0.01	0.05	0.09	0.19	0.00	0.02	0.06	0.10	0.20	0.00	0.01	0.05	0.09	0.19	0.00	0.01	0.05	0.09	0.19
LElbowR	0.00	0.13	0.32	0.55	1.06	−0.01	0.14	0.33	0.56	1.07	0.00	O.13	0.32	0.55	1.06	0.00	O.13	0.32	0.55	1.06
LwristY	0.00	0.14	0.36	0.63	1.22	0.00	0.15	0.37	0.64	1.23	0.00	0.14	0.36	0.63	1.22	0.00	0.14	0.36	0.63	1.22
Head																				
HeadY	0.00	0.28	0.57	0.90	2.05	0.00	0.29	0.58	0.91	2.06	0.00	0.28	0.57	0.90	2.05	0.00	0.28	0.57	0.90	2.05
HeadP	0.00	−0.01	−0.03	−0.05	−0.13	0.01	−0.02	−0.04	−0.06	−0.14	0.00	−0.01	−0.03	−0.05	−0.13	0.00	−0.01	−0.03	−0.05	−0.13



a. Left Arm Joint Trajectories -Gazebo Sim vs Choregraphe

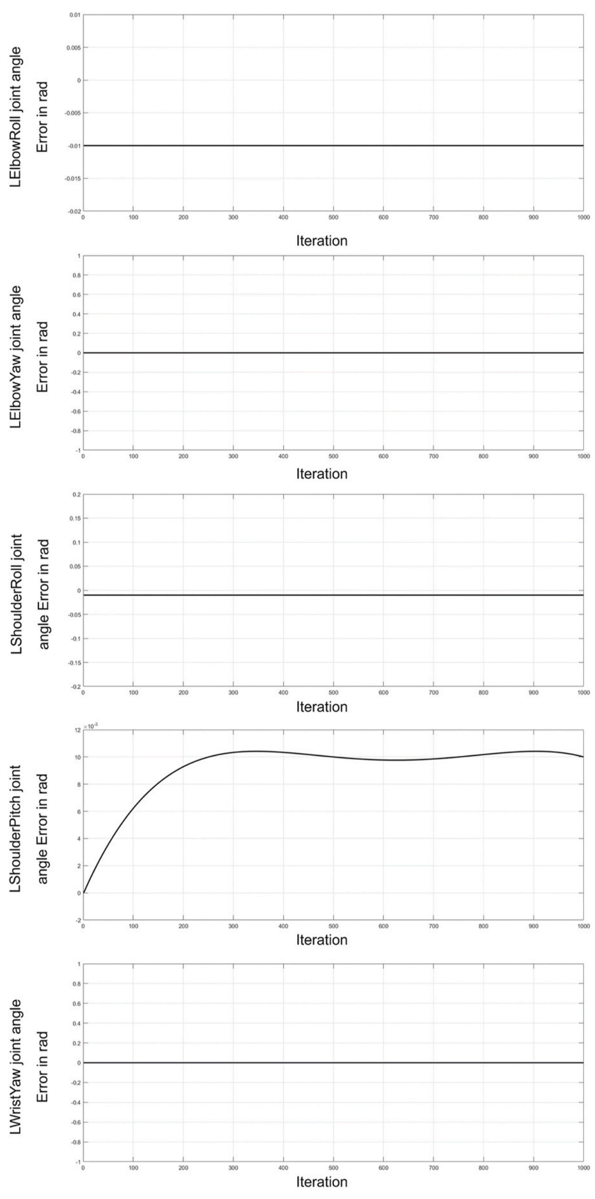


b. Right Arm Joint Trajectories -Gazebo Sim vs Choregraphe

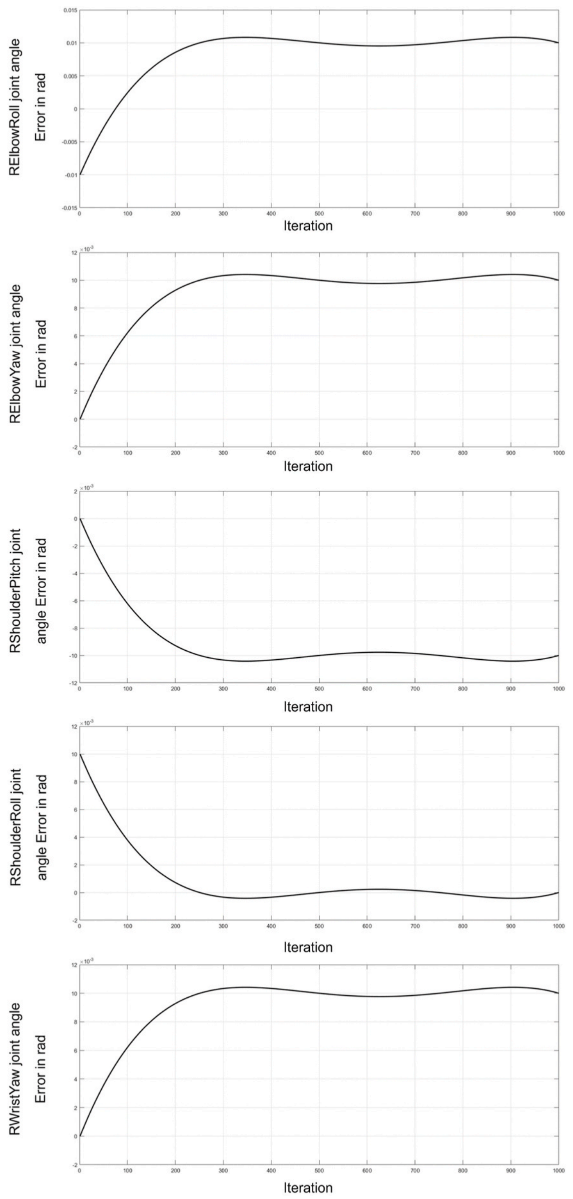


c. Head Joint Trajectories -Gazebo Sim vs Choregraphe

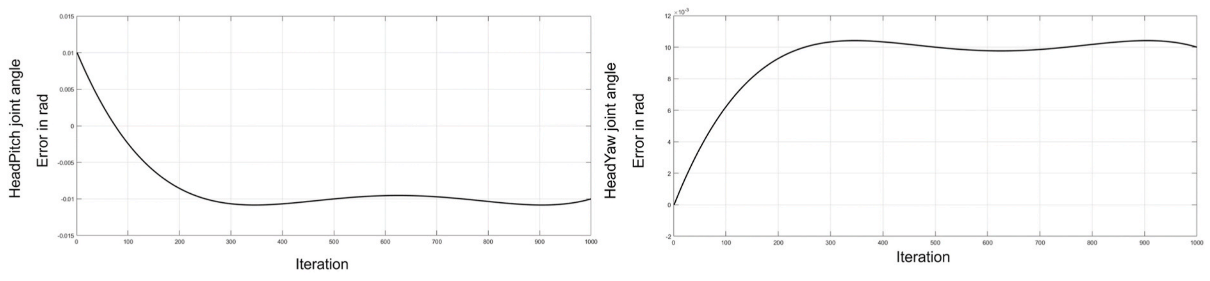
Fig. 6. Evaluating Fidelity of Simulated and Real-World Pepper - Gazebo vs. Choregraphe.



a. Left Arm Joint Trajectories error



b. Right Arm Joint Trajectories error



c. Head Joint Trajectories error

Fig. 7. Quantifying accuracy - error distribution of joint angles in gazebo simulations.

integration with various AI frameworks, it supports the potential development and testing of complex AI models. Its modular architecture enhances flexibility and adaptability, making it suitable for diverse AI research needs. It is important to state here that there is a current limitation with the kinematic structure of the hands of the robot within our simulation. The URDF in use within our model does handle all finger movements collectively with a single joint, controlled by mimic tags, a feature not supported by Gazebo Sim Fortress. This significantly limits the potential of our simulation to fully model the fine finger articulations needed for the performance of advanced motor skills, since it does not allow the independent control of each finger joint alone. This limitation affects tasks that require precise manipulations. However, it can still support broader tasks, like grasping or pushing with basic hand opening and closing. Despite this, this breakthrough unlocks a plethora of opportunities for the development and refinement of complex machine learning algorithms designed to equip robots with intricate skills, meticulous movement strategies, and the ability to seamlessly interact with the real world. By providing a safe and controlled virtual playground for robot learning, our digital twin paves the way for accelerated advancements in the field of humanoid robotics. Furthermore, the system's high-fidelity simulation capabilities suggest that AI models trained in this environment could potentially be transferred to real-world applications with minimal adjustments, reducing the sim-to-real gap and enhancing the efficiency of AI integration in robotic systems. We confidently stand by the contribution of this work to the broader robotics and AI communities, offering a robust and accessible platform that invites further research and collaboration to explore the exciting frontiers of robot capability and intelligence. This open-source platform signifies our commitment to fostering a collaborative environment where, collectively, we can empower robots to become not just tools, but valuable partners in shaping the future.

Funding

This research is funded by the Euromed University of Fes, UEMF, Morocco.

Data availability statement

The data associated with this study has been deposited in a publicly available repository. You can access the data at https://github.com/HibaSekkat/pepper_ign_moveit2.git.

CRediT authorship contribution statement

Hiba Sekkat: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Oumaima Moutik:** Conceptualization. **Badr El Kari:** Writing – review & editing, Validation, Supervision, Project administration, Conceptualization. **Yassine Chaibi:** Writing – review & editing. **Taha Ait Tchakoucht:** Data curation. **Ahmed El Hilali Alaoui:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors warmly thank M. Andrej Orsula for fruitful discussions and for his assistance with the software implementation.

References

[1] H. Choi, et al., On the use of simulation in robotics: opportunities, challenges, and suggestions for moving forward, Proc. Natl. Acad. Sci. USA 118 (1) (Jan. 2021) e1907856118, https://doi.org/10.1073/pnas.1907856118.
[2] S. Tselegkaridis, T. Sapounidis, Simulators in educational robotics: a review, Educ. Sci. 11 (1) (Jan. 2021) 11, https://doi.org/10.3390/educsci11010011.
[3] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, J. Peters, Robot learning from randomized simulations: a review, Front. Robot. 9 (Apr. 2022) 799893, https://doi.org/10.3389/frobt.2022.799893. AI.
[4] G. Suddrey, A. Jacobson, B. Ward, Enabling a pepper robot to provide automated and interactive Tours of a robotics Laboratory, in: *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2018)*, Australian Robotics and Automation Association (ARAA), 2018.
[5] T. Zhang, H. Mo, Reinforcement learning for robot research: a comprehensive review and open issues, Int. J. Adv. Rob. Syst. 18 (3) (2021) 17298814211007305.
[6] A. Ştefania Ghiţă, et al., The amiro social robotics framework: deployment and evaluation on the pepper robot, Sensors 20 (24) (2020) 7271.
[7] Z.A. barakeh, S. alkork, A.S. Karar, S. Said, T. Beyrouthy, Pepper humanoid robot as a service robot: a customer approach, in: 2019 3rd International Conference on Bio-Engineering for Smart Technologies (BioSMART), IEEE, Apr. 2019, https://doi.org/10.1109/biosmart.2019.8734250.
[8] M. Kyrarini, et al., A Survey of robots in healthcare, Technologies 9 (1) (Jan. 2021) 8, https://doi.org/10.3390/technologies9010008.
[9] J. Guggemos, S. Seufert, S. Sonderegger, Humanoid robots in higher education: evaluating the acceptance of Pepper in the context of an academic writing course using the UTAUT, Br. J. Educ. Technol. 51 (5) (2020) 1864–1883.
[10] K. Pollmann, C. Ruff, K. Vetter, G. Zimmermann, Robot vs. voice assistant: is playing with pepper more fun than playing with alexa?, in: Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction, 2020, pp. 395–397.
[11] A.K. Pandey, R. Gelin, A mass-Produced sociable humanoid robot: pepper: the first machine of its kind, IEEE Robot. Autom. Mag. 25 (3) (Sep. 2018) 40–48, https://doi.org/10.1109/MRA.2018.2833157.

[12] A. Gardecki, M. Podpora, R. Beniak, B. Klin, The pepper humanoid robot in Front Desk application, in: *2018 Progress in Applied Electrical Engineering (PAEE)*, Koscielisko, IEEE, Jun. 2018, pp. 1–7, https://doi.org/10.1109/PAEE.2018.8441069.
[13] A. Costa, E. Martinez-Martin, M. Cazorla, V. Julian, PHAROS—PHysical assistant RObot system, Sensors 18 (8) (Aug. 2018) 2633, https://doi.org/10.3390/s18082633.
[14] F. Lier, S. Wachsmuth, Towards an open simulation environment for the pepper robot, in: Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction, ACM, Mar. 2018, https://doi.org/10.1145/3173386.3177088 in HRI '18.
[15] K. Nutonen, V. Kuts, T. Otto, Industrial robot training in the simulation using the machine learning agent, Procedia Comput. Sci. 217 (2023) 446–455.
[16] T. Alhmiedat, et al., A SLAM-based localization and navigation system for social robots: the pepper robot case, Machines 11 (2) (2023) 158.
[17] C. Symeonidis, N. Nikolaidis, Simulation environments, in: Deep Learning for Robot Perception and Cognition, Elsevier, 2022, pp. 461–490.
[18] L. Cascone, M. Nappi, F. Narducci, I. Passero, DTPAAL: digital twinning pepper and ambient assisted living, IEEE Trans. Ind. Inf. 18 (2) (2021) 1397–1404.
[19] S. Balakirsky, Z. Kootbally, USARSim/ROS: A Combined Framework for Robotic Control and Simulation, 2012, pp. 101–108, https://doi.org/10.1115/ISFA2012-7179.
[20] T. Prasanth, Implementation of ROS-I on industrial robot simulation environment, Int. J. Adv. Res. Ideas Innov. Technol. 5 (2019) 1049–1055.
[21] A. Henschel, G. Laban, E.S. Cross, What makes a robot social? A review of social robots from science Fiction to a Home or Hospital near You, Curr. Robot. Rep. 2 (1) (Mar. 2021) 9–19, https://doi.org/10.1007/s43154-020-00035-0.
[22] S. Góngora Alonso, S. Hamrioui, I. de la Torre Díez, E. Motta Cruz, M. López-Coronado, M. Franco, Social robots for people with aging and Dementia: a systematic review of literature, Telemed. E-Health 25 (7) (Jul. 2019) 533–540, https://doi.org/10.1089/tmj.2018.0051.
[23] T. Belpaeme, J. Kennedy, A. Ramachandran, B. Scassellati, F. Tanaka, Social robots for education: a review, Sci. Robot. 3 (21) (Aug. 2018) eaat5954, https://doi.org/10.1126/scirobotics.aat5954.
[24] M.E. Foster, et al., The MuMMER project: engaging human-robot interaction in real-world public spaces, in: A. Agah, J.-J. Cabibihan, A.M. Howard, M. A. Salichs, H. He (Eds.), Social Robotics, Lecture Notes in Computer Science, vol. 9979, Springer International Publishing, Cham, 2016, pp. 753–763, https://doi.org/10.1007/978-3-319-47437-3_74, 9979.
[25] M. Niemelä, P. Heikkilä, H. Lammi, V. Oksman, Shopping mall robots–opportunities and constraints from the retailer and manager perspective, in: Social Robotics: 9th International Conference, ICSR 2017, Tsukuba, Japan, November 22-24, 2017, Proceedings 9, Springer, 2017, pp. 485–494.
[26] A. Gardecki, M. Podpora, Experience from the operation of the Pepper humanoid robots, in: *2017 Progress in Applied Electrical Engineering (PAEE)*, Koscielisko, Poland, IEEE, Jun. 2017, pp. 1–6, https://doi.org/10.1109/PAEE.2017.8008994.
[27] B. Acosta, W. Yang, M. Posa, Validating robotics simulators on real-world impacts, IEEE Rob. Autom. Lett. 7 (3) (2022) 6471–6478.
[28] D. Fernandez-Chaves, J.-R. Ruiz-Sarmiento, A. Jaenal, N. Petkov, J. Gonzalez-Jimenez, Robot@VirtualHome, an ecosystem of virtual environments and tools for realistic indoor robotic simulation, Expert Syst. Appl. 208 (Dec. 2022) 117970, https://doi.org/10.1016/j.eswa.2022.117970.
[29] J. Lima, R.B. Kalbermatter, J. Braun, T. Brito, G. Berger, P. Costa, A realistic simulation environment as a teaching aid in educational robotics, in: 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE), IEEE, 2022, pp. 430–435.
[30] L. Pitonakova, M. Giuliani, A. Pipe, A. Winfield, Feature and Performance Comparison of the V-REP, Gazebo and ARGoS Robot Simulators, 2018, pp. 357–368, https://doi.org/10.1007/978-3-319-96728-8_30.
[31] A. Ayala, F. Cruz, D. Campos, R. Rubio, B. Fernandes, R. Dazeley, A comparison of humanoid robot simulators: a quantitative approach, in: 2020 Joint IEEE 10th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), IEEE, Oct. 2020, https://doi.org/10.1109/icdl-epirob48136.2020.9278116.
[32] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan: IEEE, 2004, pp. 2149–2154, https://doi.org/10.1109/IROS.2004.1389727.
[33] M.S.P. de Melo, J.G. da S. Neto, P.J.L. Silva, J.M. Teixeira, V. Teichrieb, Analysis and comparison of robotics 3D simulators, in: 2019 21st Symp. Virtual Augment. Real, SVR, 2019, pp. 242–251, https://doi.org/10.1109/SVR.2019.00049.
[34] D. Ferigo, S. Traversaro, G. Metta, D. Pucci, Gym-ignition: reproducible robotic simulations for reinforcement learning, in: 2020 IEEE/SICE International Symposium on System Integration (SII), Jan. 2020, pp. 885–890, https://doi.org/10.1109/SII46433.2020.9025951.
[35] R. Lange, S. Traversaro, O. Lenord, C. Bertsch, Integrating the functional Mock-up interface with ROS and gazebo, Robot Oper. Syst. ROS Complete Ref. 5 (2021) 187–231.
[36] M. Marian, F. Stîngă, M.-T. Georgescu, H. Roibu, D. Popescu, F. Manta, A ROS-based control application for a robotic platform using the gazebo 3D simulator, in: 2020 21th Int. Carpathian Control Conf. ICCC, 2020, pp. 1–5, https://doi.org/10.1109/ICCC49264.2020.9257256.
[37] C.B. Kristensen, F. Sørensen, H. Nielsen, M. Andersen, S.P. Bendtsen, S. Bøgh, Towards a robot simulation framework for E-waste disassembly using reinforcement learning, Procedia Manuf. (2019), https://doi.org/10.1016/J.PROMFG.2020.01.030.
[38] M. Martini, A. Eirale, S. Cerrato, M. Chiaberge, PIC4rl-gym: a ROS2 modular framework for robots autonomous navigation with deep reinforcement learning, in: 2023 3rd Int. Conf. Comput. Control Robot. ICCCR, 2022, pp. 198–202, https://doi.org/10.1109/ICCCR56747.2023.10193996.
[39] C. Camargo, J. Gonçalves, M.Á. Conde, F.J. Rodríguez-Sedano, P. Costa, F.J. García-Peñalvo, Systematic literature review of realistic simulators applied in educational robotics context, Sensors 21 (12) (2021) 4031.
[40] J. Collins, S. Chand, A. Vanderkop, D. Howard, A review of physics simulators for robotic applications, IEEE Access 9 (2021) 51416–51431.
[41] S. Silva, N. Verdezoto, D. Paillacho, S. Millan-Norman, J.D. Hernández, Online social robot navigation in indoor, large and crowded environments, in: 2023 IEEE International Conference on Robotics and Automation (ICRA), IEEE, May 2023, https://doi.org/10.1109/icra48891.2023.10160603.
[42] E. Ganal, L. Siol, B. Lugrin, PePUT: a unity toolkit for the social robot pepper, in: 2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), IEEE, 2023, pp. 1012–1019.
[43] L. Li, M. Neau, T. Ung, C. Buche, Crossing real and virtual: pepper robot as an interactive digital twin, in: RoboCup 2023: Robot World Cup XXVI, Springer-Verlag, Berlin, Heidelberg, 2024, pp. 275–286, https://doi.org/10.1007/978-3-031-55015-7_23.
[44] L. Cobo Hurtado, P.F. Viñas, E. Zalama, J. Gómez-García-Bermejo, J.M. Delgado, B. Vielba García, Development and usability validation of a social robot platform for physical and cognitive stimulation in elder care facilities, in: Healthcare, MDPI, 2021, p. 1067.
[45] Y.H. Jo, S.Y. Cho, B.W. Choi, Towards a ROS2-based software architecture for service robots, Bull. Electr. Eng. Inform. 12 (5) (2023) 3027–3038.
[46] H. Sekkat, O. Moutik, L. Ourabah, B. ElKari, Y. Chaibi, T.A. Tchakoucht, Review of reinforcement learning for robotic grasping: analysis and recommendations, Stat. Optim. Inf. Comput. 12 (2) (2024) 571–601.
[47] D. St-Onge, D. Herath, The robot operating system (ROS1 &2): programming paradigms and deployment, in: Foundations of Robotics: A Multidisciplinary Approach with Python and ROS, Springer, 2022, pp. 105–126.
[48] A. Bolotnikova, P. Gergondet, A. Tanguy, S. Courtois, A. Kheddar, Task-space control interface for SoftBank humanoid robots and its human-robot interaction applications, in: 2021 IEEESICE Int. Symp. Syst. Integr. SII, 2020, pp. 560–565, https://doi.org/10.1109/IEEECONF49454.2021.9382685.
[49] M. Askarpour, M. Rossi, O. Tiryakiler, Co-simulation of human-robot collaboration: from temporal logic to 3D simulation, Electron. Proc. Theor. Comput. Sci. 319 (Jul. 2020) 1–8, https://doi.org/10.4204/eptcs.319.1.
[50] N. Pérez-Higueras, R. Otero, F. Caballero, L. Merino, HuNavSim: a ROS 2 human navigation simulator for benchmarking human-aware robot navigation, IEEE Rob. Autom. Lett. 8 (11) (Nov. 2023) 7130–7137, https://doi.org/10.1109/lra.2023.3316072.
[51] R. Groot and others, Autonomous Exploration and Navigation with the Pepper Robot, Master`s Thesis, 2018.
[52] F. Leiva, K. Lobos-Tsunekawa, J. Ruiz-del-Solar, Collision avoidance for indoor service robots through multimodal deep reinforcement learning, in: RoboCup 2019: Robot World Cup XXIII 23, Springer, 2019, pp. 140–153.
[53] M. Everett, Y.F. Chen, J. How, Collision avoidance in pedestrian-rich environments with deep reinforcement learning, IEEE Access 9 (2019) 10357–10377, https://doi.org/10.1109/ACCESS.2021.3050338.

[54] F.P. Audonnet, A. Hamilton, G. Aragon-Camarasa, A systematic comparison of simulation software for robotic arm manipulation using ROS2, in: 2022 22nd International Conference on Control, Automation and Systems (ICCAS), IEEE, 2022, pp. 755–762.
[55] X. Zhang, D.K. Lin, L. Wang, Digital triplet: a sequential methodology for digital twin learning, Mathematics 11 (12) (2023) 2661.
[56] A. Orsula, S. Bøgh, M. Olivares-Mendez, C. Martinez, Learning to grasp on the moon from 3D octree observations with deep reinforcement learning, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2022, pp. 4112–4119.
[57] S. Das, S.K. Mishra, A machine learning approach for collision avoidance and path planning of mobile robot under dense and cluttered environments, Comput. Electr. Eng. 103 (2022) 108376.