Configuring CAD models for digital twinning in Nvidia Isaac Sim

Alexandru Nicolae Serban
Student
Fontys University of Applied Sciences
Eindhoven, Netherlands

Abstract Digital twins are gaining popularity in connecting the physical and digital worlds, especially within simulation platforms such as NVIDIA Isaac Sim. This research investigates best practices for importing and configuring CAD models to create accurate and functional digital twins. Key focus areas include optimizing the CAD model for simulation, implementing efficient workflows for material and texture mapping, and ensuring compatibility with physics-based simulation requirements. The study provides a framework for engineers, encompassing techniques for processing CAD data, integrating metadata, and making use of Isaac Sim-specific tools. By adopting these practices, developers can enhance simulation accuracy, reduce setup time, and streamline the deployment of digital twins in robotics, manufacturing, and autonomous systems applications. The findings contribute to advancing methodologies in virtual prototyping and real-time simulation.

Index Terms Simulation, digital twin, replica, Isaac Sim, Omniverse, CAD model, simulator, robotics, Onshape.

I. Introduction

As the use of reinforcement learning (RL) and Artificial Intelligence (A.I.) in robotics is growing exponentially every year, engineers have to take into account how to test and train the systems to ensure a safe and stable operation without harming the hardware and the humans in the close proximity of the robots. A physics simulator is currently the best solution for training robots and mechatronics systems in the digital world by using a physics engine to simulate environments from real world scenarios. One of these simulators is Nvidia Isaac Sim. Isaac Sim is a simulation toolkit built on Nvidia's Omniverse platform. It enables photorealistic simulation and synthetic data generation for AI-enabled robotics, with applications in navigation, manipulation, and reinforcement learning (RL). It supports physics-based simulations using PhysX and RTX for ray tracing [1]. It supports applications in sim-to-real transfer and industrial robotics, such as creating digital twins of factories and training robots for collision-free operation [2]. This research paper focuses on proper importation and configuration of a CAD model in the Isaac Sim environment using tools specific to the software itself as well as ones from external sources. Studying these aspects helps the Isaac Sim community to understand how to prepare their twins for simulation while avoiding problems that could occur in the software when the models are configured.

II. Related Work and Challenges

Related Work

Nvidia Omniverse Isaac Sim is a robotic simulation tool launched in 2020 which aims to simplify the entire pipeline for developing robotic simulations. It aims to capitalize on the RTX GPU's computing capability for simulations and rendering. Nvidia Omniverse Isaac Sim uses the latest version of PhysX, and has the full suite of Nvidia rendering tools, and access to other rendering tools as well [3]. Since its launch, more and more projects started to take shape in Isaac Sim, at which robotics engineers and tech passionates contributed. Below, some interesting projects are presented which were made possible by correct implementation and optimization of the digital twins.

1) TIAGo Mobile Manipulator: A versatile robotic platform featuring a mobile base and a robotic arm. Modifications were made to its simulation model to account for self-collisions and mesh simplifications [2].

2) UR3 and Baxter: These robots were integrated into the TELESIM framework, a modular teleoperation system leveraging digital twins. The study emphasized ease of use and modularity, allowing non-experts to perform tasks like cube stacking using VR controllers or finger mapping technologies [4].

3) Multiarm Robotic System: This is a novel research platform designed for scientific exploration, featuring four robotic arms on a circular rail for enhanced mobility and dexterity. The system's design integrates collision avoidance and self-awareness mechanisms, with applications in delicate tasks requiring high precision [5].

4) Autonomous Robotic Manipulators (ARMs): Manipulator Robot Testing and Enhancement in Simulation (MARTENS) is a framework designed to test and improve robots that use deep learning for tasks like picking and placing objects. The framework finds potential failures in the robot's vision and control systems, improves the deep learning models, and helps uncover design flaws. By using simulation, MARTENS makes it easier to train robots without needing a lot of real-world data, achieving high accuracy with less effort [6].

These digital twins can be seen in Figure 1.

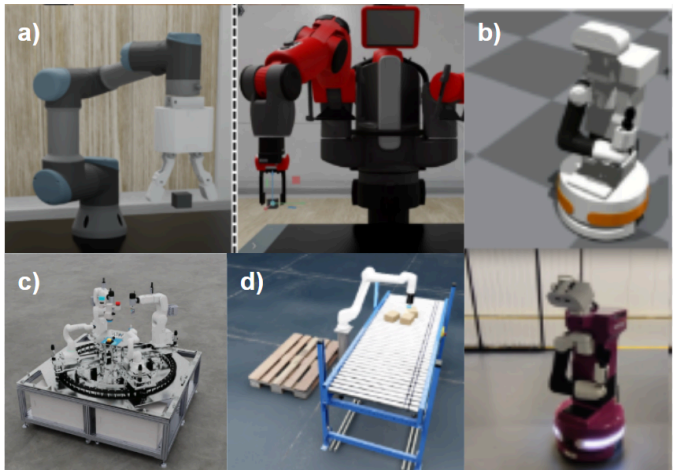

Figure 1. a) UR3 and Baxter robots [4], b) Tiago robot trained using Sim2Real reinforcement learning branch [2], c) Multiarm robotic platform for cranial surgeries [5], d) Pick and Place robot simulation [6]

Challenges

Using Isaac Sim allows users to develop and improve software and simulation for complex robotic systems, which can completely change the robotics industry in the near future. However there are still some challenges and issues that limit the development of digital twins in Isaac Sim.

1) Vendor Lock In: Isaac Sim does have rigid body dynamics, and it also supports the Omniverse connect system for external plugins. It integrates with other industry standard tools (ROS/ROS2, Maya (Autodesk Inc., 2022), SOLIDWORKS (Dassault Systemes SolidWorks Corporation, 2022), Unreal 4 (Epic Games, Inc, 2022), etc) through the Omniverse `Nucleus. While the inter-connectivity of services such as this is very attractive for collaborative purposes, it does introduce the issue of 'vendor lock in', where the user is committed to a specific software solution, as switching from the product is impractical [3].

2) Delay: One thing that is a big issue while controlling the digital twin and real robot is the delay. It can occur in any simulator and it has a high impact on the performance of the system. One example of this problem is the delay in the robots of the TELESIM project. The framework introduces a slight amount of lag (500 ms) between the user movement and the robot movement, partly due to the path planning step, the time taken by the actuator to move the robot arm to the desired position and mostly due to safety restrictions that caps the maximum speed of our robots. The faster the user moves from one position to another, the higher the delay as the arm tries to catch up [4].

3) Performance: Isaac Sim being a realistic physics and texture rendering simulator, the system requirements for running such a software are pretty high. The minimum requirements are already connected to a high end machine, which can be hard to obtain by individuals who want to create a digital twin. The requirements are described in Figure 2.

Element	Minimum Spec	Good	Ideal
OS	Ubuntu 20.04/22.04 Windows 10/11	Ubuntu 20.04/22.04 Windows 10/11	Ubuntu 20.04/22.04 Windows 10/11
CPU	Intel Core i7 (7th Generation) AMD Ryzen 5	Intel Core i7 (9th Genera-tion) AMD Ryzen 7	Intel Core i9, X-series or higher AMD Ryzen 9, Threadripper or higher
Cores	4	8	16
RAM	32GB	64GB	64GB
Storage	50GB SSD	500GB SSD	1TB NVMe SSD
GPU	GeForce RTX 2070	GeForce RTX 3080	RTX A6000
VRAM	8GB	10GB	48GB

Figure 2. Requirements to run Isaac Sim [1]

4) Importing complex systems: Small robots are straightforward to define and import into the Isaac Sim environment. Traditionally, the files used to describe a robot in terms of structure, joints, physics, meshes and colliders, called Unified Robotics Description Format (URDF) are created manually [7]. This can be a major problem when describing complex robots with multiple moving joints and detailed meshes, as it is time consuming and requires multiple iterations until the desired configuration is achieved.

III. Methodology

In this research, the Flexible Automated Future Factory (FLUFFY) system (see Figure 3), located at the Brainport Industries Campus in Eindhoven, serves as the foundation for creating a complex digital twin. The focus is placed on optimizing performance and refining the CAD model to ensure a smooth simulation. A Standard for the Exchange of Product Data (STEP) file is available for the entire system, encompassing all parts, textures, and materials. This file is utilized to configure the CAD model and import it into Isaac Sim through Onshape. Onshape, an online CAD software, enables collaborative work within a single document, making it particularly valuable for team-based projects. Moreover, Onshape integrates seamlessly with Isaac Sim, allowing direct import of CAD assemblies into an Isaac Sim environment. Dynamic constraints defined within the assembly are automatically converted into movable joints in Isaac Sim.

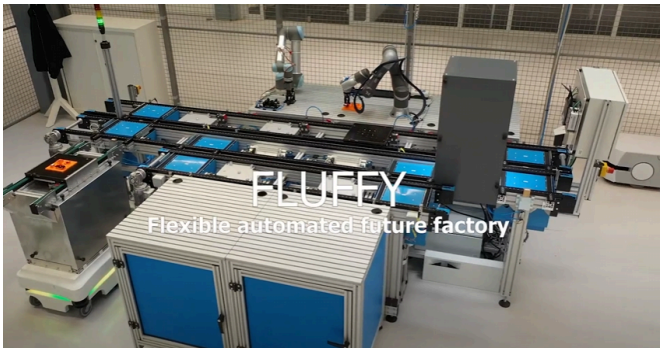

Figure 3. Flexible Automated Future Factory (FLUFFY) [8]

To perform the simulations, the following rig was used:

- **GPU:** Nvidia Geforce RTX 4060 TI, 8GB VRAM
- **CPU:** 11th Gen Intel(R) Core(TM) i7-11700KF @ 3.60GHz 3.60 GHz
- **RAM:** 24 GB
- **Storage:** 1TB HDD
- **Operating System:** Windows 11 Pro 64-bit

The compatibility checker tool in Isaac Sim was used to ensure that the system could run simulations. As shown in Figure 4, the system is capable of running Isaac Sim for simple simulations; however, performance declines as the number of components moving simultaneously increases. The primary issue lies with the system's RAM, as the recommended capacity is 32GB. For more complex simulations, additional RAM would be necessary to maintain performance.

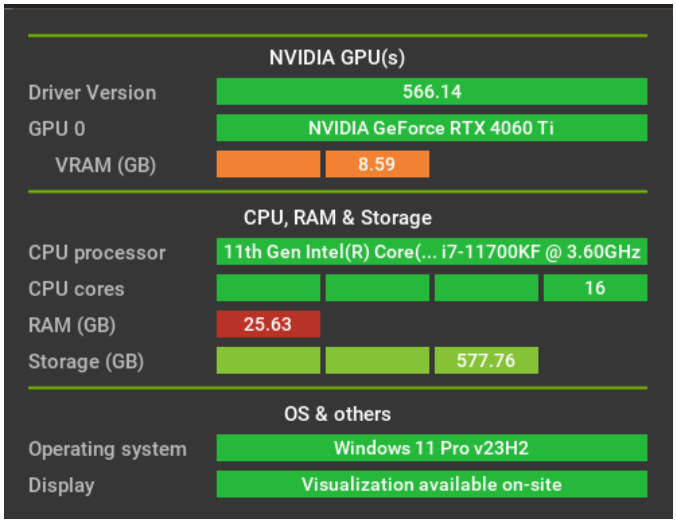

Figure 4. Compatibility Checker Isaac Sim

IV. CAD Model Configuration

Importing the STEP file

Upon opening the STEP file in SolidWorks, it was observed that it only contains surfaces and solid bodies. Since the file was exported from another CAD software, these components cannot be directly modified or mated together, which is necessary for making the system compatible with Isaac Sim. To address this issue, Onshape was utilized. This software can efficiently read, interpret, and divide STEP files into distinct parts and assemblies, enabling seamless mating and modification. After importing the STEP file, Onshape created a document containing all the individual parts and assemblies.

Simplifying the assembly

In the created assemblies, there are multiple parts which are not important to the digital twin implementation, components that are not seen from the outside, duplicate bodies created when the STEP file was exported as well as all the nuts and bolts. An example of these parts can be seen in Figure 5.

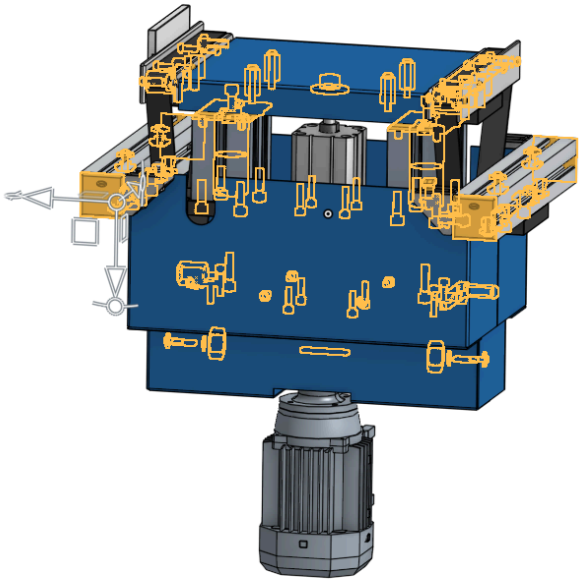

Figure 5. Parts in a FLUFFY sub-assembly that are not required for simulation (highlighted in orange)

There are thousands of parts like these in the overall assembly which negatively impact the performance of the simulation, as each individual one has to be processed and loaded by Isaac Sim in each simulation step. To increase the performance, these can be safely removed from the CAD model. In the original model there are 6257 parts. After model simplification, only 1355 parts are left.

Mates

The assembly and its sub-assemblies consist of individual parts that are completely unconnected. As a result, if the digital twin is imported into Isaac Sim in this state, all components will collapse to the ground when the simulation begins, since no relationships link them together. The Omniverse Onshape Importer documentation page [9] was followed to ensure a proper setup of the CAD model for simulation. To ensure proper physics import, all sub-assemblies without internal moving parts should be

grouped to treat them as single rigid bodies, simplifying mating and USD generation. Sub-assemblies containing internal joints must be fastened to external components using a fastened mate. Mates should follow the assembly hierarchy, ensuring that sub-assemblies are correctly ordered in relation to the main body. Additionally, the assembly must be set to a zero pose to prevent unintended offsets, as the initial position will be treated as the default for all joint limits.

The FLUFFY system's lanes contain multiple identical sub-assemblies. To maintain modularity, only one instance of each sub-assembly was mated using the approach described above and then reused across the system. Additionally, each lane was mated separately and later combined. To ensure the entire assembly remains intact when the simulation starts in Isaac Sim, the lanes were mated as follows: each lane had its own assembly, with the frame mated to the assembly's origin and all other parts and sub-assemblies mated to the frame, as shown in Figure 6.

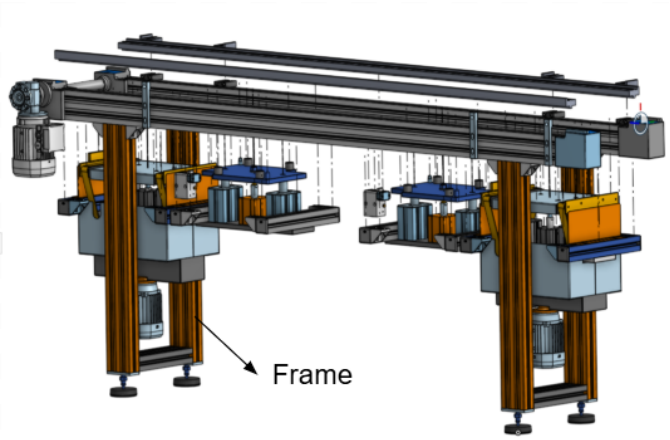

Figure 6. Frame assembly

For the dynamic parts like the pneumatic cylinders, the following method of mating was used: the assemblies were split into two parts: the fixed sub-assembly, which is attached to the frame of the lane, and the moving sub-assembly, which is supposed to move when the pneumatic piston is actuated. The parts of both sub-assemblies are mated together using fixed mates, and a slider mate is used to connect the two sub-assemblies together, see Figure 7. The slider mate is converted into a Prismatic Joint when the model is imported into Isaac Sim.

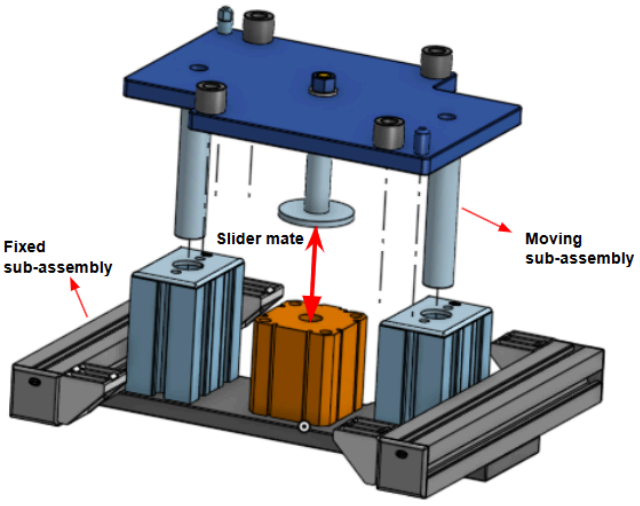

Figure 7. Piston sub-assembly

V. Importing and Configuration in Isaac Sim

Importing into Isaac Sim

After the model was ready, the "Import from Onshape" tool in Issac Sim was used to get the digital twin into the environment. To access the Onshape Library in Isaac Sim an API key has to be generated according to the "Onshape Importer" tutorial [9]. While importing the model, a material can be assigned to each part individually if the material wasn't specified in Onshape. For simplicity, all the parts were set to aluminium. In Figure 8, the Isaac Sim model is displayed alongside the Onshape model.

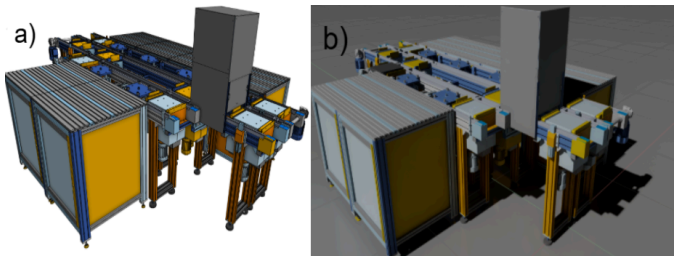

Figure 8. a) Onshape model, b) Isaac Sim model

Configuring the digital twin

Although the model is now loaded into the Isaac Sim world, it is not yet ready for simulation. If the simulation were to start at this stage, the system would fall indefinitely. To prevent this, a ground plane and a Physics Scene must be added to the world, and the model should be positioned so that its legs rest on the plane. Additionally, each lane requires a fixed joint on one of its legs to keep it stationary during the simulation.

With the system securely fixed to the simulated ground, the joints corresponding to the pneumatic cylinders can be configured by incorporating linear drives. These drives enable the joints to function as linear motors, which can be fine-tuned in terms of stiffness and damping according to Nvidia's tutorial [10]. By setting a higher damping constant than stiffness, the pistons achieve more natural movement, smoothly decelerating as they approach their target positions.

To simulate the system's conveyors, a linear velocity can be applied to their solid bodies. This process is simplified using the conveyor belt plugin, available as an Isaac Sim extension. After defining collider presets for the conveyors' solid bodies ensuring proper interaction between the trays and the conveyors the conveyor plugins can be created for each conveyor. The extension allows easy control over the direction, velocity, and animations of the conveyors within the simulation.

Improving the simulation performance

During the simulation run after configuring the model, the average frame rate dropped to as low as 5 FPS, causing NVIDIA Isaac Sim to become unresponsive and leading to overheating of the system components. The issue was traced back to the Onshape importer tool, which automatically applied a collider preset to all parts of the system. This significantly increased the GPU's processing load. In NVIDIA Isaac Sim, colliders are physics components that define an object's physical boundaries for collision detection and interaction within the Omniverse PhysX engine. They come in different forms, including simple primitives (box, sphere, capsule), convex hulls for optimized shapes, and triangle mesh colliders for high-precision interactions [11]. The physics engine handles collisions by detecting intersections, applying forces, and resolving contact responses based on parameters such as mass, friction, and restitution.

To solve this problem, all the colliders created by the importer were manually deleted and recreated only for the parts that actively interact with the trays and the environment (piston assemblies, frame legs, belts and conveyors). In Figure 9, the colliders of FLUFFY can be checked. After removing the colliders and running the simulation, the FPS increased from 5 to 30 and the software didn't crash anymore.

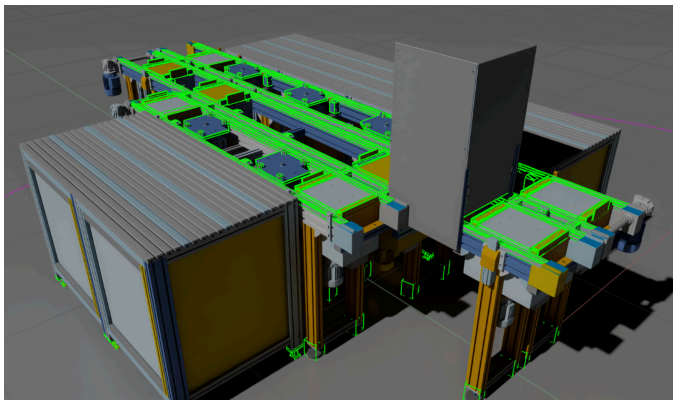

Figure 9. Colliders of the system displayed in green

Nvidia Isaac Sim is compatible with the Deep Learning Super-Sampling technology (DLSS). DLSS is a cutting-edge deep learning-based technique that improves video game graphics by upscaling lower-resolution images to higher resolutions (e.g., from 1080p to 4K) while reducing the computational load on GPUs. By using Convolutional Neural Networks (CNNs), DLSS processes low-resolution frames along with their motion vectors to upscale them in real time, providing players with higher frame rates and better visual quality without demanding as much GPU power [12]. Activating this feature doubled the amount of FPS during simulation and while editing the model inside the Isaac Sim world. In Figure 10, a comparison between the performance of each DLSS mode can be observed. The DLSS Performance mode was chosen for further simulations, as it achieves the highest amount of FPS.

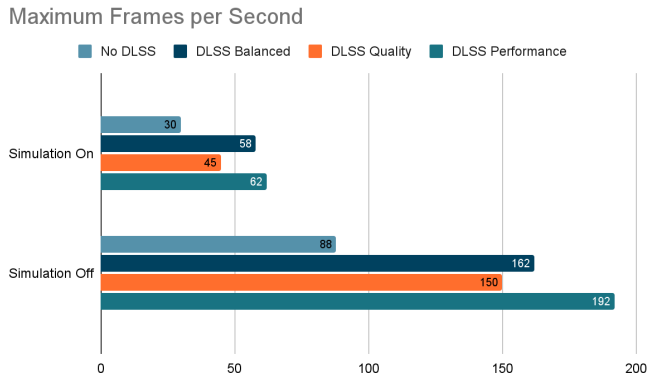

Figure 10. FPS comparison between DLSS modes while simulating and editing FLUFFY in Isaac Sim

VI. Results

The research highlighted several key improvements in performance, model importation, and simulation efficiency when utilizing Isaac Sim, particularly with the integration of optimized CAD models and advanced technologies like DLSS.

Model Import and Configuration Process

Successful Import of CAD Models: The research confirmed that importing STEP files from Onshape into Isaac Sim was seamless and straightforward. Onshape's cloud-based platform allowed for easy export, while Isaac Sim automatically converted dynamic constraints into movable joints, simplifying the model setup process.

Efficient Joint and Assembly Configuration: A key result was the process of configuring joints and mates. Simplifying assemblies and using appropriate mates ensured that imported models retained proper configurations, preventing collapse when simulations began. This resulted in a more stable and predictable behavior within the simulation environment.

Simulation Efficiency and Workflow

Streamlined Workflow: The research demonstrated that using Onshape for CAD modeling in combination with Isaac Sim for simulation significantly accelerated the development of digital twins. Onshape's cloud-based, collaborative features reduced the time spent on iterative design processes and troubleshooting, enabling engineers to focus more on refining simulation parameters rather than correcting design errors.

Reduced Setup Time: By adhering to best practices, such as simplifying parts, ensuring proper assembly structure, and utilizing Onshape's importer tool, the setup time for the digital twin was drastically reduced. This is especially advantageous in industrial applications where fast testing and deployment are critical. The research confirmed that this approach led to quicker, more efficient simulation setups, which is crucial for real-world applications where rapid iteration is essential.

Performance Enhancements

Optimized CAD Model for Better Performance: Reducing the complexity of the CAD model by eliminating unnecessary components significantly improved simulation performance. The model was reduced from 6257 parts to 1355 parts, which resulted in a substantial decrease in the computational load. By removing the colliders, the system became more stable and avoided issues like crashes due to overheating.

DLSS Integration: The implementation of NVIDIA's Deep Learning Super Sampling (DLSS) technology further enhanced simulation performance. DLSS improved FPS by nearly doubling it during both simulation and editing stages, resulting in a smoother and more responsive user experience. The performance mode of DLSS effectively balanced visual fidelity with computational efficiency, especially in simulations with many moving parts or complex interactions.

VII. Conclusion

In conclusion, this research provides valuable steps and best practices for importing and setting up CAD models in NVIDIA Isaac Sim to create accurate and effective digital twins. By focusing on optimizing CAD models, simplifying complex parts, and properly configuring joints, this study offers a guide for getting digital twins ready for simulation. It also addresses key challenges like making sure the model works well with Isaac Sim's physics engine, improving performance, and using helpful tools like Onshape and the conveyor belt plugin. The findings help make simulations more realistic and efficient, improving the testing and training of robots and digital twins in various fields. Overall, this work makes it easier for developers and engineers to create digital twins, pushing forward the development of virtual testing and real-time simulations in robotics and other industries

References

[1] D. Katainen, “NVIDIA ISAAC AND ROBOT OPERATING SYSTEM 2”.

[2] J. Albardaner, A. S. Miguel, N. García, and M. Dalmau-Moreno, “Sim-to-Real gap in RL: Use Case with TIAGo and Isaac Sim/Gym,” Mar. 27, 2024, arXiv: arXiv:2403.07091. doi: 10.48550/arXiv.2403.07091.

[3] I. Caliskanelli, “Digital Mock-Ups for Nuclear Decommissioning: A Survey on Existing Simulation Tools for Industry Applications,” Robot. Autom. Eng. J., vol. 5, no. 4, Nov. 2023, doi: 10.19080/RAEJ.2023.05.555669.

[4] F. P. Audonnet, J. Grizou, A. Hamilton, and G. Aragon-Camarasa, “TELESIM: A Modular and Plug-and-Play Framework for Robotic Arm Teleoperation using a Digital Twin,” Sep. 20, 2023, arXiv: arXiv:2309.10579. doi: 10.48550/arXiv.2309.10579.

[5] M. M. Marinho, J. J. Quiroz-Omaña, and K. Harada, “A Multiarm Robotic Platform for Scientific Exploration: Its Design, Digital Twins, and Validation,” IEEE Robot. Autom. Mag., pp. 2–12, 2024, doi: 10.1109/MRA.2023.3336472.

[6] D. Humeniuk, H. B. Braiek, T. Reid, and F. Khomh, “In-Simulation Testing of Deep Learning Vision Models in Autonomous Robotic Manipulators,” in Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, Oct. 2024, pp. 2187–2198. doi: 10.1145/3691620.3695281.

[7] “ROS Documentation,” XML Robot Description Format (URDF). [Online]. Available: https://wiki.ros.org/urdf/XML/model

[8] M. Marnix, Project FLUFFY - Fontys adaptive robotics minor, (Jul. 09, 2021). [Online Video]. Available: https://www.youtube.com/watch?v=SnsWOUc00J0

[9] “Onshape importer,” Omniverse Extensions. [Online]. Available: https://docs.omniverse.nvidia.com/extensions/latest/ext_onshape.html

[10] “Tuning Joint Drive Gains.” [Online]. Available: https://docs.omniverse.nvidia.com/isaacsim/latest/advanced_tutorials/tutorial_advanced_joint_tuning.html#drive-modes

[11] “Physics.” [Online]. Available: https://docs.omniverse.nvidia.com/isaacsim/latest/simulation_fundamentals.html#physics

[12] A. Watson, “Deep Learning Techniques for Super-Resolution in Video Games”.