Towards Automated Physics-Based Modeling: Fusion of Construction Equipment Data for Efficient Simulation

Liqun Xu[1], Dharmaraj Veeramani[2], and

Zhenhua Zhu[3]

[1,3]Department of Civil and Environmental Engineering, University of Wisconsin-Madison, USA
[2] Department of Industrial and Systems Engineering, University of Wisconsin-Madison, USA
liqun.xu@wisc.edu, raj.veeramani@wisc.edu, zzhu286@wisc.edu

Abstract

Physics-based simulations play a crucial role in the design and development of autonomous construction equipment. Currently, a key challenge in these simulations is the time-intensive task of preparing construction equipment models that accurately represent both the equipment's geometry and physics. Manual model creation for simulations becomes particularly laborious due to the integration of diverse mechanical data such as materials, joints, and drives with the geometric data. Extant methods for automatic physics-based modeling of standard modular robots are inadequate for addressing the complexities of construction equipment. Therefore, this paper investigates the feasibility of automating and streamlining the physics-based modeling process by fusing the construction equipment's mechanical data into its 3D computer-aided design (CAD) model. The proposed method involves converting the construction equipment 3D CAD model into a universal scene description (USD) model for efficient data fusion. Subsequently, the method automatically configures material parameters, collision meshes, establishes component relations, and incorporates joints and drives for the USD model. To validate the efficacy of this approach, the proposed method is applied to create a physics-based model of a Caterpillar 390F LME excavator, and simulated in a scalable robotic simulator (NVIDIA Isaac Sim). The findings demonstrate that the proposed method significantly reduces the time required for physics-based modeling compared to traditional manual methods.

Keywords –

Construction equipment modeling, Data fusion, Physics-based simulation

1 Introduction

Physics-based simulations play a pivotal role in facilitating the design and advancement of autonomous construction equipment such as automated excavators and trucks [1]. These simulations have provided an accelerated and safe approach to train, validate, and test the control algorithms and prototype designs of autonomous construction equipment before real-world implementation [1,2]. Moreover, in the quest to leverage deep learning for developing AI-enabled autonomous construction equipment, physics-based simulations can generate a wealth of annotated training data in a short amount of time [3]. This is particularly valuable in situations where data is difficult to obtain in the real world. In addition, physics-based simulations are increasingly employed by researchers to apply and refine reinforcement learning algorithms, thereby enhancing the operational intelligence of autonomous construction equipment [4,5].

Despite the advantages, a key challenge in utilizing physics-based simulations is the preparation and generation of construction equipment models that can accurately represent equipment geometry and physics. Inaccurate models can result in a sim-to-real gap, where algorithms and designs proven in simulations fail when applied in the real world [6]. Although simulation platforms such as Unity, Gazebo, and Isaac Sim offer environments to build these models, the physics-based modeling process is still time-consuming and requires modeling expertise [1,7]. This is because complex data required for the simulation, such as construction equipment materials, joints, drives, etc. need to be integrated during the modeling process, which requires a lot of manual work [2].

Some studies have proposed the use of automatic physics-based modeling methods to reduce the manual effort in the context of modular robots. Modular robots

are systems composed of standardized modules, which can be combined in various ways to adapt to different tasks or environments [8]. Jace et. al [9] presented an automatic approach to model the robot kinematics and dynamics for modular robots, given only the module data and their arrangements. Maddalena et. al [7] proposed an algorithm that takes as input the Unified Robotics Description Format (URDF) files of the single modules with their desired arrangement and provides the final URDF of the assembled robot as a result. However, these methods still require manual configuration of modeling data such as materials and joints for each module. This is not a burden for modeling modular robots because only a few modules need to be configured and they can be reused for robot modeling. However, these methods are not effective in reducing the workload of physics-based modeling of construction equipment as they are far more complex and varied than standard modular robots.

This paper introduces an automated physics-based modeling method through data fusion to streamline the process of creating equipment models for physics-based simulation. Inputs to this method include a construction equipment 3D CAD model along with required data for physics-based modeling, including materials, collision meshes, component relationships, joints, and drives. Initially, the 3D CAD model of construction equipment is converted into a universal scene description (USD) using Isaac Sim to facilitate data fusion. Then, our method automatically fuses the required data for simulation into the USD model. To demonstrate the effectiveness of our method, we created models of an excavator (Caterpillar 390F LME) using both the proposed method and the manual modeling method. Then we compared the time required by these two methods to complete the modeling. The results show that our method can greatly improve the modeling efficiency, and thereby can help promote the application of physics-based simulation in the development of autonomous construction equipment.

2 Literature Review

2.1 Physics-based Modeling and Simulation Platforms

The evolution of simulation platforms such as Unity, Unreal Engine, Gazebo, Isaac Sim, and Webots has significantly impacted the field of robotics. These platforms offer diverse functionalities and environments for robot modeling, each with unique characteristics that distinguish them from one another [10].

Unity, primarily known for its widespread use in game development, has emerged as a versatile platform for robot simulation. Its user-friendly interface and robust physics engine make it an attractive choice for simulating complex robotic systems [11]. Unity's real-time 3D development capabilities enable the creation of detailed and dynamic environments, which are essential for testing the interaction of robots with their surroundings. The platform supports a wide range of robot models, from simple wheeled robots to more complex humanoid robots, allowing for extensive experimentation and research in robotics [10].

Unreal Engine stands out for its high-fidelity graphics and realistic simulation environments [12]. This platform is particularly favored for applications requiring photorealistic rendering, such as autonomous vehicle testing [13]. Unreal Engine's advanced lighting and shading capabilities contribute to creating highly immersive simulation scenarios. It is adept at simulating sophisticated robot models, including drones and autonomous vehicles, providing a realistic platform for testing sensors and navigation algorithms [14].

Gazebo, an open-source simulation platform, is renowned for its strong community support and extensive library of robot models and environments [15]. Its ability to simulate both indoor and outdoor environments with various physics engines makes it a versatile tool for robotics research. Gazebo is particularly popular for simulating multi-robot systems, such as swarm robots, and has been instrumental in numerous robotics competitions and research projects [16].

Webots is a user-friendly, cross-platform simulation software widely used in education and research. Its ease of use and comprehensive documentation make it accessible to both beginners and experienced users [17]. Webots support a broad range of robot models, from simple mobile robots to more advanced humanoid robots, making it versatile tool for various robotic applications.

Isaac Sim, developed by NVIDIA, is tailored for robotics applications involving artificial intelligence (AI) and machine learning (ML). Its integration with NVIDIA's GPU technology enables high-performance simulations, crucial for training and testing AI algorithms [18]. Isaac Sim is adept at simulating complex robotic systems, such as robotic arms and mobile robots, and is particularly beneficial for scenarios involving ML and sensor processing.

2.2 Physics-based Modeling of Construction Equipment

Physics-based modeling has been widely used in autonomous construction equipment training and testing. To demonstrate control of large robots to perform construction tasks, Lei et al. [19] created a construction robot hand model in Isaac Sim, and trained it via reinforcement and imitation learning to conduct operations with 6 types of construction tools, such as power drill, flat screwdriver, adjustable wrench, etc.

Similarly, Sungjin et al. [20] employed Gazebo for dynamic modeling of spraying robots, evaluating their performance in construction tasks like indoor wall painting. Jaco et al., [21] built a wheeled robot model using Gazebo and then trained a map corner-based navigation model in a virtual world. Lofgren et al. [22] advanced this field by simulating an underground loader in Unity, training a deep reinforcement learning controller that autonomously adapts to varying terrains and soil conditions. Azulay and Shapiro [23] also used Gazebo for wheel loader modeling, achieving a controller adept at complex earthmoving tasks, and showcasing the potential for automation in construction.

Furthermore, physics-based simulation is used for generating synthetic data. Wilfredo et al. [24] used Unity to simulate excavator postures, creating a dataset that bypasses the need for time-intensive manual annotation. Jia et al. [25] established a drone model in Unity for capturing simulated dam images, facilitating the training of dam defect detection algorithms.

2.3 Automatic Physics-based Modeling

Despite these available physics-based modeling and simulation platforms, manually building models in simulation platforms is still time-consuming and requires modeling expertise [7]. Some studies have proposed the use of automatic modeling methods to reduce manual modeling effort, and have investigated the automatic modeling process in the context of modular robots. For example, [9] introduced an automated approach for modeling robot kinematics and dynamics, requiring only module data such as joints, drives, etc. and their arrangements. Maddalena et. al [7] proposed an algorithm that takes as input the URDF files of the single modules with their desired arrangement and provides the final URDF of the assembled robot as a result.

However, these methods have shortcomings that limit their use for modeling complex construction equipment. They require manual configuration of joints and additional data within the modeling software, obligating users to acquire proficiency in the software itself. Furthermore, for construction equipment lacking a substantial array of universal modules, these methods offer no advantage over direct manual modeling, thereby confining their applicability primarily to modular robotics.

3 Methodology

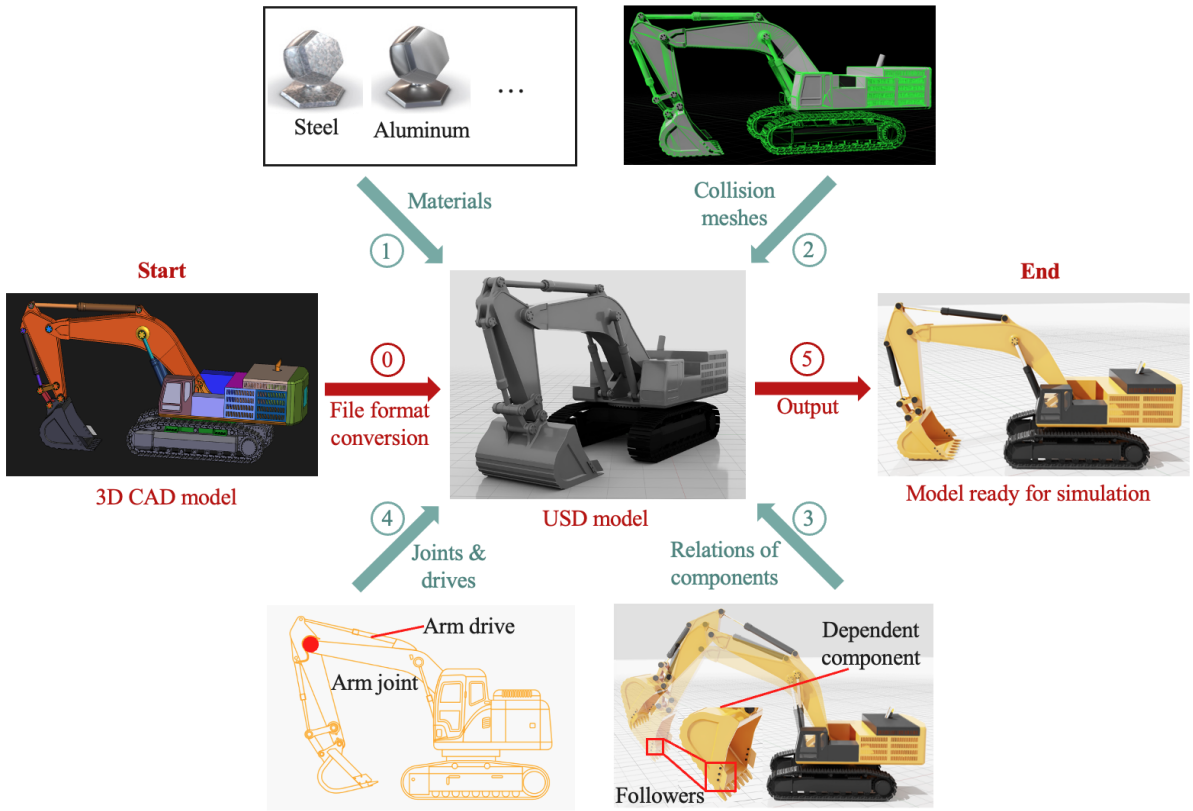

Figure 1. Overview of the automatic physics-based modeling method

The proposed method is designed to automate the physics-based modeling process of construction equipment. The inputs for the method include a 3D CAD model of construction equipment and other data required for physics-based modeling including materials, collision meshes, relations of components, joints, and drives. As shown in Figure 1, the 3D CAD model is initially converted into a universal scene description (USD) model using Isaac Sim to facilitate data fusion. Then, the various data are fused into the USD model. The overall fusion process comprises four steps: Firstly, the materials of components are set based on the materials data, which enables the components to have material information such as density and friction coefficient. Secondly, collision meshes are set to their corresponding components. This allows the components to emulate collision behaviors. Thirdly, motion dependencies between the components are established based on their relationships, which allows components to move with their dependent components accordingly. For example, bucket bolts will move with the bucket. Lastly, joints and drives are created based on their mechanical properties, which allows the components connected with the joints to move accordingly. At this point, the USD model integrated with the data is ready for simulation.

3.1 Model Conversion

To facilitate the fusion of data requisite for the simulation, the initial step involves converting the 3D CAD model of construction equipment into a USD model. USD is an open-source 3D scene description file format developed by Pixar. It can be used for 3D content creation and interchange among different tools [26,27].

There are two reasons for choosing the USD format. The first is its dual support for both intricate construction equipment modeling and complex environmental constructs [28], thus facilitating the import of construction equipment models into its operating environment. Meanwhile, USD has a Python Application Programming Interface (API), and the USD model can be easily customized through Python script [29]. This conversion lays the groundwork for subsequent data fusion. Moreover, USD supports a variety of simulation platforms including Unity [30], Unreal [30], Isaac Sim [31], etc.

3.2 Data Fusion

Subsequent to the model conversion, data fusion can start. The first step of data fusion focuses on setting material properties to each component of the USD model. This involves a process where both the mechanical and aesthetic properties of materials are defined. Mechanical properties include aspects such as density, elasticity, and friction coefficients, which are crucial for accurate physical interactions in the simulation. These mechanical properties can be retrieved from technical specifications provided by construction equipment manufacturers. Aesthetic properties, on the other hand, involve visual characteristics like color, texture, and reflectivity, enhancing the visual realism of the model. Color codes and textures for construction equipment can be obtained through Internet search. Reflectivity can be determined by selecting the corresponding material from the rendering software, such as stainless steel, paint, etc. To organize and store these properties, a JavaScript Object Notation (JSON) document, designated as JSON-1, is compiled. This document serves as a comprehensive repository for the material attributes. Following this, an automated process retrieves the components and their material parameters from the document. The USD model is then systematically scanned, and the documented material parameters are applied to each respective component. This mapping ensures that each component of the USD model is a true-to-life representation of its physical counterpart, mirroring it in both functionality and appearance.

The method then employs an automated process to parse JSON-1 and integrate the material properties into the USD model, ensuring each component reflects its real-world counterpart both in function and form.

Following the materials setting, the second step focuses on setting collision meshes to components of the USD model. Collision meshes are simplified representations of the physical shape of each component. It provides an efficient way to detect and respond to collisions between robot components and/or their surrounding environment. The setting of collision meshes enables the construction equipment model to have realistic interaction within the simulation environment. Our method provides a range of collision mesh estimation methods for users to choose from, including triangle mesh, convex decomposition, convex hull, etc. In this step, JSON-2 document is created to encapsulate the collision mesh estimation method for each component in the USD model. The proposed method then systematically parses this document, extracting component names and their corresponding collision mesh estimation methods, and cataloging them into a Python dictionary structure where the component name is the key, and the collision mesh method is the value. Following this, all components in the USD model are traversed and collision meshes are set for the components listed in the dictionary accordingly.

The third step involves setting up relationships between components. The mutual relationship amongst components determines which components can move together as a group. For example, the excavator bucket body and bolts on it move when the bucket moves, as the bolt and body are connected to the bucket, as shown in

Figure 1. In order to encapsulate the relationships among components, this method creates a relationship document JSON-3. This document employs a dictionary structure to chronicle the relationships among the various components. Keys are utilized to denote the names of dependent components, while values enumerate the associated followers. For example, the above relation is encoded as {“bucket”: [“bucket body”, “bucket bolt1”, “bucket bolt2” ... “bucket bolt6”]}. Within the USD model, these intercomponent relationships are depicted through parent-child hierarchies. All components that are children of a parent component are expected to move together when the parent component moves. Our technique proceeds to parse these relationships from the dictionary and then traverse all the components in the USD model hierarchically to check whether the current child-father relationship of each component is consistent with that in the dictionary. If any discrepancies are found, the parent and child designations are realigned accordingly. This iterative process continues until every component relationship in the USD model has been validated, guaranteeing an accurate representation of movement in the simulation environment.

The last step is to add the joints and drives. In simulations, a joint refers to a functional connection between rigid bodies that facilitates a specific range of relative motion between them. This motion is typically enabled by a drive mechanism. For instance, the rotational movement of car wheels around an axle is attributed to revolute joints. If a wheel is designated as powered, a corresponding drive will be added to actuate it. In our approach, details pertaining to the joints and drives are stored in a document called JSON-4. This document includes the designation of the joint, the components it connects to, the associated drive mechanisms, and the physical parameters of the joint, such as damping coefficient, and stiffness, among others. These parameters can be obtained by from construction equipment manufacturers or by theoretical calculations. After obtaining this data, this method will add these joints and drives accordingly to the USD model.

4 Implementation and Results

To verify the effectiveness of the proposed method, we established the same excavator model on the Isaac Sim platform using both the manual modeling method and the proposed method. Then we compared the modeling time required by the two methods.

4.1 Experimental Environment

The experimental environment used in this study includes a server with an AMD Ryzen 9 5950X CPU running Ubuntu 20.04 system, NVIDIA GeForce RTX 3090Ti GPU with 24G memory of a single graphics card, Nvidia Isaac Sim 23.01.

The construction equipment selected for this experiment is a Caterpillar 390F LME excavator. Its 3D CAD model is downloaded from GRABCAD, as shown in Figure 2. This particular model is engineered with four hydraulic cylinders responsible for actuating the movement of its boom, arm, and bucket. Additionally, it features a swing joint that facilitates the rotation of its upper structure and two actuated sprockets that empower the excavator to advance, retreat, and turn. A breakdown of all the joints and drives incorporated in this model is systematically cataloged in Table 1. This excavator is composed of 142 joints, with a majority of 138 revolute joints that allow for rotational movements and 4 prismatic joints that facilitate linear actions. In conjunction with this, the model incorporates 7 distinct drives, which are instrumental in actuating the various movements of this excavator.

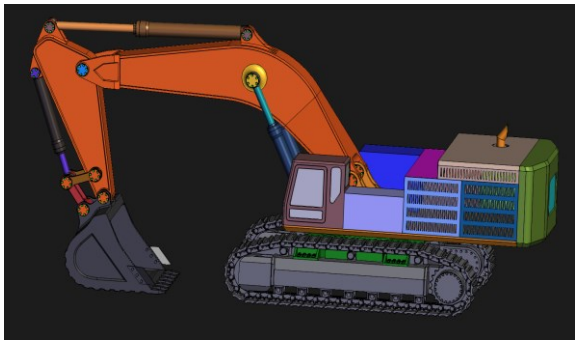

Figure 2. The 3D CAD model of Caterpillar 390F LME

Table 1 Joints and drives statics

Joint/drive type	Number
Revolute joint	138
Prismatic joint	4
Angular drive	3
Linear drive	4

4.2 Results

Upon integrating the CAD model of the excavator with the JSON documents comprising the requisite data for physics-based modeling, we successfully generated an excavator model ready for simulation. Subsequently, in our evaluation conducted on the simulation platform, we tested an array of motion functions pertinent to the excavator. The outcomes of these tests demonstrated that our model is adept at replicating all the essential motion functions of the excavator. This is depicted in Figure 3, which showcases a sequence of video frames extracted from the simulation outcomes. These frames distinctly highlight the model's capability to accurately simulate the dynamic movements of critical components, including the boom, arm, bucket, track, and the excavator's upper body.

To determine the efficiency of our proposed method, we compared the modeling time required by the proposed method and the manual method. Before starting this modeling experiment, we had no experience in robot modeling. During the experiment, we recorded the time spent learning to model using Isaac Sim, which took a total of 28 hours. The duration of excavator modeling from start to completion was then recorded, which lasted 72 hours. Finally, the time required for excavator modeling through the method proposed in this study was recorded, which was 8 hours. Almost all of these 8 hours were used to prepare the data required for modeling. The running time of the automatic modeling program was almost negligible, lasting less than 2 seconds. This comparison, shown in Table 2, reveals a stark contrast in time investment. The manual modeling process required approximately 100 hours. In contrast, our method needed only 8 hours to prepare documentation for materials, collision meshes, component relations, and joints and drives needed for physics-based modeling. Upon completion of this preparatory phase, the true efficacy of our approach becomes evident. It automatically fuses all the data from these documents into the final model, accomplishing this complex integration in an instant. This significant reduction in time, without compromising accuracy or detail, underscores the potential of our method to revolutionize the efficiency of physics-based modeling processes.

Figure 3. Video frames of excavator simulation in Isaac Sim platform

Table 2 Modeling time comparison

Modeling method	Time (hour)
Manual modeling	100
Our method	8

5 Conclusion

Physics-based simulations offer a rapid and secure platform for training, validating, and testing control algorithms, as well as prototyping designs for autonomous construction equipment. Our paper introduces an innovative method to automate the physics-based modeling process using data fusion. This method transforms a 3D CAD model of construction equipment into a USD model, seamlessly integrating the necessary data for physics-based modeling through data fusion. When users apply the proposed method to build other construction equipment models for physics-based simulation, they only need to prepare the 3D CAD model of the construction equipment and the corresponding physical data. Our method can then be used to automatically integrate these data into the CAD model, enabling physics-based modeling. The study benchmarks the efficiency of this automated method against traditional manual modeling within an identical experimental setup, revealing marked enhancements in modeling efficiency.

However, it is imperative to acknowledge certain limitations of our method. Firstly, it requires users to manually prepare the initial data required for physics-based modeling. Additionally, as of now, the method does not possess the capability to incorporate sensors into the models. In the future, we will integrate a large language model such as ChatGPT, into the physics-based modeling pipeline. It will be used to extract and prepare

initial data from construction equipment technical specifications, eliminating the need for users to manually prepare this data. In addition, we will develop a function for adding sensors. Using this functions, users would only need to input the sensor type, location, and other parameters, and the sensor would be automatically integrated into the model.

References

[1] O. Wong Chong, J. Zhang, R.M. Voyles, B.C. Min, BIM-based simulation of construction robotics in the assembly process of wood frames, Autom Constr 137 (2022) 104194. https://doi.org/10.1016/J.AUTCON.2022.104194.

[2] C.K. Liu, D. Negrut, The Role of Physics-Based Simulators in Robotics, Annu Rev Control Robot Auton Syst 4 (2021) 35–58. https://doi.org/10.1146/annurev-control-072220-093055.

[3] S. Mukhopadhyay, Y. Chen, S. Morais, N. Cennamo, J. Lee, J. Boone, C. Goodin, L. Dabbiru, C. Hudson, L. Cagle, D. Carruth, Training Artificial Intelligence Algorithms with Automatically Labelled UAV Data from Physics-Based Simulation Software, Applied Sciences 2023, Vol. 13, Page 131 13 (2022) 131. https://doi.org/10.3390/APP13010131.

[4] A.A. Apolinarska, M. Pacher, H. Li, N. Cote, R. Pastrana, F. Gramazio, M. Kohler, Robotic assembly of timber joints using reinforcement learning, Autom Constr 125 (2021) 103569. https://doi.org/10.1016/J.AUTCON.2021.103569.

[5] O. Azulay, A. Shapiro, Wheel Loader Scooping Controller Using Deep Reinforcement Learning, IEEE Access 9 (2021) 24145–24154. https://doi.org/10.1109/ACCESS.2021.3056625.

[6] Y. Chebotar, A. Handa, V. Makoviychuk, M. MacKlin, J. Issac, N. Ratliff, Di. Fox, Closing the sim-to-real Loop: Adapting simulation randomization with real world experience, Proc IEEE Int Conf Robot Autom 2019-May (2019) 8973–8979. https://doi.org/10.1109/ICRA.2019.8793789.

[7] M. Feder, A. Giusti, R. Vidoni, An approach for automatic generation of the URDF file of modular robots from modules designed using SolidWorks, Procedia Comput Sci 200 (2022) 858–864. https://doi.org/10.1016/J.PROCS.2022.01.283.

[8] V. Nezhadali, O.K. Kayani, H. Razzaq, M. Tarkian, EVALUATION OF AN AUTOMATED DESIGN AND OPTIMIZATION FRAMEWORK FOR MODULAR ROBOTS USING A PHYSICAL PROTOTYPE, DS 68-4: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 4: Product and Systems Design, Lyngby/Copenhagen, Denmark, 15.-19.08.2011 (2011) 195–204. https://www.designsociety.org/publication/30545/EVALUATION+OF+AN+AUTOMATED+DESIGN+AND+OPTIMIZATION+FRAMEWORK+FOR+MODULAR+ROBOTS+USING+A+PHYSICAL+PROTOTYPE (accessed December 8, 2023).

[9] C. Nainer, M. Feder, A. Giusti, Automatic Generation of Kinematics and Dynamics Model Descriptions for Modular Reconfigurable Robot Manipulators, IEEE International Conference on Automation Science and Engineering 2021-August (2021) 45–52. https://doi.org/10.1109/CASE49439.2021.9551680.

[10] J. Collins, S. Chand, A. Vanderkop, D. Howard, A review of physics simulators for robotic applications, IEEE Access 9 (2021) 51416–51431. https://doi.org/10.1109/ACCESS.2021.3068769.

[11] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, D. Lange, Unity: A General Platform for Intelligent Agents, (2018). https://arxiv.org/abs/1809.02627v2 (accessed December 13, 2023).

[12] C. Symeonidis, N. Nikolaidis, Simulation environments, Deep Learning for Robot Perception and Cognition (2022) 461–490. https://doi.org/10.1016/B978-0-32-385787-1.00023-3.

[13] Tom. Shannon, Unreal Engine 4 for Design Visualization Developing Stunning Interactive Visualizations, Animations, and Renderings., (2017). https://books.google.com/books/about/Unreal_Engine_4_for_Design_Visualization.html?id=HfYtDwAAQBAJ (accessed December 13, 2023).

[14] B. Alvey, D.T. Anderson, A. Buck, M. Deardorff, G. Scott, J.M. Keller, Simulated Photorealistic Deep Learning Framework and Workflows To Accelerate Computer Vision and Unmanned Aerial Vehicle Research, (2021) 3889–3898. https://github.com/ (accessed December 13, 2023).

[15] N. Koenig, A. Howard, Design and use paradigms for Gazebo, an open-source multi-robot simulator, 2004 IEEE/RSJ International

Conference on Intelligent Robots and Systems (IROS) 3 (2004) 2149–2154. https://doi.org/10.1109/IROS.2004.1389727.

[16] J. Harbin, S. Gerasimou, N. Matragkas, A. Zolotas, R. Calinescu, Model-Driven Simulation-Based Analysis for Multi-Robot Systems, Proceedings - 24th International Conference on Model-Driven Engineering Languages and Systems, MODELS 2021 (2021) 331–341. https://doi.org/10.1109/MODELS50736.2021.00040.

[17] O. Michel, WebotsTM: Professional Mobile Robot Simulation, (2004). https://arxiv.org/abs/cs/0412052v1 (accessed December 13, 2023).

[18] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, G. State, Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning, (2021). https://arxiv.org/abs/2108.10470v2 (accessed December 13, 2023).

[19] L. Huang, W. Cai, Z. Zhu, Z. Zou, Dexterous manipulation of construction tools using anthropomorphic robotic hand, Autom Constr 156 (2023) 105133. https://doi.org/10.1016/J.AUTCON.2023.105133.

[20] S. Kim, M. Peavy, P.C. Huang, K. Kim, Development of BIM-integrated construction robot task planning and simulation system, Autom Constr 127 (2021) 103720. https://doi.org/10.1016/J.AUTCON.2021.103720.

[21] J.C. Virgolino Soares, G.F. Abati, G.H. Duarte Lima, M.A. Meggiolaro, Autonomous Navigation System for a Wall-painting Robot based on Map Corners, 2020 Latin American Robotics Symposium, 2020 Brazilian Symposium on Robotics and 2020 Workshop on Robotics in Education, LARS-SBR-WRE 2020 (2020). https://doi.org/10.1109/LARS/SBR/WRE51543.2020.9306998.

[22] S. Backman, D. Lindmark, K. Bodin, M. Servin, J. Mörk, H. Löfgren, Continuous Control of an Underground Loader Using Deep Reinforcement Learning, Machines 2021, Vol. 9, Page 216 9 (2021) 216. https://doi.org/10.3390/MACHINES9100216.

[23] O. Azulay, A. Shapiro, Wheel Loader Scooping Controller Using Deep Reinforcement Learning, IEEE Access 9 (2021) 24145–24154. https://doi.org/10.1109/ACCESS.2021.3056625.

[24] W. Torres Calderon, D. Roberts, M. Golparvar-Fard, Synthesizing Pose Sequences from 3D Assets for Vision-Based Activity Analysis, Journal of Computing in Civil Engineering 35 (2021) 04020052. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000937/ASSET/4F7E1E4F-2244-4815-B1E8-4398B6104731/ASSETS/IMAGES/LARGE/FIGURE14.JPG.

[25] J. Xu, C. Yuan, J. Gu, J. Liu, J. An, Q. Kong, Innovative synthetic data augmentation for dam crack detection, segmentation, and quantification, Struct Health Monit 22 (2023) 2402–2426. https://doi.org/10.1177/14759217221122318/ASSET/IMAGES/LARGE/10.1177_14759217221122318-FIG19.JPEG.

[26] M.A. Bolstad, Large-Scale Cinematic Visualization Using Universal Scene Description, 2019 IEEE 9th Symposium on Large Data Analysis and Visualization, LDAV 2019 (2019) 85–86. https://doi.org/10.1109/LDAV48142.2019.8944362.

[27] Pixar Animation Studios, (n.d.). https://www.pixar.com/usd (accessed March 9, 2024).

[28] Introduction to USD — Universal Scene Description 24.03 documentation, (n.d.). https://openusd.org/release/intro.html#usd-can-represent (accessed March 9, 2024).

[29] Working with USD Python Libraries | NVIDIA Developer, (n.d.). https://developer.nvidia.com/usd/tutorials (accessed March 9, 2024).

[30] Universal Scene Description in Unreal Engine | Unreal Engine 4.27 Documentation, (n.d.). https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/USD/USDinUE4/ (accessed March 9, 2024).

[31] Universal Scene Description (USD) 3D Framework | NVIDIA, (n.d.). https://www.nvidia.com/en-us/omniverse/usd/ (accessed March 9, 2024).