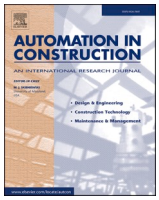
Automation in Construction 168 (2024) 105880

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon



Automated physics-based modeling of construction equipment through data fusion

Liqun Xu [a], Dharmaraj Veeramani [b], Zhenhua Zhu [a,*]

[a] Department of Civil and Environmental Engineering, University of Wisconsin-Madison, 1415 Engineering Drive, Madison, WI 53706, USA
[b] Department of Industrial and Systems Engineering, University of Wisconsin-Madison, 1513 University Avenue, Madison, WI 53706, USA

ARTICLE INFO

Keywords:
Construction equipment modeling
Data fusion
Physics-based simulation
Isaac sim

ABSTRACT

Physics-based simulations are essential for designing autonomous construction equipment, but preparing models is time-consuming, requiring the integration of mechanical and geometric data. Current automatic modeling methods for modular robots are inadequate for construction equipment. This paper explores automating the modeling process by integrating mechanical data into 3D computer-aided design (CAD) models. A template library is developed with hierarchy and joint templates specific for equipment. During model generation, appropriate templates are selected based on the equipment type. Unspecified joint template data is extracted from technical specifications using a large language model (LLM). The 3D CAD model is then converted into a Universal Scene Description (USD) model. Users can adjust the part names and hierarchy within the USD model to align with the hierarchy template, and joint data is automatically integrated, resulting in a simulation-ready model. This method reduces modeling time by over 87 % compared to manual methods, while maintaining accuracy.

1. Introduction

Physics-based simulations are essential for the design and advancement of autonomous construction equipment, such as automated wheel loaders and trucks [1]. These simulations provide an accelerated and safe means to train, validate, and test control algorithms and prototype designs of autonomous construction equipment before real-world implementation [1,2]. Additionaly, physics-based simulations can swiftly generate extensive training data that are necessary for leveraging deep learning (DL) based control algorithms for autonomous construction equipment [3]. This is particularly valuable in scenarios where real-world data acquisition is challenging. Physics-based simulations are also increasingly used to apply and refine reinforcement learning (RL) algorithms, thereby enhancing the operational intelligence of autonomous construction equipment [4,5].

Despite their advantages, a significant challenge in utilizing physics-based simulations lies in the preparation and generation of construction equipment simulation models that accurately represent the equipment's kinematics. This process, known as physics-based modeling, involves creating virtual models that faithfully replicate the physical properties and behaviors of machinery [2]. In this article, we focus specifically on equipment modeling, excluding environment modeling and interactions. Realistic simulations require integrating complex data—such as equipment joints, drives, and collision meshes—into CAD models [2]. This data integration is crucial because it ensures that the simulated equipment behaves in a manner consistent with its physical counterpart. Without this detailed modeling, there is a risk of a simulation-to-reality gap, where algorithms and designs validated in simulations may fail when applied in real-world scenarios [6]. Although simulation platforms such as Unity [7], Gazebo [8], and Isaac Sim [9] provide environments for model creation, the physics-based modeling process remains time-consuming and requires modeling expertise [1,10]. Although engineers can make parameter adjustments on a similar mechanical model, this method is still limited by the availability of similar models and the need for external CAD software for modifications [11,12].

Some studies have proposed automated physics-based modeling methods for modular robots. Modular robots, composed of standardized modules, can be configured in various ways to adapt to different tasks or environments [13]. Jace et al. [14] presented an automated approach to model the kinematics of modular robots based on module data and their arrangements. Maddalena et al. [10,14] proposed an algorithm that processes Unified Robotics Description Format (URDF) files of

* Corresponding author.
E-mail addresses: lxu322@wisc.edu (L. Xu), raj.veeramani@wisc.edu (D. Veeramani), zzhu286@wisc.edu (Z. Zhu).

https://doi.org/10.1016/j.autcon.2024.105880
Received 25 July 2024; Received in revised form 23 October 2024; Accepted 13 November 2024
Available online 21 November 2024

individual modules and their desired arrangement to generate the final URDF of the assembled robot. These methods still require manual configuration of modeling data, such as joints and drives, for each module. Although such methods [10,14] are feasible for modular robots with a limited number of reusable modules, they become significantly time-consuming when applied to construction equipment modeling. This is due to the greater diversity of construction equipment components and the limited reusability of these components, making the process comparable to manual modeling [15–17].

This paper introduces an automated physics-based modeling method through data fusion to streamline the creation of construction equipment models for physics-based simulation. Inputs to this method include the construction equipment type, its 3D CAD model and technical specifications. Initially, during the template preparation stage, a template library containing hierarchy templates and joint templates for each type of construction equipment was prepared. During the simulation model generation stage, the corresponding hierarchy and joint templates are selected from the template library based on the equipment type being modeled by the user. Subsequently, a large language model (LLM) extracts unspecified data in the joint template from the equipment's technical specifications and populates the joint template with this data. Following this, the construction equipment's 3D CAD model is converted into a Universal Scene Description (USD) model to facilitate data fusion. Finally, users need only to adjust the names and hierarchy of parts in the USD model to align with the hierarchy template. The data from the joint template can then be automatically integrated into the USD model, resulting in a simulation-ready model.

To demonstrate the effectiveness of our method, we created a simulation model of a wheel loader (Caterpillar 982 m) and an excavator using both the proposed method and a manual modeling method. Firstly, we validated the accuracy of our method by comparing the specifications of the wheel loader model created by our method and the manual modeling method. Both methods exhibited identical measured specifications. Moreover, the specifications measured in the created model closely matched those of the actual wheel loader. Additionally, we compared the time required to build the wheel loader model and the excavator for each method. The results indicate that our method reduces modeling time by 87 % for the wheel loader and 91 % for the excavator compared to the manual modeling method. These findings suggest that our method can greatly enhance modeling efficiency without compromising accuracy, thereby promoting the application of physics-based simulations in the development of autonomous construction equipment.

2. Related works

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 [18].

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 [19]. 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 complex humanoid robots, allowing for extensive experimentation and research in robotics [18].

Unreal Engine stands out for its high-fidelity graphics and realistic simulation environments [20]. This platform is particularly favored for applications requiring photorealistic rendering, such as autonomous vehicle testing [21]. Unreal Engine's advanced lighting and shading capabilities enable the creation of 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 [22].

Gazebo, an open-source simulation platform, is renowned for its strong community support and extensive library of robot models and environments [23]. 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 [24].

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 [25]. Webots supports a broad range of robot models, from simple mobile robots to more advanced humanoid robots, making it a versatile tool for various robotic applications.

Isaac Sim, developed by NVIDIA, is tailored for robotics applications involving artificial intelligence (AI). Its integration with NVIDIA's GPU technology enables high-performance simulations, crucial for training and testing AI algorithms [26]. Isaac Sim is adept at simulating complex robotic systems, such as robotic arms and mobile robots, and is particularly beneficial for scenarios involving DL and sensor processing.

2.2. Automatic physics-based modeling

Despite these available physics-based modeling and simulation platforms, manually creating models in simulation platforms is still time-consuming and requires modeling expertise [10]. 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. Modular robots, composed of reuseable modules, can be configured in various ways to adapt to different tasks or environments [13]. The typical process of automatic modeling for modular robots involves two steps. First, the kinematics of each individual module are configured manually in advance. Second, the modules are assembled automatically according to the desired arrangement, resulting in the generation of a URDF file that represents the assembled robot. For example, Nainer et al. [14] introduced an automated approach for modeling robot kinematics, requiring only parameter data of the individual modules such as joints, drives, etc. and their arrangements. Maddalena et al. [10] proposed an algorithm that takes as input the URDF files of the single modules with their desired arrangement and provides the final URDF file of the assembled robot as the output.

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

2.3. Physics-based modeling of construction equipment

In the existing literature on physics-based modeling of construction equipment, to the best of our knowledge, no automated physics-based modeling method has been proposed to date. Research in this area has predominantly focused on the application of simulation models, which are primarily employed for training control algorithms [27] and generating synthetic data [28,29].

Physics-based modeling has been widely used in autonomous construction equipment control algorithms training and testing. To demonstrate control of large robots to perform construction tasks, Lei et al. [30] created a construction robot hand model in Isaac Sim, and trained it via reinforcement and imitation learning to conduct operations with six types of construction tools, such as power drill, flat screwdriver, and adjustable wrench. Similarly, Sungjin et al. [31]

employed Gazebo for dynamic modeling of spraying robots, evaluating their performance in construction tasks like indoor wall painting. Jaco et al. [32] built a wheeled robot model using Gazebo and then trained a map corner-based navigation model in a virtual world. Lofgren et al. [28] 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 [27] also used Gazebo for wheel loader modeling, achieving a controller adept at complex earthmoving tasks, and showcasing the potential for automation in construction.

Beyond control algorithms, physics-based simulation has also been instrumental in generating synthetic data [29,33]. Wilfredo et al. [34] used Unity to simulate excavator postures, creating a dataset that bypasses the need for time-intensive manual annotation. Jia et al. [35] established a drone model in Unity for capturing simulated dam images, facilitating the training of dam defect detection model.

3. Methodology

In this paper, we present a procedure for automatic physics-based modeling of construction equipment through data fusion, as illustrated in Fig. 1. This method uses joint templates, hierarchy templates, and 3D CAD model to represent the kinematics of the machinery. The 3D CAD model provides the link information of every component (specific dimensions). The template files document the joint information (e.g., names of the joints, the connected parts, the limits of the motion, the type of drive, and the motion dependency between components). The method takes as input the equipment type, its 3D CAD model and technical specifications, and produces a simulation-ready model. The procedure consists of five steps: data template preparation, data template selection, extraction of undetermined data, model conversion, and data fusion. First, we prepared a template library in advance, which contains hierarchy templates and joint templates for each type of construction equipment. Second, the corresponding hierarchy and joint templates are selected from the template library based on the equipment type input by the user. Third, during data extraction, any undetermined data in the joint template is retrieved from the technical specifications using a large language model (LLM) and incorporated into the template. Fourth, the 3D CAD model is converted into a USD model to facilitate data fusion. Finally, in the data fusion step, the data from the hierarchy and joint templates are integrated into the USD, resulting in a simulation-ready model.

3.1. Data template preparation and selection

When constructing a simulation model, certain data for each kind of construction equipment can be pre-determined based on the equipment part diagram and its working priciple, such as the parts comprising the equipment and the relationships between the movements of these parts. To record this data, reusable templates are prepared for each kind of equipment prior to modeling. Specifically, two templates are created for each kind of equipment: a hierarchy template and a joint template. For existing construction equipment, these templates are pre-configured and stored in a library for future use. In the case of newly designed construction equipment, users can modify the templates of similar existing equipment to accommodate the new designs. For example, if a novel

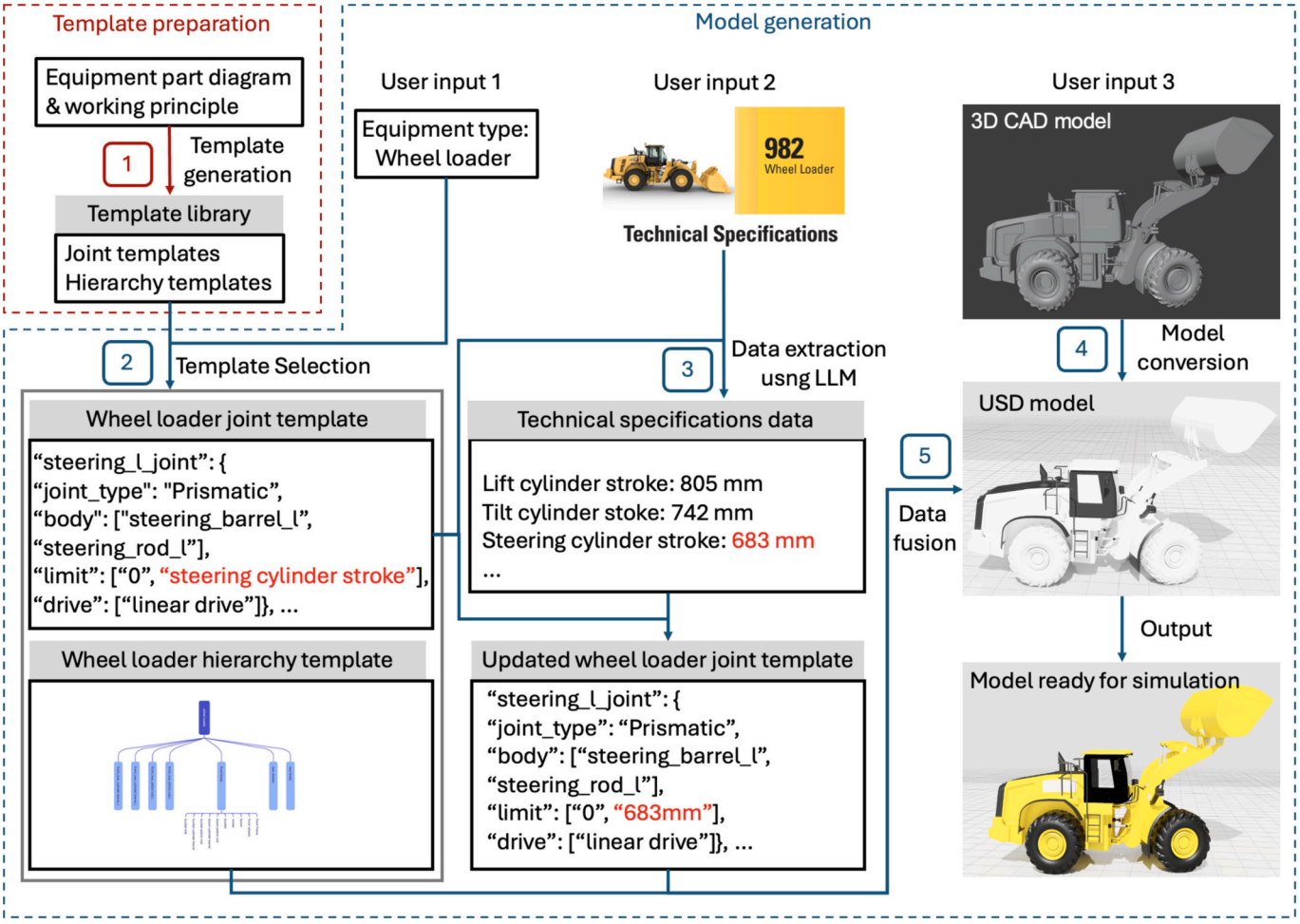


Fig. 1. Overview of the automatic physics-based modeling procedure.

excavator with a unique actuation mechanism is introduced, the user would adjust the joint and hierarchy template to reflect the new actuation ends and structural differences. This flexibility ensures that the proposed methodology can support the development of innovative equipment designs, not just parameterization of existing structures. When generating a simulation model using the proposed method, the corresponding hierarchy and joint template is selected based on the equipment type being modeled by the user.

3.1.1. Hierarchy template preparation

The hierarchy template includes the parts name and their relationships. The relationship among parts determines which parts can move together as a group, essential for accurately modeling their interactions. This relationship can be described as a parent-child relationship. When the parent part moves, all of its child parts should move with it. To represent the parent-child relationship between parts, we have prepared a range of equipment hierarchy templates tailored to various construction equipment, such as wheel loaders, trucks, forklifts, and more. These templates are formatted as tree diagrams.

These tree diagrams can aid users in understanding the hierarchical structure and dependencies within the equipment. For example, Fig. 2 shows a partial illustration of the hierarchical relationship of parts within a wheel loader. The parent part, "wheel loader," is subdivided into several child parts, including the "rear body," "rear wheels," and "front body". Additionally, the "front body" is further decomposed into child parts such as the "front frame" and "front wheels," among others.

3.1.2. Joint template preparation

The joint template includes the parameters for every joint between the parts of the equipment. 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. Otherwise, no drive needs to be added. In our procedure, joints and drives are implemented in the PhysX extensions library and five kinds of joints are used to simulate connections between equipment parts [36].

- Fixed joint: locks the orientations and origins rigidly together.

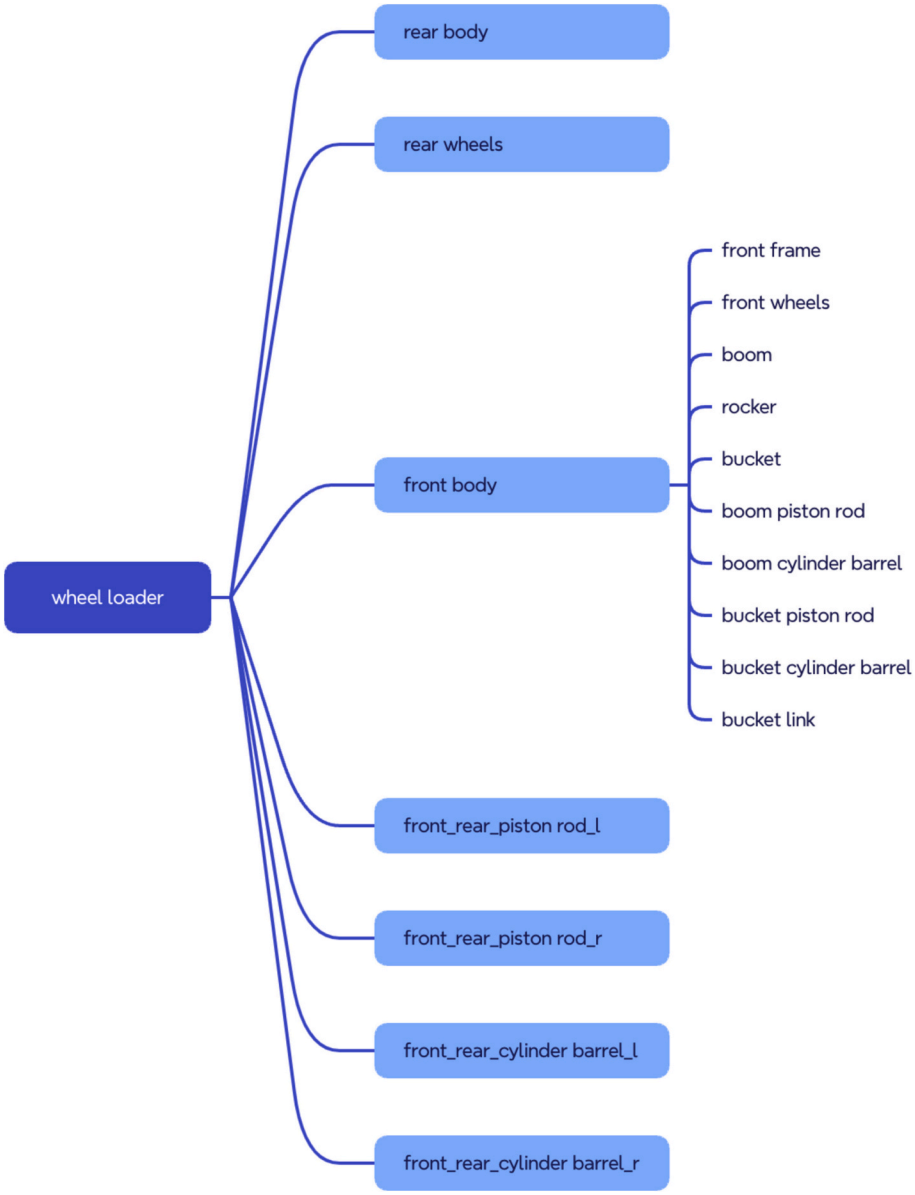


Fig. 2. Hierarchy of a wheel loader.

- Distance joint: keeps the origins within a certain distance range.
- Spherical joint: keeps the origins together but allows the orientations to vary freely.
- Revolute joint: keeps the origins and X-axes of the frames together and allows free rotation around this common axis.
- Prismatic joint: keeps the orientations identical, but allows the origin of each frame to slide freely along the common X-axis.

Two types of drives are used to actuate joints [36].

- Linear drive: is used to control the translational movement of a joint along a specific axis. It is typically used for prismatic joints, which allow linear movement.
- Angular drive: is used to control the rotational movement of a joint around a specific axis. It is typically used for revolute joints, which allow rotational movement.

To represent joint parameters in construction equipment, our method has prepared joint templates designed for various construction equipment in advance. These templates are created based on the equipment part diagram and its working principles, and are formatted in JSON. These templates contain essential parameters for each joint, including the joint's name, type (e.g., fixed joint, revolute joint, etc.), the parts it connects, joint limits, and drive type. It is noteworthy that the names of the parts are the same as the names of the parts in the hierarchy template. Given that certain data, such as joint limits, vary significantly between different models of the same equipment (for instance, a Cat 982 M wheel loader and a Cat 950 M wheel loader), these data cannot be determined prior to modeling. Consequently, these data are not specified in the template. Fig. A1 provides an example of a joint template for a wheel loader. The joint mentioned in line 15, "steering_cylinder_barrel_l2steering_piston_rod_l_joint," is a "Prismatic" joint connecting the "steering_cylinder_barrel_l" and "steering_piston_rod_l" components. The lower limit of this joint is set to "0," but the upper limit is unknown and requires referencing the "Steering Cylinder Stroke" from the technical specifications. This joint is equipped with a "Linear Drive."

In the simulation model generation stage, the corresponding hierarchy and joint templates are selected from the template library based on the equipment type being modeled by the user. For the joint template, some data fields may be unspecified. The names of these unspecified data fields are passed to the undetermined data extraction step to be identified in the equipment's technical specifications.

3.2. Undetermined data extraction

To identify the data left unspecified in the joint template, a LLM is utilized to extract the data from the equipment technical specifications [37] as shown in Fig. 3. This can avoid the effort in manually retrieving this data from lengthy technical specifications. In the data extraction process, the first step is to obtain the equipment's technical

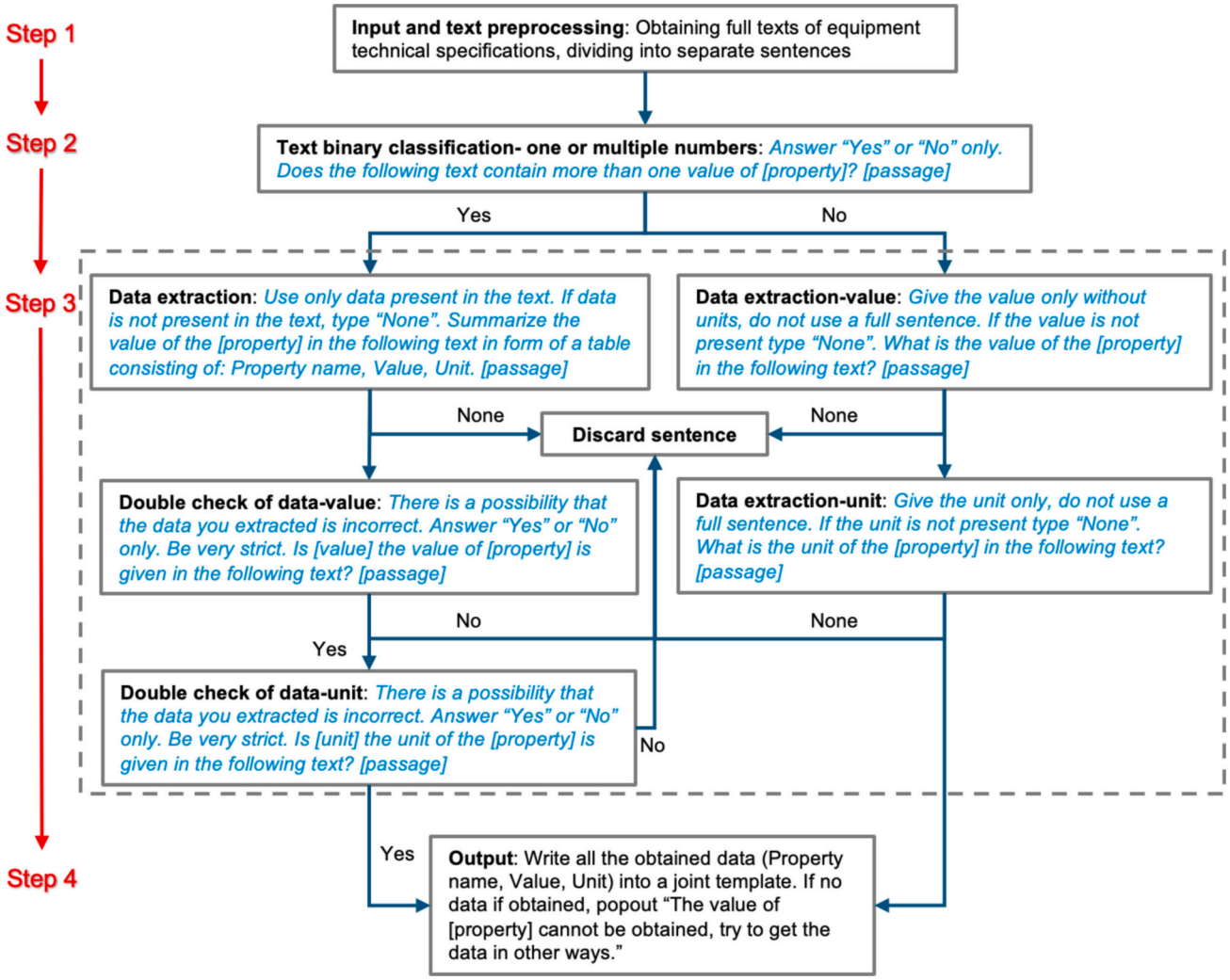


Fig. 3. Flowchart describing the method of extracting structured data using a LLM.

specifications in PDF format and perform preprocessing to split the text into separate sentences according to the general rules of sentence termination. This is the basis for the following steps.

The second step splits the sentences into two categories: those with single values and those with multiple values. The sentences containing a singular value are inherently much simpler, given the straightforward relationship among the property name, its value, and unit. In this case, these data tend to be extracted with high precision by the LLM. Conversely, the extraction of data from sentences with multiple values is more intricate, necessitating further analysis to determine the interrelationships among values, names, and units to accurately pair them. Consequently, this step aims to ascertain the presence of singular or multiple data points within a passage. The determination of the data complexity guides the subsequent process along one of two divergent pathways.

Based on the data complexity determined in the second step, different strategies are used to extract data from the paragraphs in the fifth step. For sentences containing a single value, inquiries regarding the value and its unit are posed separately. An option to negate these inquiries is presented to diminish the probability of LLM fabricating responses. If a negative answer is given to any of these queries, the respective sentence is subsequently discarded and exempted from data extraction. Conversely, for sentences with multiple values, LLM is requested to summarize the data in tabular format, which facilitates orderly data management. Nonetheless, this approach bears the risk of engendering inaccurate data, even with the provision for negative responses. Therefore, the procedure double-checks that the data in the obtained table is indeed contained in the sentence. Again, we explicitly allow negative answers in case the extracted tables may contain some inaccuracies. Similarly, as before, if the answer to any prompt is negative, we discard that sentence. In the final step, all the extracted data are integrated into the corresponding positions in the joint template to obtain the updated joint template. If certain data cannot be found in the technical specifications provided by the user, they will be advised to seek this information through alternative methods. For example, if the limit for “steering_rod_l” in the joint template of wheel loader is unavailable in the technical specifications, users will be encouraged to consult with engineers or search online for the necessary details.

3.3. Model conversion

To facilitate the fusion of data requisite for the simulation, the model conversion step involves converting the 3D CAD model of the construction equipment into a USD format. USD is an open-source 3D scene description file format developed by Pixar [38]. It can be used for 3D content creation and interchange among different tools such as Isaac Sim [9] and Unity [39–41]. The CAD Convertor [42] is used to convert the CAD model into the USD format. This convertor supports multiple CAD file formats, such as DWG (for Autodesk), RVT (for Revit) and STL.

There are three reasons for converting the CAD model into the USD format. Firstly, USD's dual support for both intricate machinery modeling and complex environmental constructs [43] facilitates the import of construction equipment models into its operating environment. This dual capability ensures that both the equipment and the surrounding environment are accurately represented, thereby providing a comprehensive simulation framework. Secondly, USD features a Python Application Programming Interface (API), which facilitates the customization of USD models through Python scripting [44]. This capability lays the foundation for the subsequent automatic integration of data into the USD model. Furthermore, the compatibility of USD with a wide array of simulation platforms, including Unity [45], Unreal [21], and Isaac Sim [46], underscores its versatility and utility in diverse simulation scenarios.

3.4. Data fusion

In the data fusion process, the first step is to fuse data from the selected hierarchy template, which involves setting up relationships between parts in the USD model. In practice, users are required to rename the equipment parts and adjust their parent-child relationships within the converted USD model to align with the predefined hierarchy template. Ensuring that the part names and their relationships accurately match the templates can facilitate the following automated joint data fusion process.

The second step is to fuse the data from the updated joint template. To automatically create joints in the USD model, a Python script was prepared. This script traverses each joint as specified in the updated joint template. It reads the joint's name, the parts it connects, the joint limits, and the drive type. Subsequently, the corresponding parts in the USD model are identified based on the joint names specified in the joint template. Once the corresponding parts are located within the USD model, the joints of the specified type are created. Finally, the joint limits and drive types are written into the USD model, ensuring a precise representation of the equipment's mechanical structure. This automated process ensures that each joint is accurately integrated into the USD model, reflecting the specific mechanical interactions and constraints as outlined in the technical specifications.

4. Implementation and results

To evaluate the proposed procedure, a wheel loader was modeled using both the manual approach and the proposed procedure. We compared the specifications of the real wheel loader with those of the created models to assess the accuracy of the model generated by the proposed procedure and the manual approach. Additionally, we compared the modeling time required by our method with that of the manual approach to evaluate the efficiency of our method.

4.1. Implementation

The experimental environment used in this study includes a server with an AMD Ryzen 95,950× CPU running Ubuntu 20.04 system, NVIDIA GeForce RTX 3090Ti GPU with 24G memory of a single graphics card. The modeling and simulation platform is Isaac Sim 4.0.0.

The construction equipment modeled for this experiment is a Caterpillar 982 M wheel loader and a Caterpillar 390FL excavator. They were chosen as test models because of their widespread use in the construction industry and the availability of their 3D CAD model file online. Its technical specifications are downloaded from the Internet. Their 3D CAD model is downloaded from BlenderKit [47] and GrabCAD [48], as shown in Fig. 4. This wheel loader model is engineered with four hydraulic cylinders responsible for actuating the movement of its boom and bucket. Additionally, it features two hydraulic cylinders that facilitate the turn and two actuated rear wheels that empower the wheel loader to advance and retreat. The excavator model features four hydraulic cylinders that control the movement of its boom, arm, and bucket. It also includes two drive sprockets, enabling the excavator to move forward, backward, and turn. Table 1 presents the statistics for joints and drives in two types of construction equipment: a wheel loader and an excavator. The table indicates that the wheel loader is equipped with 14 revolute joints, 4 prismatic joints, 1 angular drive, and 4 linear drives. In contrast, the excavator has a significantly higher number of revolute joints (138) and angular drives (3), while maintaining the same number of prismatic joints (4) and linear drives (4) as the wheel loader.

In the hierarchy template preparation process, we obtained the motion dependencies between the machinery parts by analyzing the wheel loader and excavator parts diagram [15,16,49,50] and its operation video [51–53]. The hierarchy template was formatted in JSON. This JSON file uses a nested dictionary structure to record the relationship between various parts. Each dictionary contains two keys,

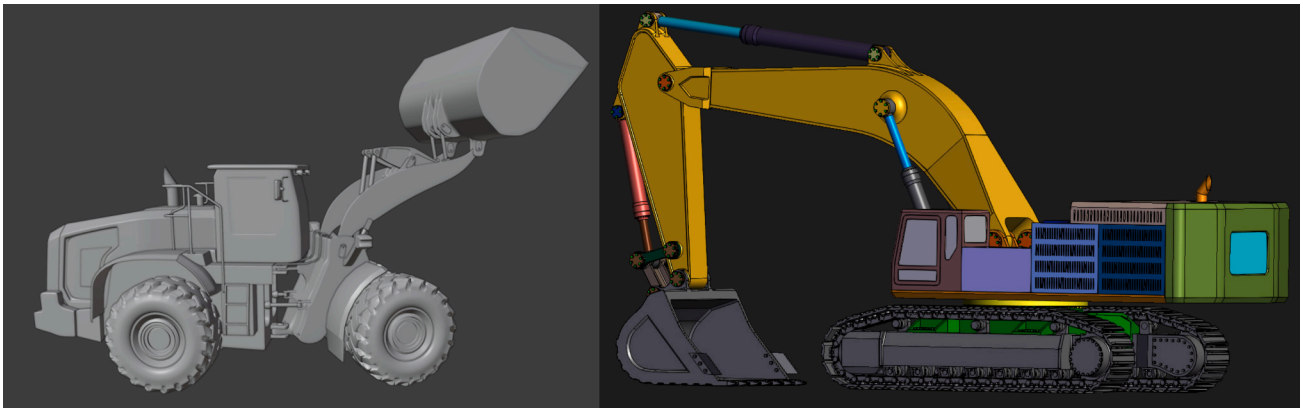


Fig. 4. 3D CAD model of Caterpillar 982 M wheel loader and Caterpillar 390FL excavator.

Table 1
Joints and drives statics.

Joint/drive type	Wheel loader	Excavator
Revolute joint	14	138
Prismatic joint	4	4
Angular drive	1	3
Linear drive	4	4

"rigid body" and "child". Among them, "rigid body" refers to the rigid body attribute of the part, and its value includes "Y" which is a rigid body, and "N" which is not a rigid body. The term "child" refers to which parts will move with the parent part, and its value enumerates the name of the part that moves with it.

In the joint template preparation process, we analyzed the wheel loader and excavator operation video [51–53] to determine the types of motion between the parts at the wheel loader and excavator joints, the parts connected by the joints, and the drive types at the joints. These joint data were recorded in the joint template in JSON format. The JSON file uses a dictionary structure to record each joint's parameters, containing four keys: ."joint type", "body", "limit", and "drive". The "joint type" is a string with values "revolute" or "prismatic," representing revolute joint and prismatic joint, respectively. The value of "body" is a list, with each element representing the name of a part connected by each joint. The value of "limit" is a list containing two elements, the lower and upper limit of the joint movement constraints. Since these constraints can vary between different models of the same equipment, they are specified as constraint limit names in the template (e.g., "Steering Cylinder Stroke" to refer to the constraint controlling the turning cylinder) to maintain applicability across various models. The value of "drive" is a list, with an element representing the drive type, which can be "linear drive", "angular drive", or an empty string indicating no drive.

For model conversion, the CAD Convertor [42] was used to convert the CAD model of wheel loader into an USD file. In the conversion process, the CAD model was first imported through the 'File - > Import' option in Isaac Sim. Import options were set to their default settings. Once import options configured, the converted USD file was imported directly into Isaac Sim.

During the data extraction process, the GPT-4.0 API [54] was used to extract undetermined data in the joint template from the technical specifications, as shown in step 2 in Fig. 1. The extracted data was then added to the joint template and the updated joint template was obtained. A video demonstration of utilizing LLM to extract undetermined data in the wheel loader joint template from the technical specifications can be found at this link: https://youtu.be/Lb5men6Mj1g. Then the hierarchy data from the hierarchy template was first fused into the USD model. This was accomplished by users, who rename the equipment parts and adjust their parent-child relationships within the converted USD model to align with the predefined hierarchy template. After fusing the hierarchy data, the joints data from the updated joint template was automatically fused into the USD model by running the Python script in the "Script Editor" in Isaac Sim. Finally, the model ready for simulation was obtained.

4.2. Results and discussion

4.2.1. Modeling accuracy

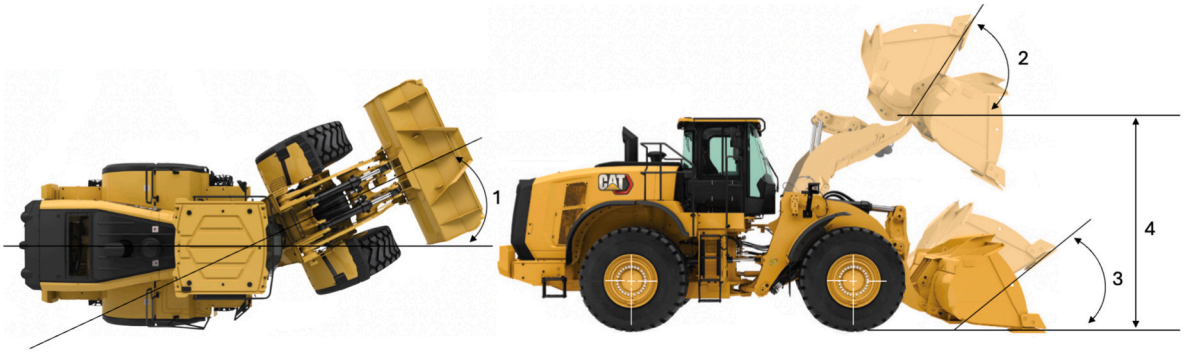
Upon integrating the USD model of the wheel loader and excavator with the necessary data for physics-based modeling, we successfully generated a wheel loader and a excavator model ready for simulation. To qualitatively evaluate the generated models, we tested a range of motion functions pertinent to the wheel loader and excavator. The test results can be found at these links (https://youtu.be/jq4DPszxn0E, https://youtu.be/TFOzm_VohHY). The evaluation demonstrated that our model accurately replicates all the essential motion functions of the machinerys, confirming its functionality for simulation purposes.

To quantitatively evaluate the models created using both the manual modeling approach and the proposed procedure, we compared several specifications of the created models with those of a real wheel loader, as shown in Fig. 5 and Fig. 6. The model created using the proposed procedure exhibited the same measured specifications as the model built using manual method, demonstrating that our procedure achieves the same accuracy as manual modeling. Additionally, the specifications measured in the created model closely matched those of the actual wheel loader and excavator. This consistency indicates that the model generated by the proposed method accurately reflects the kinematics of the actual wheel loader and excavator, thereby confirming the effectiveness and accuracy of our modeling approach.

4.2.2. Modeling time

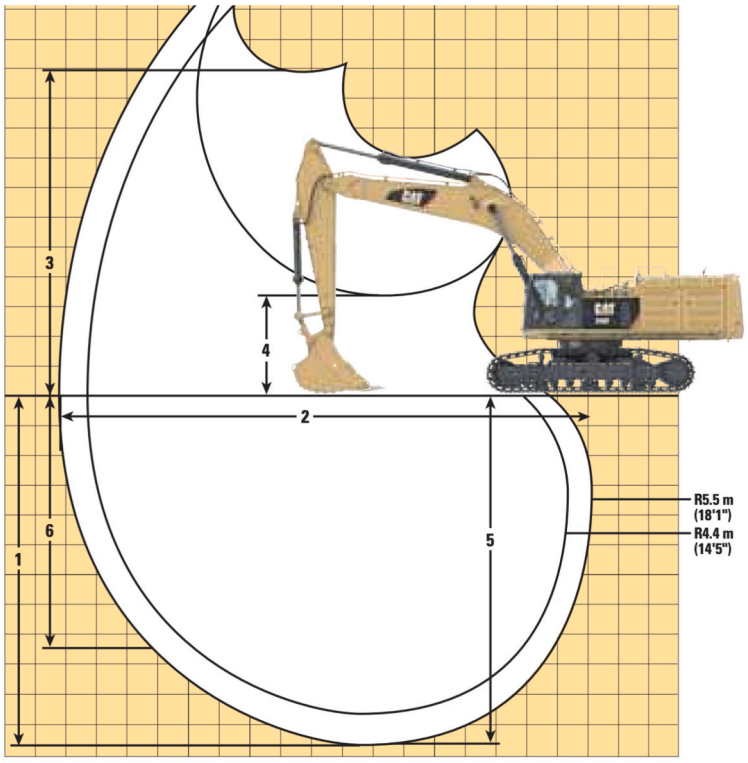
To determine the efficiency of our procedure, we compared the modeling time required by our procedure and the manual method. When comparing the running time, we did not include the 3D CAD model preparing time for both the proposed method and the manual modeling method. Additionally, the time spent learning the software was also not included. We then recorded the time it took the modeler to manually model the wheel loader and excavator, which was 23 h and . Finally, the time required by the modeler to model the wheel loader and excavator using our proposed procedure was recorded.

Table 2 compares the time required to model a wheel loader and an excavator using a manual method versus the proposed method. The manual process, which includes checking the mechanical operation principle, technical specifications, and the actual modeling, took 23 h for the wheel loader and 55.5 h for the excavator. In contrast, the proposed method drastically reduces the time required. Data extraction



NO.	Specification name	Value from specifications	Value from created model
1	Maximum steering angle	25 degrees	24.93 degrees
2	Rack back at maximum lift	57 degrees	57.12 degrees
3	Rack back at ground	39 degrees	39.07 degrees
4	Hinge pin height at maximum lift	4741 mm	4741.44 mm

Fig. 5. Comparison of specification values and corresponding values from the created model for Caterpillar 982 M wheel loader.



NO.	Specification name	Value from specifications	Value from created model
1	Maximum digging depth	11800 mm	11813 mm
2	Maximum reach at ground line	17250 mm	17242 mm
3	Maximum loading height	10960 mm	10975 mm
4	Minimum loading height	3320 mm	3311 mm
5	Maximum depth cut for 2240 mm level bottom	11700 mm	11722 mm
6	Maximum vertical wall digging depth	8380 mm	8390 mm

Fig. 6. Comparison of specification values and corresponding values from the created model for Caterpillar 390FL excavator.

Table 2
Modeling time comparison.

Modeling method		Wheel loader	Excavator
Manual modeling method	Checking the mechanical operation principle	1.5 h	3.5 h
	Checking the technical specifications	2.5 h	4.0 h
	Modeling	19 h	48 h
	Total	23 h	55.5 h
Our method	Data extraction	6 s	13 s
	Data fusion	3 h	5 h
	Total	3 h 6 s	5 h 13 s

took only 6 s for the wheel loader and 13 s for the excavator, while data fusion took 3 h and 5 h, respectively. For both the wheel loader and the excavator, nearly all of the data fusion time was spent adjusting the USD model to align with the hierarchy template, with the automated modeling program itself running in under 7 s for the wheel loader and under 10 s for the excavator. Overall, the total time using the automated method is significantly less, at 3 h 6 s for the wheel loader and 5 h 13 s for the excavator, clearly demonstrating the efficiency of the proposed method compared to the manual approach.

The results of our study demonstrate significant advancements in the efficiency of physics-based modeling for construction equipment. By automating the model creation process, we achieved an 87 % reduction in time for the wheel loader and a 91 % reduction for the excavator compared to manual methods, all without compromising accuracy. This efficiency enables more rapid prototyping and testing of new equipment designs and control algorithms, thereby accelerating the development process. Moreover, the high fidelity of the models generated by our method ensures that simulations closely mimic real-world scenarios, making them a reliable tool for operator training and safety assessments. In practical terms, this means that construction companies can use these simulations not only to enhance the design and functionality of new equipment but also to improve operator skills in a controlled, risk-free environment.

5. Conclusion and future work

Physics-based simulation is crucial in the design and development of autonomous construction equipment. However, preparing construction equipment models that accurately represent the equipment's kinematics is a time-consuming task in simulation. Compared with manual modeling, automatic modeling can significantly reduce modeling time while maintaining accuracy. This paper proposed a method for automated physics-based modeling of construction equipment based on data fusion. A wheel loader and an excavator model were created using both the proposed method and the manual modeling method. The accuracy of the created model and the time required for modeling were evaluated. The results indicate that the proposed method achieves the same accuracy as the manual modeling method while significantly accelerating the process. Specifically, the proposed method requires only 13 % of the time for the wheel loader and 9 % of the time for the excavator compared to the manual modeling process.

However, there are still some limitations in the proposed method that need to be addressed in future work. First, the equipment templates were manually created, making it time-consuming to prepare templates for each type of construction equipment. Future work will focus on developing a method for automatically generating templates based on minimal input information by users. Second, during the data fusion step, the names and hierarchy of the parts in the USD model are manually adjusted to align with the hierarchy template. This is currently the most time-consuming aspect of model generation. We plan to explore automated methods using 3D part recognition algorithms to streamline the adjustment of the USD model hierarchy.

CRediT authorship contribution statement

Liqun Xu: Writing – original draft, Validation, Methodology, Investigation, Formal analysis. **Dharmaraj Veeramani:** Writing – review & editing, Supervision, Project administration, Funding acquisition. **Zhenhua Zhu:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

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.

Data availability

Data will be made available on request.

Acknowledgement

This paper is based in part upon the work supported by the Wisconsin Alumni Research Foundation (WARF) under Project No. AAM3225. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of WARF.

Appendix A. Appendix

joints_template_cat982m.json > ...

```json
{
    "rear_wheel_joint": {
        "joint_type": "Revolute",
        "Body": ["rear_frame","rear_wheels"],
        "Drive": ["Angular Drive"]},
    "rear2front_joint": {
        "joint_type": "Revolute",
        "Body": ["front_frame","rear_front_pin"]},
    "front_frame2steering_piston_rod_l_joint": {
        "joint_type": "Revolute",
        "Body": ["steering_piston_rod_l","steering_piston_rod_l_pin"]},
    "front_frame2steering_piston_rod_r_joint": {
        "joint_type": "Revolute",
        "Body": ["steering_piston_rod_r","steering_piston_rod_r_pin"]},
    "steering_cylinder_barrel_l2steering_piston_rod_l_joint": {
        "joint_type": "Prismatic",
        "Body": ["steering_cylinder_barrel_l","steering_piston_rod_l"],
        "Limit": ["0", "Steering Cylinder Stroke"],
        "Drive": ["Linear Drive"]},
    "steering_cylinder_barrel_r2steering_piston_rod_r_joint": {
        "joint_type": "Prismatic",
        "Body": ["steering_cylinder_barrel_r","steering_piston_rod_r"],
        "Limit": ["0", "Steering Cylinder Stroke"],
        "Drive": ["Linear Drive"]},
    "front_wheel_joint": {
        "joint_type": "Revolute",
        "Body": ["front_frame","front_wheels"]},
    "front_frame2boom_joint": {
        "joint_type": "Revolute",
        "Body": ["boom","boom_pin"]},
    "front_frame2boom_cylinder_barrel_joint": {
        "joint_type": "Revolute",
        "Body": ["boom_cylinder_barrel","boom_cylinder_barrel_pin"]},
```

Fig. A1. Example of a joint template for a wheel loader.

```
    "boom2boom_piston_rod_joint": {
        "joint_type": "Revolute",
        "Body": ["boom","boom_piston_rod"]},
    "boom_cylinder_barrel2boom_piston_rod_joint": {
        "joint_type": "Prismatic",
        "Body": ["boom_piston_rod","boom_cylinder_barrel"],
        "Limit": ["0", "Lift Cylinder Stroke"],
        "Drive": ["Linear Drive"]},
    "front_frame2bucket_cylinder_barrel_joint": {
        "joint_type": "Revolute",
        "Body": ["bucket_cylinder_barrel","bucket_cylinder_barrel_pin"]},
    "bucket_piston_rod2bucket_cylinder_barrel_joint": {
        "joint_type": "Prismatic",
        "Body": ["bucket_piston_rod","bucket_cylinder_barrel"],
        "Limit": ["0", "Tilt Cylinder Stroke"],
        "Drive": ["Linear Drive"]},
    "bucket_piston_rod2rocker_joint": {
        "joint_type": "Revolute",
        "Body": ["rocker","bucket_piston_rod_pin"]},
    "boom2rocker_joint": {
        "joint_type": "Revolute",
        "Body": ["boom","rocker_pin"]},
    "rocker2bucket_link_joint": {
        "joint_type": "Revolute",
        "Body": ["rocker","bucket_link_pin"]},
    "bucket_link2bucket_joint": {
        "joint_type": "Revolute",
        "Body": ["bucket","bucket_link"]},
    "boom2bucket_joint": {
        "joint_type": "Revolute",
        "Body": ["boom","bucket_pin"]}
}
```

Fig. A1. (*continued*).

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, Ann. Rev. Control Robot. Autonom. 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, Appl. Sci. 13 (2023) 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, in: Proceedings - IEEE International Conference on Robotics and Automation 2019-May, 2019, pp. 8973–8979, https://doi.org/10.1109/ICRA.2019.8793789.

[7] Unity Real-Time Development Platform, 3D, 2D, VR & AR Engine. https://unity.com/, 2024 (accessed May 12, 2024).

[8] Gazebo. https://gazebosim.org/home, 2024 (accessed May 12, 2024).

[9] Isaac Sim - Robotics Simulation and Synthetic Data, NVIDIA Developer. https://developer.nvidia.com/isaac/sim, 2024 (accessed May 12, 2024).

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

[11] R. Fujimoto, C. Bock, W. Chen, E. Page, J.H. Panchal, Research Challenges in Modeling and Simulation for Engineering Complex Systems, 2017, p. 87, https://doi.org/10.1007/978-3-319-58544-4.

[12] R. Leroux, M. Pantel, I. Ober, J.M. Bruel, Model-based systems engineering for systems simulation, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11246 LNCS, 2018, pp. 429–448, https://doi.org/10.1007/978-3-030-03424-5_29/FIGURES/11.

[13] V. Nezhadali, O.K. Kayani, H. Razzaq, M. Tarkian, Evaluation of an automated design and optimization framework for modular robots using a physical prototype, in: 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, pp. 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).

[14] C. Nainer, M. Feder, A. Giusti, Automatic generation of kinematics and dynamics model descriptions for modular reconfigurable robot manipulators, in: IEEE International Conference on Automation Science and Engineering 2021-August, 2021, pp. 45–52, https://doi.org/10.1109/CASE49439.2021.9551680.

[15] What are the main components of a wheel loader, 2024. https://lugongma.com/what-are-the-main-components-of-a-wheel-loader/ (accessed July 11, 2024).

[16] Wheel Loader Parts - Construction Equipment Parts. https://ceparts.com/products/machine-type/wheel-loader/, 2024 (accessed July 11, 2024).

[17] C.Y. Ling, R. Ghazali, R.S. Hameed, A.H. Fadel, An expert system for engine excavator troubleshooting, J. Soft Comp. Data Min. 1 (2020) 53–61, https://doi.org/10.30880/jscdm.2020.01.02.006.

[18] 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.

[19] 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. https://arxiv.org/abs/1809.02627v2, 2018 (accessed December 13, 2023).

[20] C. Symeonidis, N. Nikolaidis, Simulation environments, Deep Learn. Robot Percept. Cognit. (2022) 461–490, https://doi.org/10.1016/B978-0-32-385787-1.00023-3.

[21] Tom Shannon, Unreal Engine 4 for Design Visualization Developing Stunning Interactive Visualizations, Animations, and Renderings. https://books.google.com/books/about/Unreal_Engine_4_for_Design_Visualization.html?id=HfYtDwAAQBAJ, 2017 (accessed December 13, 2023).
[22] 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, pp. 3889–3898. https://github.com/ (accessed December 13, 2023).
[23] 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) 3, 2004, pp. 2149–2154, https://doi.org/10.1109/IROS.2004.1389727.
[24] J. Harbin, S. Gerasimou, N. Matragkas, A. Zolotas, R. Calinescu, Model-driven simulation-based analysis for multi-robot systems, in: Proceedings - 24th International Conference on Model-Driven Engineering Languages and Systems 2021, MODELS, 2021, pp. 331–341, https://doi.org/10.1109/MODELS50736.2021.00040.
[25] O. Michel, WebotsTM: Professional Mobile Robot Simulation. https://arxiv.org/abs/cs/0412052v1, 2004 (accessed December 13, 2023).
[26] 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. https://arxiv.org/abs/2108.10470v2, 2021 (accessed December 13, 2023).
[27] 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.
[28] S. Backman, D. Lindmark, K. Bodin, M. Servin, J. Mörk, H. Löfgren, Continuous control of an underground loader using deep reinforcement learning, Machines 9 (2021) 216, https://doi.org/10.3390/MACHINES9100216.
[29] L. Xu, H. Liu, B. Xiao, X. Luo, Z. Zhu, Synthetic simulated data for construction automation: a review, construction research congress 2024, CRC 2024 (1) (2024) 527–536, https://doi.org/10.1061/9780784485262.054.
[30] 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.
[31] 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.
[32] 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, in: 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.
[33] L. Xu, H. Liu, B. Xiao, X. Luo, D. Veeramani, Z. Zhu, A systematic review and evaluation of synthetic simulated data generation strategies for deep learning applications in construction, Adv. Eng. Inform. 62 (2024) 102699, https://doi.org/10.1016/J.AEI.2024.102699.
[34] W. Torres Calderon, D. Roberts, M. Golparvar-Fard, Synthesizing pose sequences from 3D Assets for vision-based activity analysis, J. Comput. Civ. Eng. 35 (2021) 04020052, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000937/ASSET/4F7E1E4F-2244-4815-B1E8-4398B6104731/ASSETS/IMAGES/LARGE/FIGURE14.JPG.
[35] 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-FIG 19.JPEG.
[36] Joints — Physx 5.3.1 Documentation. https://nvidia-omniverse.github.io/PhysX/physx/5.3.1/docs/Joints.html#rack-and-pinion-joint, 2024 (accessed May 22, 2024).
[37] M.P. Polak, D. Morgan, Extracting accurate materials data from research papers with conversational language models and prompt engineering, Nat. Commun. 15:1 (2024) 1–11, https://doi.org/10.1038/s41467-024-45914-8.
[38] Pixar Animation Studios. https://www.pixar.com/usd, 2024 (accessed May 26, 2024).
[39] M.A. Bolstad, Large-scale cinematic visualization using universal scene description, in: 2019 IEEE 9th Symposium on Large Data Analysis and Visualization 2019, LDAV, 2019, pp. 85–86, https://doi.org/10.1109/LDAV48142.2019.8944362.
[40] Pixar Animation Studios. https://www.pixar.com/usd, 2024 (accessed March 9, 2024).
[41] Unity - Manual: USD. https://docs.unity3d.com/2020.1/Documentation/Manual/com.unity.formats.usd.html, 2024 (accessed March 9, 2024).
[42] CAD Converter — Omniverse Extensions Latest Documentation. https://docs.omniverse.nvidia.com/extensions/latest/ext_cad-converter.html, 2024 (accessed June 4, 2024).
[43] Introduction to USD — Universal Scene Description 24.03 Documentation. https://openusd.org/release/intro.html#usd-can-represent, 2024 (accessed March 9, 2024).
[44] Working with USD Python Libraries | NVIDIA Developer. https://developer.nvidia.com/usd/tutorials, 2024 (accessed March 9, 2024).
[45] Universal Scene Description in Unreal Engine | Unreal Engine 4.27 Documentation. https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/USD/USDinUE4/, 2024 (accessed March 9, 2024).
[46] Universal Scene Description (USD) 3D Framework | NVIDIA. https://www.nvidia.com/en-us/omniverse/usd/, 2024 (accessed March 9, 2024).
[47] Wheel loader | FREE Industrial Vehicles models | BlenderKit. https://www.blenderkit.com/asset-gallery-detail/2dd30522-086c-4566-9b2a-b9234f89458f/, 2024 (accessed May 26, 2024).
[48] Caterpillar 390F L | 3D CAD Model Library | GrabCAD. https://grabcad.com/library/caterpillar-390f-l-1, 2024 (accessed September 14, 2024).
[49] Wheel Loader Part Diagram. https://www.conequip.com/part-diagram-wheel-loader, 2024 (accessed July 11, 2024).
[50] Excavator Parts Diagram | Company Wrench. https://www.companywrench.com/excavator-part-diagram/, 2024 (accessed September 14, 2024).
[51] (30) How to Operate a Wheel Loader (ep. 065) - YouTube. https://www.youtube.com/watch?v=eLYVCQ1_aGY, 2024 (accessed June 9, 2024).
[52] (30) Revolite's 3D Modelling & Animation- Wheel Loader #wheelloader #revolite #3danimation #3dmodeling - YouTube. https://www.youtube.com/watch?v=VR-E21o0o7k, 2024.
[53] (150) Excavator Training & Operation (Beginner), Heavy Equipment Operator Training - YouTube. https://www.youtube.com/watch?v=QWvKa1Bkak8, 2020 (accessed September 14, 2024).
[54] Models - OpenAI API. https://platform.openai.com/docs/models/gpt-3-5-turbo, 2024 (accessed June 9, 2024).