Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Synthetic Image Data Generation via Rendering Techniques for Training AI-Based Instance Segmentation

Dickson Kho Yik Cheng[1], Norazlianie Sazali[1,2,*], Ismayuzri Ishak[1], Saiful Anwar Che Ghani[3]

[1] Faculty of Manufacturing and Mechatronic Engineering Technology, University Malaysia Pahang, 26600 Pekan, Pahang, Malaysia
[2] Centre of Excellence for Advanced Research in Fluid Flow (CARIFF), Universiti Malaysia Pahang Al-Sultan Abdullah, Kampung Melayu Gambang, 26300 Kuantan, Pahang, Malaysia
[3] Faculty of Mechanical and Automotive Engineering, University Malaysia Pahang, 26600 Pekan, Pahang, Malaysia

ARTICLE INFO

Article history:
Received 30 April 2024
Received in revised form 7 October 2024
Accepted 15 October 2024
Available online 20 November 2024

Keywords:
Synthetic image data generation; Rendering techniques; BlenderProc; COCO annotations

ABSTRACT

Synthetic image data generation has gained popularity in computer vision and machine learning in recent years. The work introduces a technique for creating artificial image data by utilizing 3D files and rendering methods in Python and Blender. The technique employs BlenderProc, a rendering tool for generating artificial images, to efficiently create a substantial amount of data. The output of the method is saved in JSON format, containing COCO annotations of objects in the images, facilitating seamless integration with current machine-learning pipelines. The paper shows that the created synthetic data can be used to enhance object data during simulation. The method can enhance the accuracy and robustness of machine-learning models by modifying simulation parameters like lighting, camera position, and object orientation to create a variety of images. This is especially beneficial for applications that require significant amounts of labelled real-world data, which can be time-consuming and labour-intensive to obtain. The study addresses the constraints and potential prejudices of creating synthetic data, emphasizing the significance of verifying and assessing the generated data prior to its utilization in machine learning models. Synthetic data generation can be a valuable tool for improving the efficiency and effectiveness of machine learning and computer vision applications. However, it is crucial to thoroughly assess the potential limitations and biases of the generated data. This paper emphasizes the potential of synthetic data generation to enhance the accuracy and resilience of machine learning models, especially in scenarios with limited access to labelled real-world data. This paper introduces a method that efficiently produces substantial amounts of synthetic image data with COCO annotations, serving as a valuable resource for professionals in computer vision and machine learning.

1. Introduction

In automated assembly processes with robotic systems, accurately detecting and manipulating components in their workspace is a crucial challenge [1,2]. Integrating AI-based instance segmentation methods has become a key solution to overcome these obstacles [3]. Instance

* *Corresponding author.*
E-mail address: azlianie@umpsa.edu.my

https://doi.org/10.37934/araset.53.1.237248

segmentation is a specific task in computer vision that involves identifying and distinguishing individual objects within an image or video, while also classifying each object [4]. This methodology primarily depends on the capabilities of deep neural networks, advanced AI models carefully crafted to identify patterns and connections in large amounts of input data [5].

The effectiveness of deep neural networks in segmentation tasks depends heavily on the quality and quantity of training data available to them [6]. Having a varied and extensive dataset helps the model understand many different object examples and their unique characteristics, which improves its ability to make predictions accurately and generally [7-9]. Acquiring a large, annotated dataset containing real-world objects is a difficult task that requires a lot of effort and resources [10].

Creating real object data involves a detailed process that includes collecting images or videos, accurately annotating objects, and carefully curating balanced and diverse datasets [11,12]. Every stage requires meticulous focus on specifics, where manual annotation alone requires numerous hours dedicated to outlining exact boundaries around each object and assigning suiTable class labels [13]. Obtaining real-world data is hindered by various obstacles, such as privacy restrictions, restricted access to certain objects or environments, and the scarcity of specific setups. Ensuring the dataset covers different lighting conditions, object orientations, and spatial arrangements increases complexity, requiring capturing images or videos under various controlled conditions [14,15].

Generating real object data is time-consuming and poses a significant obstacle to creating and implementing AI-based instance segmentation models in robotic systems [16]. This bottleneck causes inefficient performance in automated assembly processes, leading to reduced efficiency, increased error rates, and rising costs [17]. Therefore, finding new methods to speed up the creation of large training datasets is crucial for improving the performance of robotic systems in automated assembly tasks [18,19].

This paper aims to provide a solution to expedite the data acquisition process by automatically generating synthetic datasets. This method aims to overcome the limitations of acquiring extensive, varied, and labelled real-world datasets for training AI-driven computer vision models, specifically instance segmentation models, by utilizing Computer-Aided Design (CAD) data and a compatible rendering engine. This paper outlines a methodical strategy for creating synthetic data by utilizing CAD data and a corresponding renderer as the central component of the process. The following sections will explain the complexities of this process, clarifying the workflow from obtaining the CAD model to annotating data, and ending with the post-processing of synthetic data for optimal use in various applications.

2. Methodology

2.1 BlenderProc Pipeline

Synthetic data is crucial in computer vision for training and assessing machine learning models, especially in cases where real-world data is scarce or poses privacy issues [20]. One method of creating synthetic data involves utilizing the BlenderProc pipeline and 3D files [21,22]. BlenderProc is a sturdy system created for rendering 3D information in Blender, an open-source software for 3D modelling and animation. This method includes executing the rendering process for 3D files and acquiring the COCO annotations of the objects in the scene [23]. The pipeline enables the creation of high-quality synthetic data for use in different computer vision tasks.

The BlenderProc pipeline relies on Python scripts to manage and automate the creation of synthetic data. Users can tailor the rendering pipeline to their specific requirements, like training machine learning models, conducting simulations, or testing algorithms, by utilizing Python's adaptability and user-friendly interface [24]. BlenderProc utilizes Blender for the rendering process,

taking advantage of its wide array of tools and features for developing, modifying, and rendering 3D models and environments [25] Utilizing blend files guarantees compatibility and user-friendliness, as BlenderProc smoothly integrates with Blender's proprietary file format [26].

2.2 Flow Chart

The flowchart for generating synthetic data with BlenderProc consists of essential steps aimed at creating authentic and useful datasets for different computer vision applications, as shown in Figure 1. The first step of the BlenderProc workflow involves loading and manipulating 3D objects in the environment. This involves importing 3D models in various formats and carefully adjusting attributes such as position, rotation, and scale to intricately compose the scene. Accurately setting camera parameters is crucial for creating a realistic and authentic artificial scene. Users can adjust the camera's position, orientation, and various settings such as focal length, field of view, and depth of field. This allows them to recreate specific visual effects or mimic real-world camera setups, ultimately enhancing the quality of the images produced.

The precise setup of lighting conditions is a crucial part of creating synthetic data. BlenderProc allows users to use different types of lighting, including area, sunlight, points, or spotlights, and adjust properties like intensity, colour and shadow settings. These adjustments are crucial for creating accurate and realistic representations of the 3D environment, which are essential for the success of future computer vision projects. Once users have finished setting up the objects, camera, and lighting, they start the scene generation process by running the BlenderProc Python script. This script generates 3D scenes and produces custom synthetic images designed for various computer vision goals.

After creating a 3D scene successfully, BlenderProc automatically creates COCO annotations in JSON format for synthetic images. The annotations provide essential information about objects in the scene, including bounding boxes, segmentation masks, and keypoints. They are essential tools for training and assessing machine learning models, enhancing the effectiveness and resilience of future computer vision projects. When the quality of the 3D scene is not as expected, an iterative refinement process begins. Users continuously fine-tune different settings related to objects, cameras, or lighting conditions to perfect the scene according to the specific criteria set for the computer vision task.

2.3 Concept

The concept behind BlenderProc lies in its aim to automate and streamline the synthesis of image data through a modular, customizable, and user-friendly pipeline that seamlessly integrates Blender's functionality with the specific requirements for generating synthetic datasets. This methodology entails constructing custom pipelines tailored to unique requirements, delineating structure and parameters via configuration files (in YAML or JSON format), and leveraging Blender's Python API to automate data synthesis. By harnessing Blender's rendering capabilities, BlenderProc generates photorealistic synthetic images while facilitating automatic annotation generation in popular formats such as COCO or PASCAL VOC. This comprehensive approach not only expedites dataset creation but also enhances the realism and diversity of synthetic datasets, catering to various computer vision tasks including object detection, segmentation, and classification.

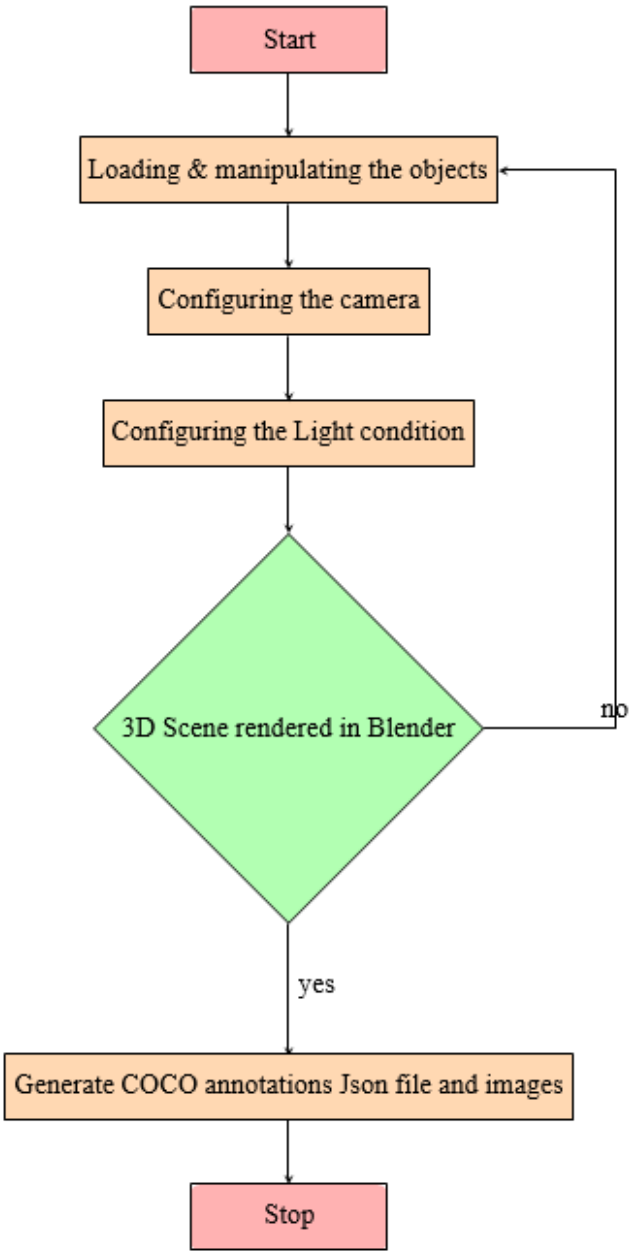


Fig. 1. Flow chart

2.4 Implementation

BlenderProc's implementation process includes coordinating a pipeline and setting up the necessary modules to create the desired dataset. To create synthetic images of screws using BlenderProc, follow these steps: Start by verifying that the 3D screw files are compatible with Blender and meet modelling and texturing standards. Subsequently, proceed to install Blender and BlenderProc by adhering to the guidelines provided in the documentation. Next, create a Python script utilizing the BlenderProc API to automatically produce synthetic data. Load screw objects, adjust their positions and rotations, and create varied environments and perspectives by introducing randomized lighting and camera configurations. Enable physics simulation to achieve lifelike interactions in the scene. Configure rendering settings, enable segmentation masks for objects, render the scene, and save the generated data with COCO-format annotations for training and

evaluation. These steps result in the creation of the intended synthetic image dataset, shown in Figure 2.

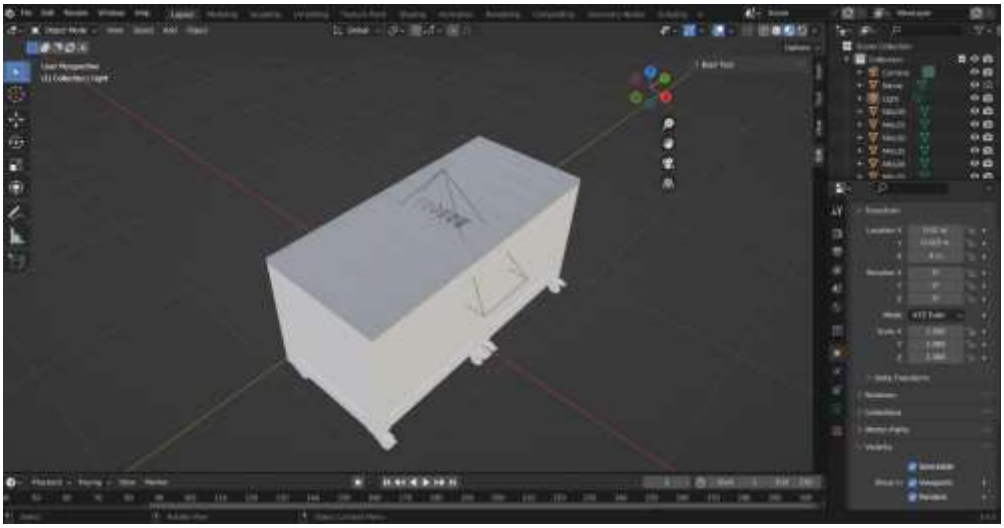


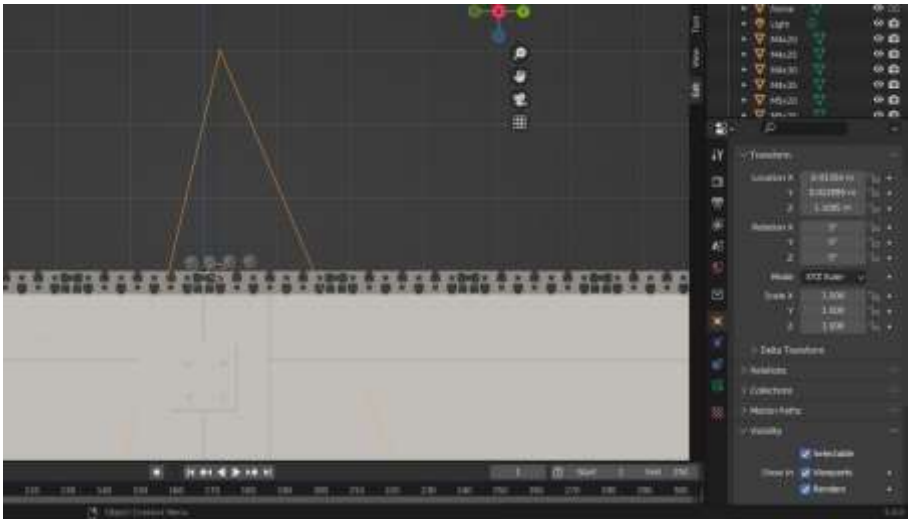
Fig. 2. The 3D files for simulation

3. Results and Discussion

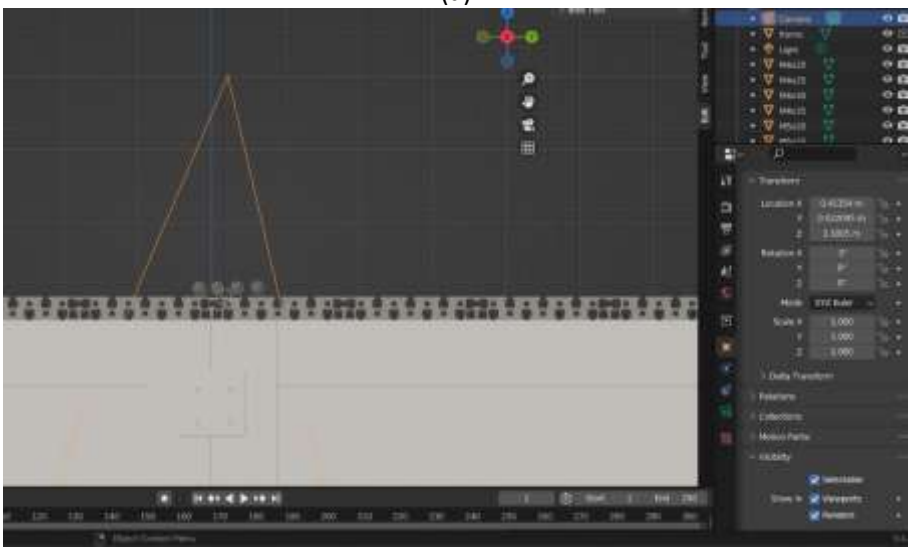
The BlenderProc pipeline is an advanced tool that utilizes Blender, an open-source 3D modelling and animation software, to create synthetic image data. We performed 2500 simulations in this study, introducing random factors to create a varied and extensive dataset, suiTable, for training and assessing machine learning models.

3.1 Simulation Results

The simulation process involves randomly placing objects in the scene to ensure that each rendered image has a distinct arrangement of objects. Applying physics to objects creates natural interactions that lead to realistic training scenarios. Moreover, camera movement is randomized in these simulations. The camera moves dynamically along the y-axis and rotates within a range of -5 to 5 degrees. Illustrated in Figure 3, The intentional changes in the camera's position and orientation help capture scenes from different angles and perspectives, enhancing the dataset with essential variability for effective model training.



(a)



(b)

Fig. 3. The camera is being adjusted to have Euler rotation of (a) 5 degree (b) -5 degree

The synthetic data generation process involves integrating random lighting conditions, as shown in Figure 4. BlenderProc enables the randomization of light positions, types of light sources, and intensity in Figure 5, creating various lighting scenarios for synthetic images. Increasing the variety of lighting conditions improves the dataset's ability to perform well in different environments.

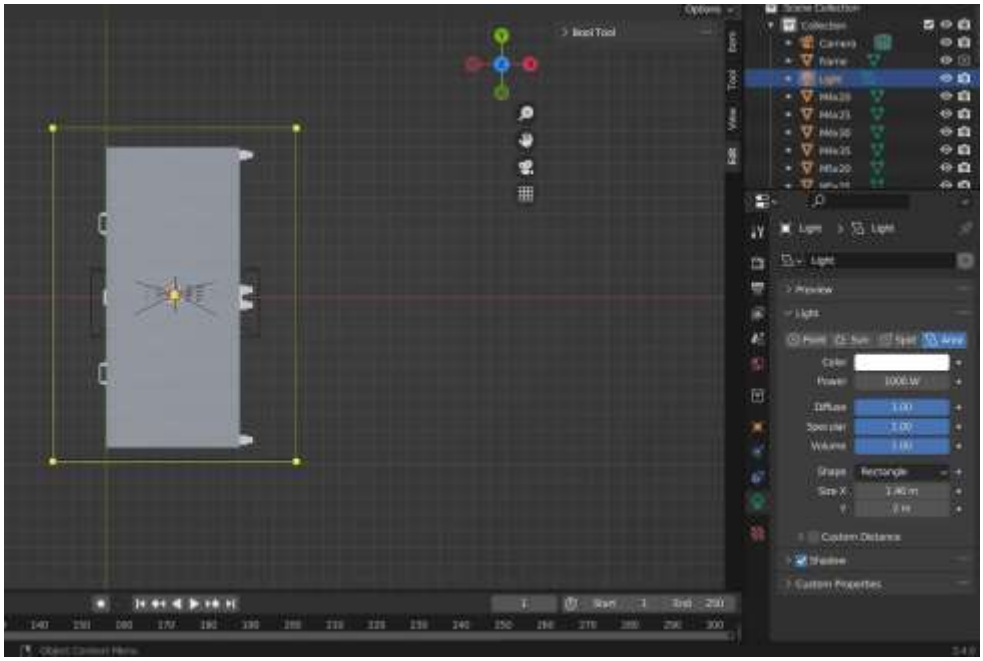
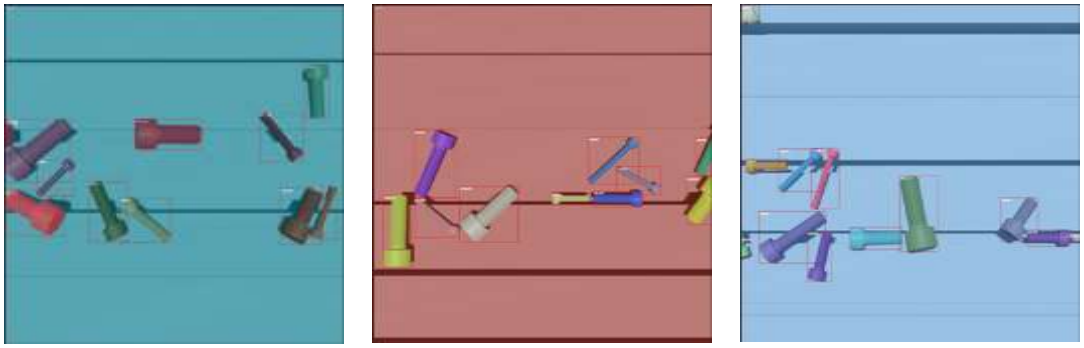


Fig. 4. The lighting setting

The type of lighting is set to "AREA," which means that the lighting is only focused on certain areas. The intensity of the lighting is being set to a value between 200W and 800W. The position of the lighting is fixed or can also be random around the camera and the objects.

(a) (b) (c)

Fig. 5. Pictures of generated image data (a) First generated image, (b) Second generated image and (c) Third generated image



(a) (b) (c)

Fig. 6. COCO visualization of (a) First generated image, (b) Second generated image and (c) Third generated image

3.2 Number of Screws in the Generated Data

Table 1 specifies that the camera must identify a total of 24 screws during the rendering process. The screws' positions on the table are randomized using a Python script and the BlenderProc pipeline.

Table 1
Table of number of screws

	Number occurs			
Screws	X20	X25	X30	X35
M4	1272	1264	1295	1211
M5	1272	1312	1248	1290
M6	1227	1312	1242	1300
M8	1236	1233	1244	1259
M10	1286	1247	1270	1265
M12	1260	1301	1266	1281

Figure 7 displays a histogram illustrating the distribution of the number of screws in each generated image. The figures above clearly show that the positions of the screws were randomized during the simulation. As a result, only specific screws are visible in the camera view, while others are hidden because of their random placement. Due to the inherent randomness in the screws' positions, only a portion of the screws is anticipated to be visible in each image captured by the camera. Additionally, the camera's continuous Y-axis movement will not show all screws at once. This observation highlights the importance of evaluating the effectiveness and influence of the randomization technique on the quantity of screws present in each produced image, as determined from the data sheet.

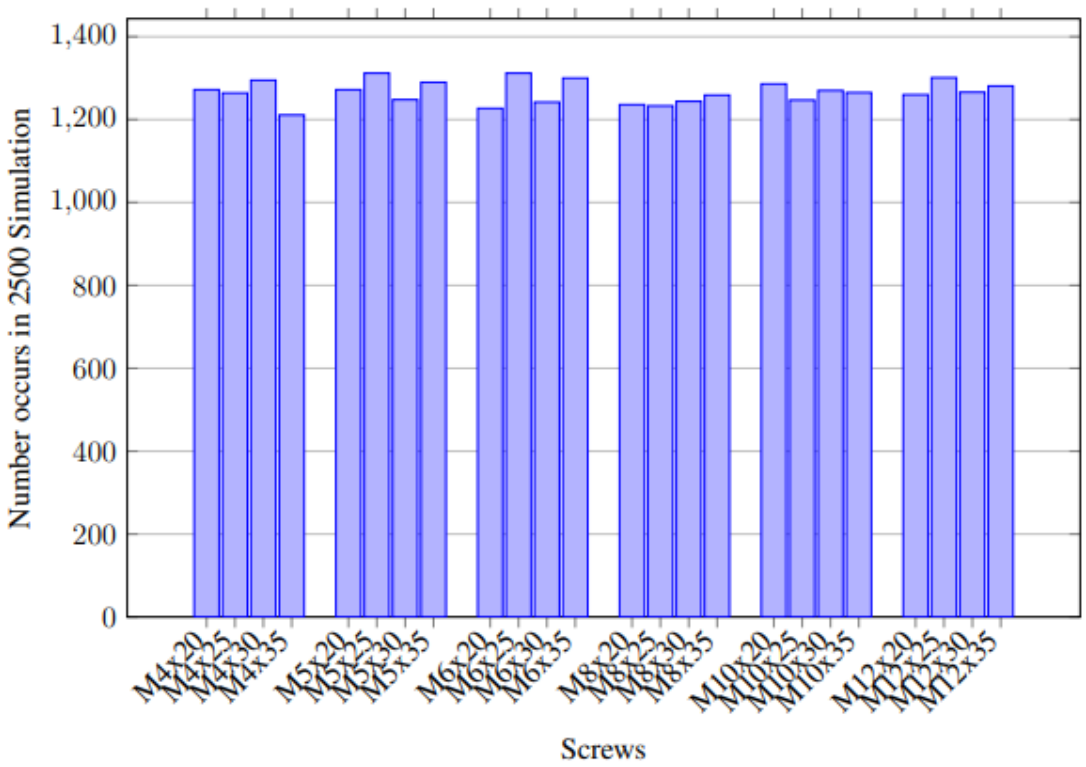


Fig. 7. Histogram

The table and histogram summarize the occurrences of different types of screws in 2500 simulations. The screws are identified by their diameter, indicated by "M," followed by a number (M4, M5, M6, M8, M10, and M12), and their length, measured in millimetres (20, 25, 30, and 35). The x-label on the graph represents the screws' specifications by combining the diameter and length (e.g., M4x20), while the y-label shows the frequency of each screw type.

The M4 screws have the following combinations: M4x20 has 1272 instances, M4x25 has 1264, M4x30 has 1295, and M4x35 has 1211. In the same way, M5 screws have four different sizes: M5x20 (1272 times), M5x25 (1312 times), M5x30 (1248 times), and M5x35 (1290 times). These are the number of times that M6 screws happen: 1227 times for M6x20, 1312 times for M6x25, 1242 times for M6x30, and 1300 times for M6x35. M8 screws come in four different sizes: 1236 times for M8x20, 1233 times for M8x25, 1244 times for M8x30, and 1259 times for M8x35.

The last number in the table is 1265. M10x20 appears 1286 times, M10x25 1247 times, M10x30 1270 times, and M10x35 1265 times. M12 screws come in four different sizes: M12x20 (1260 times), M12x25 (1301 times), M12x30 (1266 times), and M12x35 (1281 times).

The data provides a detailed summary of the distribution of various screw types according to their diameter and length, as well as their frequency of occurrence. This information is useful for understanding the frequency and utilization of these screws in different applications.

3.3 Data Comparison to Real Data

Real data acquisition involves using a similar randomization process for the screwspositions. Following this, the screws in the images are manually labelled or plotted using online labelling tools. Figure 8 displays pictures of the first, second, and third real images, demonstrating that the real images are not significantly different from the generated synthetic image data. Acquiring real data is a more time-consuming process than generating synthetic data.

(a) (b) (c)

Fig. 8. Pictures of (a) First real image, (b) Second real image and (c) Third real image

3.4 Discussion

It is crucial to critically analyse the process of creating synthetic image data with BlenderProc to progress the field and improve its effectiveness. Examining the data is crucial for ensuring accuracy, enhancing authenticity, maximizing model efficiency, recognizing constraints, and promoting innovation [23].

One aspect that requires more research is the velocity and effectiveness of the artificial data creation process. The simulation time is slow despite attempts to optimize CPU thread utilization, suggesting possible inefficiencies. A thorough study is required to investigate hardware elements like CPU and GPU performance, as well as memory and storage capacities. Software optimizations in BlenderProc, such as codebase efficiency and rendering settings, need to be carefully examined to find areas for enhancement and improve simulation speed [28,29].

Additionally, a thorough examination of parameters concerning the simulation's camera settings and lighting conditions is necessary to create synthetic data that closely mimics real-world situations. By analysing parameters such as field of view, camera distance, sensor size, and focus distance, one can adjust camera settings to enhance the realism of image generation [30]. Examining various types of light, intensities, and their effects on shadows and reflections is essential for generating artificial data with improved realism and visual accuracy [31]. Researchers can optimize and innovate in the fields of computer vision and machine learning by exploring these areas and understanding their impact on the process of generating synthetic data.

4. Conclusions

This study has examined the use of synthetic image data generation methods with BlenderProc, a robust Python tool for producing and manipulating 3D files. We used BlenderProc's flexibility to create a variety of intricate scenarios with random object and camera placements, along with different lighting conditions. We used this method to generate a comprehensive dataset of artificial images for additional analysis and testing.

The data was annotated using the popular COCO JSON format, making it simple to integrate with current tools for analysis and visualization. We gained valuable insights into the spatial and contextual relationships within the synthetic images by visualizing the JSON files. Conducting multiple simulations allowed us to gather a varied and inclusive dataset, which was beneficial for identifying the frequency of specific objects, like screws, in the dataset.

We conducted a comprehensive evaluation of the JSON files to assess the effectiveness of our synthetic image data generation method, gaining a clear understanding of its benefits and limitations. This study shows that BlenderProc has the potential to be a powerful tool for creating synthetic data. It also emphasizes the significance of well-designed simulations in generating strong and dependable datasets. Future research in this area may focus on improving generation methods and exploring additional applications of synthetic image data in various fields.

Acknowledgement

This research was funded by a grant from Universiti Malaysia Pahang for funding under grant RDU232707 and UIC231511.

References

[1] You, Hengxu, Yang Ye, Tianyu Zhou, Qi Zhu, and Jing Du. "Robot-enabled construction assembly with automated sequence planning based on ChatGPT: RoboGPT." *Buildings* 13, no. 7 (2023): 1772. https://doi.org/10.3390/buildings13071772

[2] Song, Rui, Fengming Li, Wei Quan, Xuting Yang, and Jie Zhao. "Skill learning for robotic assembly based on visual perspectives and force sensing." *Robotics and Autonomous Systems* 135 (2021): 103651. https://doi.org/10.1016/j.robot.2020.103651

[3] Sarker, Iqbal H. "AI-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems." *SN Computer Science* 3, no. 2 (2022): 158. https://doi.org/10.1007/s42979-022-01043-x

[4] Manakitsa, Nikoleta, George S. Maraslidis, Lazaros Moysis, and George F. Fragulis. "A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision." *Technologies* 12, no. 2 (2024): 15. https://doi.org/10.3390/technologies12020015

[5] Shatnawi, Hashem, and Mohammad N. Alqahtani. "Delving into the Revolutionary Impact of Artificial Intelligence on Mechanical Systems: A Review." *Semarak International Journal of Machine Learning* 1, no. 1 (2024): 31-40. https://doi.org/10.37934/sijml.1.1.3140

[6] Guo, Yanming, Yu Liu, Theodoros Georgiou, and Michael S. Lew. "A review of semantic segmentation using deep neural networks." *International journal of multimedia information retrieval* 7 (2018): 87-93. https://doi.org/10.1007/s13735-017-0141-z

[7] Ahmed, Shams Forruque, Md Sakib Bin Alam, Maruf Hassan, Mahtabin Rodela Rozbu, Taoseef Ishtiak, Nazifa Rafa, M. Mofijur, A. B. M. Shawkat Ali, and Amir H. Gandomi. "Deep learning modelling techniques: current progress, applications, advantages, and challenges." *Artificial Intelligence Review* 56, no. 11 (2023): 13521-13617. https://doi.org/10.1007/s10462-023-10466-8

[8] Nosrati, Hamed, and Masoud Nosrati. "Artificial intelligence in regenerative medicine: applications and implications." *Biomimetics* 8, no. 5 (2023): 442. https://doi.org/10.3390/biomimetics8050442

[9] Abdullah, Nashimah, Wan Nur Zafirah Wan Razak, Nur Aliya Ezzaty Azali, Khairun Nasrin Azman, Khairunnisa Mohd Kharfizi, Siti Nur Syakinah Jansi, Nurru Anida Ibrahim, Salisa Abdul Rahman, and Siti Norbakyah Jabar. "Electric Vehicle Adoption: A Comparative Analysis in Malaysia and ASEAN Countries." *Semarak International Journal of Electronic System Engineering* 1, no. 1 (2024): 60-68. https://doi.org/10.37934/sijese.1.1.6068

[10] Zhao, Zehui, Laith Alzubaidi, Jinglan Zhang, Ye Duan, and Yuantong Gu. "A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations." *Expert Systems with Applications* (2023): 122807. https://doi.org/10.1016/j.eswa.2023.122807

[11] Alzubaidi, Laith, Jinshuai Bai, Aiman Al-Sabaawi, Jose Santamaría, Ahmed Shihab Albahri, Bashar Sami Nayyef Al-dabbagh, Mohammed A. Fadhel *et al.,* "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications." *Journal of Big Data* 10, no. 1 (2023): 46. https://doi.org/10.1186/s40537-023-00727-2

[12] Elsayed, Abbas Al-Sadig Ahmed, Siti Nur Atiqah Halimi, and Mohd Azizi Abdul Rahman. "Image Detection and Distance Estimation for Rear Collision Avoidance using YOLOv4 on Jetson Nano." *Journal of Advanced Research in Computing and Applications*, 32, no.1 (2023): 22-29. https://doi.org/10.37934/arca.32.1.2229

[13] Dutta, Abhishek, and Andrew Zisserman. "The VIA annotation software for images, audio and video." In *Proceedings of the 27th ACM international conference on multimedia*, pp. 2276-2279. 2019. https://doi.org/10.1145/3343031.3350535

[14] Navarro-Guerrero, Nicolás, Sibel Toprak, Josip Josifovski, and Lorenzo Jamone. "Visuo-haptic object perception for robots: an overview." *Autonomous Robots* 47, no. 4 (2023): 377-403. https://doi.org/10.1007/s10514-023-10091-y

[15] Hashim, Mohd Ekram Alhafis, Nur Safinas Albakry, Wan Azani Mustafa, Banung Grahita, Miharaini Md Ghani, Hafizul Fahri Hanafi, Suraya Md Nasir, and Catherina ana Ugap. "Understanding the Impact of Animation Technology in Virtual Reality: A Systematic Literature Review." *International Journal of Computational Thinking and Data Science* 1, no. 1 (2024): 53-65. https://doi.org/10.37934/CTDS.1.1.5365

[16] Zhuang, Chungang, Shaofei Li, and Han Ding. "Instance segmentation based 6D pose estimation of industrial objects using point clouds for robotic bin-picking." *Robotics and Computer-Integrated Manufacturing* 82 (2023): 102541. https://doi.org/10.1016/j.rcim.2023.102541

[17] Ongbali, S. O., S. A. Afolalu, S. A. Oyedepo, A. K. Aworinde, and M. A. Fajobi. "A study on the factors causing bottleneck problems in the manufacturing industry using principal component analysis." *Heliyon* 7, no. 5 (2021). https://doi.org/10.1016/j.heliyon.2021.e07020

[18] Ji, Sanghoon, Sukhan Lee, Sujeong Yoo, Ilhong Suh, Inso Kwon, Frank C. Park, Sanghyoung Lee, and Hongseok Kim. "Learning-based automation of robotic assembly for smart manufacturing." *Proceedings of the IEEE* 109, no. 4 (2021): 423-440. https://doi.org/10.1109/JPROC.2021.3063154

[19] Zhang, Yaqian, Kai Ding, Jizhuang Hui, Jingxiang Lv, Xueliang Zhou, and Pai Zheng. "Human-object integrated assembly intention recognition for context-aware human-robot collaborative assembly." *Advanced Engineering Informatics* 54 (2022): 101792. https://doi.org/10.1016/j.aei.2022.101792

[20] Giuffrè, Mauro, and Dennis L. Shung. "Harnessing the power of synthetic data in healthcare: innovation, application, and privacy." *NPJ digital medicine* 6, no. 1 (2023): 186. https://doi.org/10.1038/s41746-023-00927-3

[21] Hong, Yeji, Somin Park, Hongjo Kim, and Hyoungkwan Kim. "Synthetic data generation using building information models." *Automation in Construction* 130 (2021): 103871. https://doi.org/10.1016/j.autcon.2021.103871

[22] Adam, Rebecca, Paulius Janciauskas, Thomas Ebel, and Jost Adam. "Synthetic training data generation and domain randomization for object detection in the formula student driverless framework." In *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, pp. 1-6. IEEE, 2022. https://doi.org/10.1109/ICECCME55909.2022.9987772

[23] Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft coco: Common objects in context." In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740-755. Springer International Publishing, 2014. https://doi.org/10.1007/978-3-319-10602-1_48

[24] Wang, Chaoming, Tianqiu Zhang, Xiaoyu Chen, Sichao He, Shangyang Li, and Si Wu. "BrainPy, a flexible, integrative, efficient, and extensible framework for general-purpose brain dynamics programming." *elife* 12 (2023): e86365. https://doi.org/10.7554/eLife.86365

[25] Ali, Sajid, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence." *Information fusion* 99 (2023): 101805. https://doi.org/10.1016/j.inffus.2023.101805

[26] Denninger, Maximilian, Dominik Winkelbauer, Martin Sundermeyer, Wout Boerdijk, Markus Wendelin Knauer, Klaus H. Strobl, Matthias Humt, and Rudolph Triebel. "Blenderproc2: A procedural pipeline for photorealistic rendering." *Journal of Open Source Software* 8, no. 82 (2023): 4901. https://doi.org/10.21105/joss.04901

[27] Srisathan, Wutthiya A., Chavis Ketkaew, and Phaninee Naruetharadhol. "Assessing the effectiveness of open innovation implementation strategies in the promotion of ambidextrous innovation in Thai small and medium-sized enterprises." *Journal of Innovation & Knowledge* 8, no. 4 (2023): 100418. https://doi.org/10.1016/j.jik.2023.100418

[28] López, Cefe. "Artificial intelligence and advanced materials." *Advanced Materials* 35, no. 23 (2023): 2208683. https://doi.org/10.1002/adma.202208683

[29] Cao, Hui, Rustem Zaydullin, Terrence Liao, Neil Gohaud, Eguono Obi, and Gilles Darche. "Adding gpu acceleration to an industrial cpu-based simulator, development strategy and results." In *SPE Reservoir Simulation Conference?*, p. D011S010R001. SPE, 2021. https://doi.org/10.2118/203936-MS

[30] Lu, Zhangji, and Lilong Cai. "Camera calibration method with focus-related intrinsic parameters based on the thin-lens model." *Optics Express* 28, no. 14 (2020): 20858-20878. https://doi.org/10.1364/OE.392731

[31] Huang, Sanao, Ke Xu, Ming Li, and Mingren Wu. "Improved visual inspection through 3D image reconstruction of defects based on the photometric stereo technique." *Sensors* 19, no. 22 (2019): 4970. https://doi.org/10.3390/s19224970