Recent Adv Sci Eng, Vol. 6, No. 1, pp. 1-7, June 2026

Recent Advances in Science and Engineering
Web page info: https://rase.yildiz.edu.tr
DOI: 10.14744/rase.2026.0001

Research Article

Simulator selection for robot training with reinforcement learning: A systematic comparison of sequential and parallel approaches

Ömer Mutlu Türk KAYA, Erkan USLU

Department of Computer Engineering, Yıldız Technical University, İstanbul, Türkiye

ARTICLE INFO

Article history
Received: 01 July 2025
Revised: 12 December 2025
Accepted: 23 March 2026

Key words:
Reinforcement learning, robot training, simulator selection.

ABSTRACT

Reinforcement Learning (RL) enables robots to acquire complex behaviors through trial-and-error interactions, placing simulators at the heart of scalable and efficient training pipelines. This study presents a qualitative comparison of simulators employed for robot training through RL, focusing on both sequential and parallel training architectures. While conventional simulators have advantages in terms of fidelity and sim-to-real, it is limited in terms of training time and generalization. On the other hand, modern simulators that allow parallel training offers significant improvements in terms of learning speed and data variety but also bring an advanced hardware cost. In this context, the study analyzes widely used simulators according to training architecture, training speed, parallel computation capacity, ROS compatibility, accessibility and scientific prevalence. Based on the analysis, a conceptual simulator selection tool was developed using a polychotomous key to assist researchers in identifying the most suitable simulation environment.

Cite this article as: Kaya ÖMT, Uslu E. Simulator selection for robot training with reinforcement learning: A systematic comparison of sequential and parallel approaches. Recent Adv Sci Eng 2026;6:1:1-7.

INTRODUCTION

Traditional methods and algorithms initially used to enable robots to perform various tasks yielded effective results in well-defined and predictable scenarios. However, with the increase in the number of moving obstacles and the variety of behaviors in the operational environment, the complexity and uncertainty of the environment has increased. Therefore, the state space that the robot interacts with is expanded and it has become almost impossible to predict all situations and to design a structure that takes action against every situation.

At this point, with the influence of advances in the field of artificial intelligence, learning-based methods that work with a large number of data have come to the fore in order to overcome the difficulties encountered. Among these methods, RL algorithms, which learn by trial and error by interacting with the environment, have gained an important place in the performance of complex robot tasks. RL allows a robot to learn the most appropriate behavioral policy by

***Corresponding author**
*E-mail adress: omer.kaya@yildiz.edu.tr

Published by Yıldız Technical University, İstanbul, Türkiye
This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

interacting with the environment and through reward-punishment feedback. Thus, the robot achieves flexibility and adaptability to cope with a wide variety of undefined situations without the need for human intervention.

However, since robots are mechatronic entities with hardware limitations, it is not feasible to carry out the trial and learning processes directly on physical robots in terms of both time and resources. For this reason, simulators that model real-world conditions have been developed for the trial and training of robots. In the early applications, since traditional algorithms were generally used, the developed simulators did not allow thousands of robots to work in simultaneous scenarios internally. Therefore, in the early simulators, RL-based robot training was mainly carried out in sequential training architecture. In this architecture, each scenario is initialized within the simulation environment, terminated upon completion of the training process and subsequently restarted for the next scenario following the same procedure. With this method, training processes can take hours or even days [1].

In order to overcome this bottleneck in terms of time, the idea of parallel simulation has gained great importance in RL. In this approach, many copies of the same robot are trained simultaneously in different scenarios. This both increases the diversity of training data and significantly shortens the learning time. So much so that training processes that used to take hours or days can be completed in minutes.

Existing robot simulation environments can be examined in two main groups in terms of RL training architecture. These are as follows:

- Traditional, sequential training-driven simulators
- Modern, parallel training-driven simulators

Both approaches have been addressed in different studies in literature. Numerous simulation tools have been compared in the context of robotics and RL in studies such as [1–4] and [5]. These comparisons were made according to features such as the libraries used, central processing unit (CPU) usage, real time factor (RTF), generalization among the physics engines, open/closed source status, visualization, file format, etc.

Although this study shares similarities with previous comparative works in terms of utilizing qualitative insights derived from the analysis of scientific sources, it offers originality through the distinct comparison criteria it employs. In addition, the information obtained as a result of the examinations and comparisons is interpreted and a conceptual simulator selection tool is proposed in the polychotomous key format for researchers and practitioners who carry out robot training via RL to choose the appropriate simulator. The Method section of the study explains how and under what headings this comparison was made and how the tool was developed. The information obtained was compared in the Results and Discussion section in terms of the effects of the architectural differences of the simulators on the training process. In addition, the proposed simulator selection tool was shared in that section. In the Conclusion section, a general evaluation of the study was stated.

MATERIALS AND METHODS

This study is grounded in a conceptual analysis of existing scientific literature and technical documentation. The study draws on commonly used seven simulators in the fields of robotics and RL, along with relevant scientific literature as source material. As a methodological approach, the information obtained from the selected sources was analyzed through a qualitative comparative framework.

The criteria determined for the qualitative comparison analysis are as follows:

- **Training Architecture –** The RL-based robot training paradigm supported by the simulator.
- **Training Speed** – The learning speed achieved based on the training architecture and the computational resources utilized.
- **Parallel Computation (CPU / GPU) –** The type of computational unit employed and the simulator's capability to support internal concurrent training.
- **ROS Compatibility –** Indicates whether the simulator includes a bridge or a software package that enables integration with the Robot Operating System (ROS). ROS is a widely adopted open-source middleware that provides a standardized communication infrastructure and toolset for building modular, scalable, and hardware-independent robotic systems. Due to its dominance in both academic and industrial research, compatibility with ROS significantly improves a simulator's usability, extendability, and potential for real-world deployment.
- **Scientific Prevalence** – Number of publications published in the last five years that explicitly refer to the simulator within the context of RL-based robotics, using disambiguated and simulator-specific Google Scholar search queries.
- **Accessibility –** The licensing model and availability status of the simulator, such as whether it is open-source, open-access, or commercially licensed.

The findings derived from this analysis were conceptually synthesized and a selection tool was developed using a polychotomous key to assist in identifying the most suitable simulator.

RESULTS AND DISCUSSION

Conventional robot simulators used for RL-based robot training typically employ the sequential training approach. Gazebo and Webots are the most common examples of this approach. These simulators provide high

fidelity with detailed physics modeling and sensor simulation capabilities [6]. They are widely preferred by robotics researchers not only for their compatibility with ROS but also for their ability to facilitate an easier transition between simulation and real-world deployment [6]. In sequential training architectures, simulation steps are executed in a single thread, allowing only one scenario to run at a time. If parallelization of training scenarios is desired, more than one simulator must be run at the same time and the information obtained from this must be combined outside the simulator. The fact that these simulators usually run on the CPU also limits the number of external parallelizations. In addition, simulators with sequential training architecture do not allow more than one agent to run and learn simultaneously within itself. The only exception here is PyBullet.

PyBullet is a CPU-based simulator and offers the opportunity to parallelize training within itself with the use of multiple threads [7]. However, since this is done on the CPU, it is limited by the number of threads.

Although the sequential approach is powerful in terms of real-time simulation and accuracy of complex interactions, it is slow in meeting the need for large-scale data in RL training [6]. For example, Gazebo uses intensive computational resources due to intensive physics calculations and is not suitable for parallel execution of large numbers of episodes [6]. This can cause the simulation to become a bottleneck in Deep Reinforcement Learning (DRL) training where hundreds of thousands of interactions are required. Therefore, the simulators running in sequential training architecture have high accuracy and integration advantages, but they are restrictive in terms of sample collection speed and scalability.

Unity is a hybrid software that offers both sequential and parallel training architectures. Although Unity is primarily designed for game development, it is also utilized in robot training processes involving RL [8]. Unity enables RL-based robot training in both sequential scenarios such as Gazebo and Webots and in parallel environments it creates within itself. Training process can be performed both on the graphics processing unit (GPU) and on the CPU. In this context, the RL models used in the Unity ML-Agents framework are run on a cross-platform inference library called Barracuda by default. Although Barracuda supports model execution on both the CPU and GPU, experiments conducted in [9] have shown that RL models work more efficiently on the CPU in most cases. The main reason for this is that the models used are usually not large enough to fully utilize the processing power of the GPU. In such cases, the time spent transferring the model to the GPU and getting the results back can be longer than the time it takes to run the model, reducing the overall efficiency of the system. In addition, the GPU being busy with graphical processing tasks at the same time can create a similar bottleneck by not being able to allocate enough resources for inference operations.

MATLAB & Simscape Multibody is another hybrid platform that enables RL-based robot training in both sequential and parallel architectures. Initially developed as a tool for matrix computations and algorithm development, MATLAB now also offers 3D simulation capabilities through a wide range of toolboxes and add-ons. This simulation capability was first introduced in 2002 under the name SimMechanics [10]. Subsequent improvements and updates were consolidated under the name Simscape Multibody. It is a commercial product [11] and, unlike other simulation tools, does not rely on open-source external physics engines. It uses MathWorks' own proprietary physics enginE. Although not as ROS-compatible as Gazebo and Webots, it offers a higher level of ROS integration compared to Unity and other simulation tools. For this purpose, it also provides a dedicated toolbox called 'ROS Toolbox' [12]. Although MATLAB & Simscape Multibody supports GPU acceleration for RL, this acceleration is primarily limited to policy and value function updates. Parallelization in it refers to multi-core CPU execution and GPU-accelerated learning, not GPU-based physics simulation. The physics simulation in Simscape Multibody is executed on the CPU, similar to Unity. Therefore, the overall training speed is highly dependent on the balance between simulation fidelity and learning architecture.

In order to overcome the shortcomings of sequential and hybrid approaches, advanced simulators have been developed that parallelizes both physics computations and RL processes end-to-end on the GPU, allowing thousands of environments to be run simultaneously. Thanks to this architecture, the RL agent collects data from thousands of different scenarios simultaneously and training time is significantly reduced. In fact, NVIDIA's Isaac Gym/Lab platform has achieved improvements in training speed by up to 2-3 orders of magnitude compared to traditional CPU-based approaches by performing all physics computation and policy learning in GPU memory [1]. Brax, another end-to-end parallel training simulator, claims to be able to train tasks similar to OpenAI Gym MuJoCo in minutes [13]. OpenAI Gym is a standardized toolkit for developing and comparing reinforcement learning algorithms. MuJoCo, often used within Gym environments, is a high-fidelity physics engine commonly preferred for continuous control tasks in robotics.

There is one newly emerging end-to-end GPU-based simulator with active development that is Genesis. It is an open-source physics simulation platform designed for robotics and embodied AI. Although the simulator was introduced to the public, it currently lacks a canonical peer-reviewed publication formally documenting its design choices and performance characteristics [14]. Based on an extensive literature survey, no peer-reviewed study

was found that systematically compares Genesis with the other simulators evaluated in this work. Although the developers report exceptionally high simulation throughput and claim superior performance compared to Isaac Lab and other GPU-accelerated simulators, these results are primarily based on self-prepared technical benchmarking reports [15]. At the time of writing, independent, peer-reviewed validation of these performance claims remains limited, and existing community-driven evaluations present mixed findings [16]. For these reasons, Genesis was included in the comparative overview to reflect emerging simulation trends, but it was intentionally excluded from the proposed simulator selection tool. The integration of Genesis into the selection framework is considered part of future work, pending further maturation of the platform and the availability of independent comparative studies.

Parallel simulation environments have emerged as a significant innovation in literature as they solve the problem of data collection with large-scale parallel sampling and accelerate RL algorithms. However, these approaches also have some limitations. High-end GPUs are required to handle thousands of real-time physics simulations – a GPU with at least 16 GB VRAM and 32 GB or more of system RAM is recommended to run Isaac Gym/Lab efficiently [17]. This may not always be available to every research group. The GPU-based simulator is usually special-purpose; integration into existing robotics software ecosystems such as ROS or adding new customized scenarios can be more challenging than their sequential counterparts. Indeed, Isaac Sim users have stated that using this platform requires a significant learning curve regarding the details of the game engine and physics library [18]. Therefore, parallel architectures offer a narrower scope in terms of flexibility and usability in exchange for their high performance.

The 'scientific prevalence' criterion refers to the number of publications published in the last five years that specifically refer to the simulator Gazebo [19], Webots [20], PyBullet [7], Unity [8], MATLAB & Simscape Multibody [11], Isaac Lab [1], Brax [13], and Genesis [14] in the RL-based robotics context. Therefore, the uniform, simulator-specific, and context-aware Google Scholar search queries were used during the measurement process. This method is reasonable in the criterion case because it is quantitative, comparative and reflects the actual usage within the research community. The used search queries are depicted in Table 1.

Overall, this comparison demonstrates that the interaction of RL and simulation architecture is a critical decision. In this context, the simulators discussed in the study are compared in Table 2 based on the characteristics such as training architecture, training speed, parallel computing capacity, ROS compatibility, scientific prevalence and accessibility.

Although the scientific prevalence measurement method does not guarantee the most realistic results, it shows the strong, consistent and quantitative outcomes. By using the queries, the scientific prevalence numbers in the table were determined according to the last five years on 01.01.2026.

There is one exceptional case which is Genesis. Due to the generic nature of the name "Genesis" and being new in the literature, reliable quantification of scientific prevalence is not currently feasible. Therefore, Genesis is treated as an emerging simulator and reported without a numeric prevalence value in Table 2.

While sequential simulation architecture is preferred in realistic physics applications, parallel architecture provides a great advantage for the faster training of DRL agents that learn with big data.

Our study, which evaluates these approaches examined in a scattered manner in the sources, is a guide for both researchers and practitioners for the selection of a suitable simulator. To accommodate different user preferences, the simulator selection tool—constructed using a polychotomous key—is presented in two alternative formats: a tree-based structure (Fig. 1) and a table-based structure (Table 3).

Table 1. The context-aware Google search queries for the scientific prevalence metric

No	Simulator	Search query
1	Gazebo	"Gazebo" AND robot* AND (simulation OR simulator) AND "reinforcement learning"
2	Webots	"Webots" AND robot* AND (simulation OR simulator) AND "reinforcement learning"
3	PyBullet	"PyBullet" AND robot* AND (simulation OR simulator) AND "reinforcement learning"
4	Unity	"Unity" AND robot* AND (simulation OR simulator OR "ML-Agents") AND "reinforcement learning"
5	MATLAB & Simscape Multibody	"Simscape Multibody" AND robot* AND (simulation OR simulator) AND "reinforcement learning"
6	Isaac Lab	("Isaac Lab" OR "Isaac Gym") AND robot* AND (simulation OR simulator) AND "reinforcement learning"
7	Brax	"Brax" AND robot* AND (simulation OR simulator) AND "reinforcement learning"
8	Genesis	"Genesis" AND robot* AND (simulation OR simulator) AND "reinforcement learning"

Table 2. Comparison of simulators used for RL-based robot training

No	Simulator	Training architecture	Training speed	Parallel computation (CPU / GPU)	ROS compatibility	Scientific prevalence	Accessibility
1	Gazebo	Sequential	Low	None	High	7340	Open source
2	Webots	Sequential	Low	None	High	1240	Open source
3	PyBullet	Sequential	Medium	Available (CPU only)	Medium	3920	Open source
4	Unity	Sequential/parallel	Medium	Available	Medium	8370	Open access
5	MATLAB & Simscape Multibody	Sequential/parallel	Medium	Available	High	277	Commercial
6	Isaac lab	Parallel	High	Available (GPU only)	Low	2480	Open source
7	Brax	Parallel	High	Available (GPU only)	None	551	Open source
8	Genesis	Parallel	High	Available (GPU only)	None	N/A (Emerging)	Open source

Table 3. Simulator selection tool constructed using a polychotomous key in a table-based structure

Step	Name	State	Result
1	ROS Compatibility	Undesired	Brax
		Desired	Go to Step 2
2	Hardware capability	High-performance GPU capable of handling physics and RL training end-to-end	Isaac Lab
		Moderate CPU and GPU available	Go to Step 3
		Only CPU available	Go to Step 4
3	Accessibility	A commercial license with technical support is preferred	Matlab & Simscape Multibody
		Open-access is preferred, with commercial license & technical support available if needed	Unity
4	RL training speed	Moderate training speed is prioritized; limited fidelity is acceptable	PyBullet
		Low training speed is acceptable, high fidelity is desired	Go to Step 5
5	Scientific prevalence	Preference for less-studied tools that offer room for wide novel research	Webots
		A widely-used platform with extensive documentation and example diversity is preferred	Gazebo

CONCLUSION

This study examines the simulators used in the context of robot training via RL based on sequential and parallel approaches and presents a qualitative comparison and a selection tool for choosing the appropriate simulator. Sequential training-compatible simulators ensure high fidelity and effective sim-to-real transfer but suffer from long training times and limited state space. On the other hand, simulators that offer parallel training can significantly reduce training time and enhance data diversity, enabling faster and more generalized learning; however, it may be limiting in terms of high hardware cost and low flexibility. The findings indicate that the choice of a simulator has a substantial impact on effective RL-based robot training and propose a selection tool to aid in identifying the most suitable simulator.

AUTHORSHIP CONTRIBUTIONS

Authors equally contributed to this work.

DATA AVAILABILITY STATEMENT

The published publication includes all graphics and data collected or developed during the study.

CONFLICT OF INTEREST

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

ETHICS

There are no ethical issues with the publication of this manuscript.

STATEMENT ON THE USE OF ARTIFICIAL INTEL-

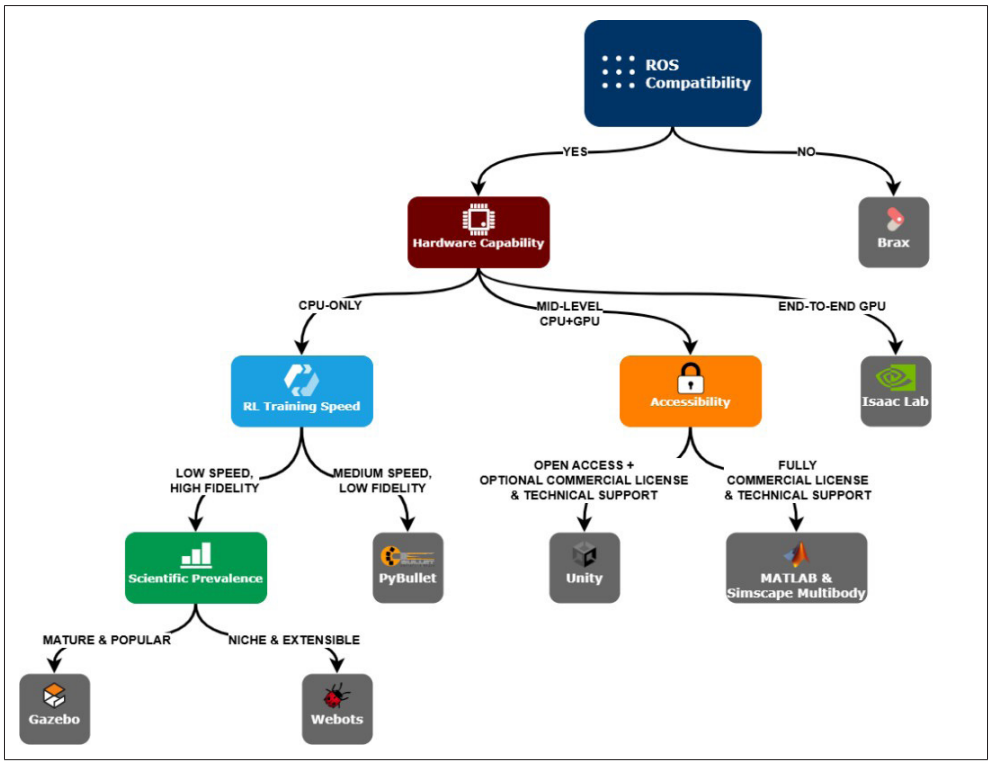


Figure 1. Simulator selection tool constructed using a polychotomous key in a tree-based structure.

LIGENCE

Artificial intelligence tools were used to assist in language editing and writing support. The scientific content, analysis, and conclusions were developed solely by the authors.

REFERENCES

[1] Makoviychuk V, Wawrzyniak L, Guo Y, Lu M, Storey K, Macklin M, et al. Isaac gym: High performance GPU-based physics simulation for robot learning. In: 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Track on Datasets and Benchmarks; 2021.

[2] Körber M, Lange J, Rediske S, Steinmann S, Glück R. Comparing popular simulation environments in the scope of robotics and reinforcement learning. arXiv; 2021. Preprint. doi: 10.48550/arXiv.2103.04616

[3] Kaup M, Wolff C, Hwang H, Mayer J, Bruni E. A review of nine physics engines for reinforcement learning research. arXiv:2407.08590 2024. Preprint. doi: 10.48550/arXiv.2407.08590

[4] Audonnet FP, Florent P. A systematic comparison of simulation software for robotic arm manipulation using ROS2. In: 2022 22nd International Conference on Control, Automation and Systems (ICCAS); Jeju, Korea, Republic of; 2022. [CrossRef]

[5] Valdenegro-Toro MP, Valdenegro-Toro M. Can reinforcement learning for continuous control generalize across physics engines? arXiv:2010.14444; 2020. Preprint. doi: 10.48550/arXiv.2010.14444

[6] Knowledgebase R. Robotics project guide – choose a simulator. 2025. Available at: https://roboticsknowledgebase.com/wiki/robotics-project-guide/choose-a-sim/#:~:text=Gazebo%20is%20a%20widely,world%20applications Accessed on Mar 24, 2026.

[7] Coumans E, Bai Y. PyBullet quickstart guide. 2016-2023. Available at: https://docs.google.com/document/d/10sXEhzFRSnvFcl3XxNGhnD4N2SedqwdAvK3dsihxVUA Accessed on Mar 24, 2026.

[8] Juliani A, Berges V-P, Teng E, Cohen A, Harper J, Elion C, Goy C, Gao Y, Henry H, Mattar M, Lange D. Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627; 2018. Preprint. doi: 10.48550/arXiv.1809.02627

[9] Mitsunami K. Part 3: Unity ML-Agents on Arm: How we created game AI. 2022 Dec 8. Available at: https://community.arm.com/arm-community-blogs/b/mobile-graphics-and-gaming-blog/posts/3-unity-ml-agents-on-arm-how-we-created-game-ai Accessed on Mar 24, 2026.

[10] Grepl R, Lee B, Singule V, Svejda P, Vlachý D, Zezula P. MBS modelling with SimMechanics: Case studies in research and education. In: MATLAB 2007 Conference; Prague, Czech Republic; 2007.

[11] The MathWorks I. Simscape Multibody. 1994-2025 The MathWorks, Inc. Available at: https://www.mathworks.com/products/simscape-multibody.html. Accessed on Jun 03, 2025.

[12] The MathWorks I. ROS toolbox. 1994-2025 The MathWorks, Inc. Available at: https://www.mathworks.com/products/ros.html. Accessed on Jun 03, 2025.

[13] Freeman CD, Frey E, Raichuk A, Girgin S, Mordatch I, Bachem O. Brax: A differentiable physics engine for large scale rigid body simulation. In: 35th Conference on Neural Information Processing Systems (NeurIPS 2021) – Track on Datasets and Benchmarks; 2021.

[14] Genesis Authors. A generative and universal physics engine for robotics and beyond. Dec 2024. Available at: https://github.com/Genesis-Embodied-AI/Genesis. Accessed on Jan 2, 2026.

[15] Xian Z. A detailed performance benchmarking report on Genesis. 2024. Available at: https://placid-walkover-0cc.notion.site/genesis-performance-benchmarking?pvs=4. Accessed on Jan 2, 2026.

[16] Tao S. Stone's Substack. 2024 Dec 21. Available at: https://stoneztao.substack.com/p/the-new-hyped-genesis-simulator-is. Accessed on Jan 3, 2026.

[17] Developers TILP. Local installation. 2025 Apr 11. Available at: https://isaac-sim.github.io/IsaacLab/main/source/setup/installation/index.html

[18] Sandberg D. 2024- Feedback request from Isaac Sim users. 2023 Sep. Available at: https://forums.developer.nvidia.com/t/2024-feedback-request-from-isaac-sim-users/267919#:~:text=The%20main%20challenge%20for%20me%2C,the%20easiest%20level%20of%20abstraction Accessed on Mar 24, 2026.

[19] Koenig N, Howard A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566); Sendai, Japan; 2004.

[20] Michel O. Cyberbotics Ltd. Webots™: Professional mobile robot simulation. Int J Adv Robot Syst 2004;1:39–42. [CrossRef]