Performance Comparison of Typical Physics Engines Using Robot Models With Multiple Joints

Chengye Liao, Yarong Wang, Xuda Ding, *Member, IEEE*, Yi Ren, *Member, IEEE*, Xiaoming Duan, and Jianping He, *Senior Member, IEEE*

***Abstract*—Physics engines are essential components in simulating complex robotic systems. The accuracy and computational speed of these engines are crucial for reliable real-time simulation. This letter comprehensively evaluates the performance of five common physics engines, i.e., ODE, Bullet, DART, MuJoCo, and PhysX, and provides guidance on their suitability for different scenarios. Specifically, we conduct three experiments using complex multi-joint robot models to test the stability, accuracy, and friction effectiveness. Instead of using simple implicit shapes, we use complete robot models that better reflect real-world scenarios. In addition, we conduct experiments under the default most suitable simulation environment configuration for each physics engine. Our results show that MujoCo performs best in linear stability, PhysX in angular stability, MuJoCo in accuracy, and DART in friction simulations.**

***Index Terms*—Simulation and animation, computer architecture for robotic and automation.**

I. Introduction

SIMULATIONS are of paramount importance in the validation of theoretical concepts and algorithms related to robotics. With the increasing popularity of learning based algorithms for robotics, the performance of simulators as experimental platforms has become a critical factor in evaluating the feasibility of related algorithms. Algorithms that have demonstrated good performance in simulators may fall short of expectations when being transferred to the real world, a phenomenon known as the reality gap [1]. The performance of simulators, and more specifically, the quality of the physics engine employed by the simulator, can directly impact the size of the reality gap. There have been some efforts to reduce the reality gap. Domain randomization [2] is a typical method used to train policies that can adapt to various dynamic environments. This method involves randomizing the dynamics of the simulator during the training process to enable the trained policies to generalize to a wide range of dynamic environments. Xie et al. [3] successfully trained a control policy that can be deployed in the real world by leveraging techniques such as the incorporation of stochastic noise, precise system identification, and appropriate action space selection. The goal is to achieve better control performance in both virtual and physical environments and reduce the reality gap. The two aforementioned methods are both concerned with the optimization of controllers, without attempting to optimize the dynamics of simulated robots.

Another way to reduce the reality gap is to select the most appropriate physics engine for the target scenarios during the simulation process [4]. Some well-known comprehensive virtual platforms, including Gazebo [5], UE5 [6] and V-REP [7], are widely used in robotics simulation. These virtual platforms have a complete toolchain, including a rendering module for displaying simulation effects, a communication module for realizing virtual-real communication, and a physics engine for physics calculation. Different physics engines have their own characteristics, and their performance may differ when being applied in various simulation scenarios. The purpose of this letter is to identify the simulation scenarios that different physics engines excel, enabling robotics researchers to select the appropriate physics engine according to their simulation needs. Many assessments of the performance of various physics engines are available in the literature, but few of them take the needs of robotic simulations into account. Also, most of the works on the evaluation of different physics engines mainly test the performance using some basic physical models in simplified scenarios [8]. In the work of Boeing [9], eight physics engines are tested using simple implicit shapes such as spheres and cubes made with several different materials. Roennau et al. [10] conduct experiments using seven performance metrics of interest in robotics, but all the shapes simulated in the experiments are still simple implicit shapes. Using these shapes alone for testing cannot fully reflect the capacity of a physics engine to simulate robots. In general, the shapes of robots and colliders in the environment are often complex and cannot be represented implicitly using mathematical formulas. We need to use mesh models to explicitly represent these shapes. Mesh models are three-dimensional shapes approximated by a series of triangular faces, and the angles between these faces are almost never

Manuscript received 3 May 2023; accepted 4 September 2023. Date of publication 27 September 2023; date of current version 6 October 2023. This letter was recommended for publication by Associate Editor R. Chandra and Editor A. Bera upon evaluation of the reviewers' comments. The work of Xiaoming Duan was supported in part by Shanghai Pujiang Program under Grant 22PJ1404900 and in part by the Natural Science Foundation of Shanghai under Grant 23ZR1428900. This work was supported by the CIE-Tencent Robotics X Rhino-Bird Focused Research Program. *(Corresponding author: Xiaoming Duan.)*

Chengye Liao, Yarong Wang, Xuda Ding, Xiaoming Duan, and Jianping He are with the Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: lil-kotyo@sjtu.edu.cn; wangyr16@sjtu.edu.cn; dingxuda@sjtu.edu.cn; xduan@sjtu.edu.cn; jphe@sjtu.edu.cn).

Yi Ren is with Tencent Robotics X Lab, Shenzhen 518000, China, and also with Advanced Manufacturing Lab, Huawei Technologies, Shenzhen 560037, China (e-mail: evanyren@tencent.com).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3320019, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3320019

smooth. This poses significant challenges for collision handling in physics engines.

Furthermore, few works have evaluated physics engines for the simulation of complex multi-joint robots such as wheeled robots and legged robots. Erez et al. [11] aim to develop general evaluation systems that can be used for the evaluation of physics engines such as ODE [12], Bullet [13], Havok [14], PhysX [15] and MuJoCo [16] with four different basic models including a robotic arm, a humanoid model, a planar kinematic chain and randomly-oriented capsules falling onto the floor, to test stability and accuracy, respectively. The robots used in [11] are still composed of simple implicit shapes, and their components are not physically connected. The displacement deviation per simulation step, the max grasp time, the energy and momentum conservation are used as the evaluation metrics. Meier et al. [4] assess different simulation physics engines in V-REP, including Bullet, ODE and Newton [17], to find the most suitable physics engine as the backend to reduce the reality gap of Kilobot simulation [18]. The only metric they employ is the distance between the simulation results and the target position. This distance metric hides a lot of information, making it difficult to analyze which part of the physics engine is causing the problem. Additionally, the introduction of control signals also makes it more difficult to analyze the cause of the deviation. The above works are more focused on establishing benchmarks while ignoring the various features and advantages of different physics engines. For instance, when a simulated robot has multiple joints, stability is often the most critical factor to consider since collisions between components connected to each active joint of the robot could lead to unexpected disturbing forces.

Due to the unique characteristics and optimal environment configurations of different physics engines, evaluating them based on traditional benchmark methods that use control variables may introduce biases and fail to fully capture their respective advantages. Therefore, a more comprehensive and nuanced evaluation approach is needed to provide a fair and accurate assessment of the performance of different physics engines in the context of multi-joint robots.

In this letter, our focus is on evaluating the performance of physics engines in the context of multi-joint robots while considering robot kinematics. To provide guidance on choosing physics engines that fit the needs of multi-joint robot simulation, we inspect three types of complex robot models on five commonly used open-source physics engines, i.e., ODE, Bullet, DART [19], MuJoCo, and PhysX in their respective most suitable configurations. As shown in Fig. 1, the three robot models adopted in this letter are a multi-joint mobile chassis robot with 69 parts and 68 joints (called Bigcar), a robotic arm with mobile chassis (called Bigarm), and a group of mobile chassis robots (called Bigcars). All robot models are equipped with universal wheels embedded with sixteen passive rollers and are much more complex than those used in the works mentioned earlier. Stability, accuracy, and friction effectiveness tests are conducted to evaluate the performance of different physics engines in simulating complex robot models. The contributions of this letter are summarized as follows:

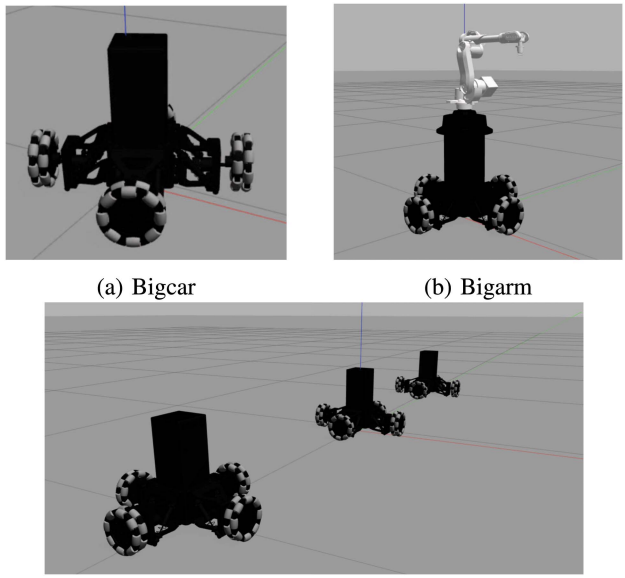


Fig. 1. Complex multi-joint robot models.

- To our knowledge, this is the first work on analyzing the advantages and disadvantages of each mainstream physics engine in the context of multi-joint robot simulations. We consider robot dynamics in the test scenarios and provide detailed guidance on choosing the suitable physics engine for robotics researchers.
- We propose new evaluation metrics for stability, accuracy, and friction effectiveness to fully explore the characteristics of different physics engines.

This article is organized as follows. We introduce the common virtual simulation platforms in Section II. Section III gives the evaluation methodology of three test experiments on five simulation platforms. In Section IV, we provide the results for each simulation test and analyze each virtual simulation platform. Finally, Section V summarizes the results of the experiments and concludes the paper.

II. Simulation Platforms

Gazebo is one of the most popular integrated physics engine platforms. Gazebo integrates four physics engines: ODE, Bullet, DART, and Simbody [20]. ODE is the default physics engine of Gazebo and is also commonly used in robot simulations. Gazebo provides the convenience of switching between the four physics engines, high-quality rendering, and a convenient graphical programming interface, and we use it as one of the simulation platforms in this letter to test the performance of ODE and DART. We do not test Simbody because the multi-joint mobile chassis robot used for testing does not fit Simbody well in Gazebo. PyBullet [21] is chosen as the simulation platform for Bullet, which is a fast and easy-to-use Python module for robotics simulation and machine learning. As for PhysX, we select UE5 as a carrier. UE5 is a game engine with rich functions and an

active development community, and it brings stable simulation and good visual effects to researchers. MuJoCo is a physics engine designed to facilitate research and development in the fields of robotics, biomechanics, graphics, and animation [22]. It is the first simulator designed for model optimization, and it is specifically optimized for the contact between objects. Unlike the game engine UE5, MuJoCo prioritizes accuracy over stability. MuJoCo uses the convex Gauss Principle [23] to account for contact forces and has a uniquely-defined inverse dynamics facilitating data analysis and control applications. In addition, the engine is quite flexible and provides multiple parameters that can be tuned to approximate a wide range of contact phenomena.

These simulation platforms are widely used in games, digital twins, and various simulations, with active community support and detailed usage documentation.

III. Evaluation Methodology

The validity of the evaluation method is the key to ensuring the reliability of the test results. Therefore, it is very important to determine a set of scientifically valid and feasible evaluation methods. We design three typical experiments to test the stability, accuracy, and friction effectiveness of physics engines for simulating complex multi-joint robots.

A. Stability

There are two aspects to the stability of robot systems in simulations: the static stability of the robot reflects its stability in its natural state, while the dynamic stability reflects whether the robot can generate stable motion when control signals are applied. In this letter, we mainly focus on the static stability, which is affected by the collision handling module of the physics engine. As multi-joint robots often consist of a large number of components, collisions between components may cause spontaneous impulsive forces within the robot. Furthermore, the robot mesh is always in contact with the ground, and if there is penetration between the robot and the ground, it can also cause abnormal impulses. Both of these can make the robots experience unstable shaking in its natural state. In this letter, we study the static stability of the robot, including linear stability and angular stability. For linear stability, we load the complex multi-joint robots on the ground in the five physics engines, without applying any external force other than gravity, and then measure the position drift d_{linear} of the robots defined in (1) in the simulation environment,

$$d_{\text{linear}} = \sqrt{d_{\text{x}}^2 + d_{\text{y}}^2 + d_{\text{z}}^2}, \tag{1}$$

where d_{x}, d_{y}, and d_{z} are the offsets in three coordinate directions, respectively. The mean and variance of position drift data are used as metrics to evaluate the linear stability of different physics engines. As for angular stability, we record the swaying angular velocity of the robots, and define d_{angular} in (2) as the metric for the angular stability of different physics engines,

$$d_{\text{angular}} = \sqrt{\omega_{\text{pitch}}^2 + \omega_{\text{roll}}^2}, \tag{2}$$

where ω_{pitch} and ω_{roll} are the pitch and roll angular velocities of the robot, respectively. Since these angular velocities would tip the robot over, the pitch and roll angular velocities should always be zero ideally.

B. Accuracy

Accuracy is the difference between the desired behavior and what the robot actually does. The multi-joint robots are more complex in motion calculation than simple implicit shapes, and they are well suited for testing the accuracy of different physics engines. We choose to apply constant speed control to the four wheels of the multi-joint robots so that they can drive along the straight line of $y = -x$ in theory, and record their real-time positions. The standard deviation of their deviation from $y = -x$ are used as the performance measures of the accuracy of each physics engine. The position drifts d_{accuracy} is defined as follows

$$d_{\text{accuracy}} = |x + y|, \tag{3}$$

where x, y are the coordinates of the robot. In order to evaluate the accuracy more comprehensively, we also evaluate the accuracy of the physics engine under the conditions of different simulation step sizes. Adjusting the size of the simulation step can affect the solver's calculation speed and solution accuracy.

C. Friction Effectiveness

Friction is one of the indispensable environmental factors in robot simulations. In order to evaluate the friction simulation effect of the five physics engines, we place a 3-meter high smooth slope on the ground of each physics engine and release the multi-joint robots at the top of the slope from a stationary initial state. After the robots leave the slope, they will continue to slide forward on the ground and finally stop due to the friction between the wheels and the ground. We record the sliding distance L_{des} of the robots after leaving the slope and the initial speeds of the robots when they leave the slope, and then compare the sliding distance of the robots in the simulation with the theoretical sliding distances L calculated by (4) according to the law of conservation of energy,

$$L = \frac{v^2}{2g\mu}, \tag{4}$$

where v is the initial speed of the robots when they leave the slope, g is the gravitational acceleration and μ is the friction factor between the wheels and the ground. Then, the distance deviation d_{friction} is defined as

$$d_{\text{friction}} = \frac{|L_{\text{des}} - L|}{L}. \tag{5}$$

The three experiments described above examine the performance of different physics engines from different aspects, which can serve as basic guidelines for choosing appropriate robot physics engines. The testing results are presented in the next section.

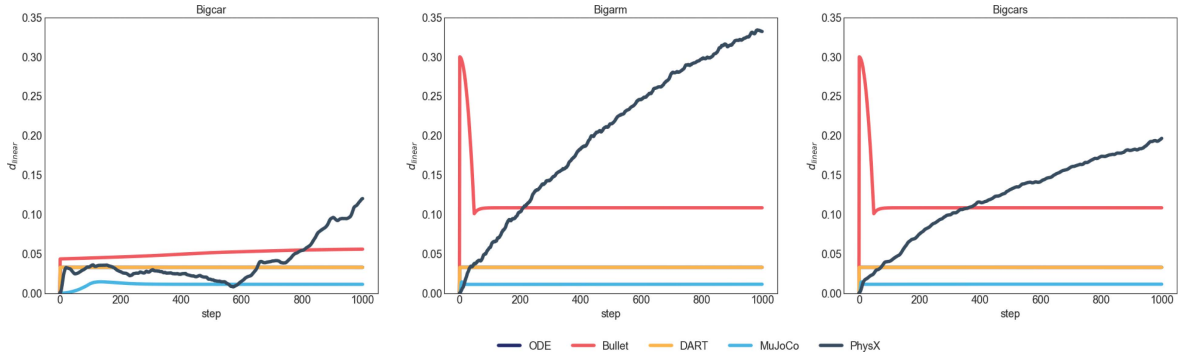


Fig. 2. Drifting displacement d_{linear} curves of three models.

TABLE I
MEANS AND STANDARD DEVIATIONS OF THE DRIFTING DISPLACEMENT CURVES

Robot Model	Physics engine	Mean	Standard deviation
Bigcar	ODE	0.0328	2.7172E-06
	Bullet	0.0505	4.0097E-03
	DART	0.0328	2.3891E-05
	MuJoCo	**0.0108**	2.6541E-03
	PhysX	0.0396	2.6624E-02
Bigarm	ODE	0.0328	4.5469E-06
	Bullet	0.1144	2.9919E-02
	DART	0.0328	4.4620E-06
	MuJoCo	**0.0112**	6.4799E-04
	PhysX	0.2016	9.4334E-02
Bigcars	ODE	0.0328	3.8098E-06
	Bullet	0.1144	2.9919E-02
	DART	0.0327	4.8083E-05
	MuJoCo	**0.0113**	6.5375E-04
	PhysX	0.1222	5.0730E-02

TABLE II
MEANS OF THE d_{ANGULAR} WITH BIGCAR AT THREE DIFFERENT SIMULATION STEP SIZES

Physics engine	Mean		
	step size-0.1x	step size-1x	step size-10x
ODE	0.2609	0.2672	0.9302
Bullet	0.8128	0.3030	2.2450
DART	0.1801	0.5059	0.7253
MuJoCo	0.08240	0.0800	0.0850
PhysX	0.3089	0.3361	0.8406

IV. SIMULATION RESULTS

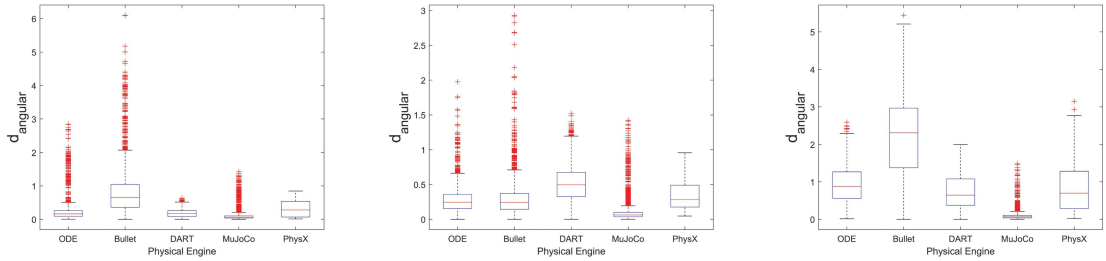
All experiments in this article are performed on a standard PC with a 2.30 GHz Intel i7-10875H CPU, a 16 GB RAM and an NVIDIA GeForce RTX 2060. It is worth noting that the experiments are performed with the default most suitable configuration for each physics engine, and the semi-implicit Euler integration method is used in all physics engines, except for experiments that require modifications of the simulation step size or the friction coefficient configuration.

A. Stability

In the experiment of linear stability, we place the robot models at the coordinate origin of each physics engine and record the position drifts of the robots for 1000 frames of the simulation under the default applicable configuration of each physics engine. The drifting displacement curves of the robots are shown in Fig. 2. The means and standard deviations of the five curves are shown in Table I. The drift results of DART and ODE are very similar, so their curves almost overlap in Fig. 2. In terms of the linear stability, MuJoCo is the best among the five physics engines, followed by DART and ODE, then Bullet, and finally PhysX.

In the second simulation experiment, we apply constant speed control to drive the robot along the straight line of $y = -x$. The pitch and roll angular velocities can be obtained by simply measuring the angular velocity of Bigcar with respect to the x and y axes. Then one can calculate the angular stability using (2). When the simulation step size is increased by a factor of ten, the robot models shake violently when they move in ODE, DART, and Bullet. When the simulation step size is reduced by ten times, the robot models shake in Bullet. The abnormal shaking caused the robot to displace about two meters before stopping, resulting in a huge deviation between the actual trajectory and the ideal trajectory. As for MuJoCo, it can only perform various simulation tests on the robots when the simulation step size is less than 0.0002. Otherwise, the robots will be unstable. While ODE, Bullet, and DART are all run in the default configuration with a simulation step size close to 0.001, the simulation step size of PhysX is larger, close to 0.05.

The angular stability d_{angular} experienced by the Bigcar at three different simulation step sizes, 0.1 times the default simulation step size, the default simulation step size and 10 times the default simulation step size, are shown in Fig. 3. The means of d_{angular} is shown in Table II. According to Fig. 3, all four other physics engines show more outliers except for PhysX. In PhysX, the mean and the maximum of d_{angular} are relatively low and the angular stability is the best, while in Bullet, the mean and the maximum of d_{angular} are highest and the angular stability is the worst. In summary, the angular stability of PhysX is the



(a) Simulation step size-0.1x (b) Simulation step size-1x (c) Simulation step size-10x

Fig. 3. d_{angular} with Bigcar at three different simulation step sizes.

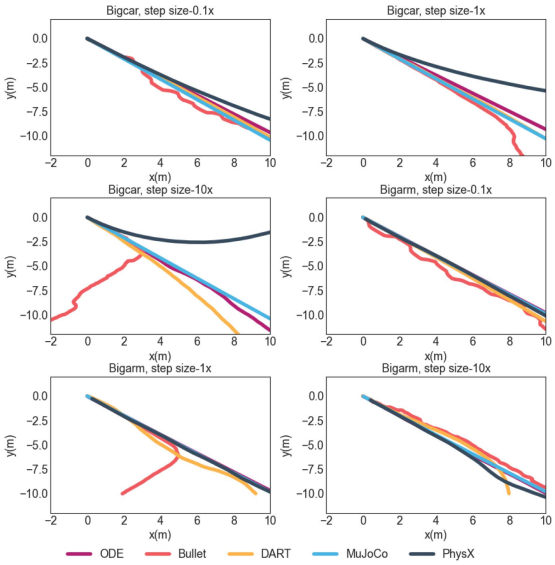


Fig. 4. Trajectories of Bigcar and Bigarm at different simulation step sizes.

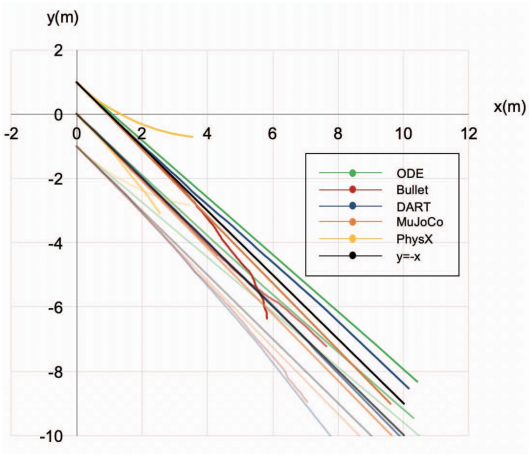


Fig. 5. Trajectories of Bigcars at the default simulation step size.

best, followed by MuJoCo, then ODE and DART, and finally Bullet.

B. Accuracy

The experiments on accuracy with Bigcar and Bigarm are conducted at three different simulation step sizes, 0.1 times the default guideline step size, the default simulation step size and 10 times the default simulation step size. With other configurations fixed, the larger the simulation step size, the faster the simulation speed of the physics engines. We apply a constant torque to each wheel of the robots so that the robots travel along the straight line in the $-\frac{\pi}{4}$ direction from the coordinate origin ideally. We record the trajectories of the Bigcar and Bigarm until the coordinates of the robot in the x or y direction become 10 meters as shown in Fig. 4. Table III shows the means and standard deviations of the difference between the trajectories of the robots and the $y = -x$ line in different physics engines using different simulation step sizes. For Bigcars, we perform the experiment at the default simulation step sizes and record the trajectories of three robots, which are represented by lines with different opacity in Fig. 5. From Table III, we can observe that MuJoCo has the highest accuracy, followed by DART, then ODE, then Bullet and PhysX.

To compare the accuracy of the physics engines under the same setting, we also conduct experiments with a fixed simulation step size on Bigcar. Due to the upper limit on the simulation step size imposed by MuJoCo, we choose a step size of $1/64$ milliseconds, consistent with the initial step size used in the work of [11]. From the last column of Table III, it can be seen that when all physics engines are simulated with a fixed time step of $1/64$ milliseconds, MuJoCo exhibits the highest accuracy, followed by Bullet, DART, and ODE, with PhysX showing the poorest performance.

In general, the accuracy of a physics engine increases as the simulation step size decreases. However, PhysX presents an exception to this trend. When the step size decreases to $1/64$ milliseconds (less than 0.1 times the default step), the error increases significantly instead of decreasing. Moreover, the accuracy of MuJoCo varies little with the change of the simulation step size. The accuracy of DART is good overall but fluctuates from model to model. The performance of ODE is also average. Robots in DART and ODE jitter when the simulation step size increases. As for Bullet, its stability is so poor that the robots jitter violently at both small and large simulation step sizes. When the simulation step size is reduced, the accuracy tests cannot even be completed. In the simulation test with Bigcars, only two robots are stable, and the third one suffers from a numerical explosion and is not shown in the figure.

TABLE III
MEANS AND STANDARD DEVIATIONS OF THE DIFFERENCE BETWEEN THE TRAJECTORIES OF THE ROBOTS AND $y = -X$ IN DIFFERENT PHYSICS ENGINES AT DIFFERENT SIMULATION STEP SIZES

Robot model	Physics engine	Step size-0.1x		Step size-1x		Step size-10x		Step size-1/64 ms	
		mean value	standard deviation	mean value	standard deviation	mean value	standard deviation	mean value	standard deviation
Bigcar	ODE	0.0404	0.0643	0.3128	0.2105	0.5579	0.4350	-0.0252	**0.1518**
	Bullet	0.1059	0.0940	0.5070	0.4294	-4.0660	3.9191	0.0450	0.7601
	DART	**-0.0058**	**0.0067**	**-0.0714**	**0.0529**	-1.1284	0.8116	-0.0222	0.1819
	MuJoCo	-0.2130	0.1280	-0.1585	0.0823	**-0.1976**	**0.1149**	**0.0193**	0.1989
	PhysX	0.7297	0.8797	1.8448	1.6348	3.0190	2.6507	1.7025	1.8538
Bigarm	ODE	0.0281	0.0377	0.1129	0.1030	**-0.0172**	**0.0318**	-	-
	Bullet	-0.1312	-0.1312	-1.8897	2.4147	0.6292	0.2866	-	-
	DART	**-0.0201**	0.0901	-0.4229	0.4361	0.0846	0.6147	-	-
	MuJoCo	0.0384	**0.0315**	0.0618	**0.0523**	0.0585	0.0588	-	-
	PhysX	-0.0338	0.0594	**-0.0290**	0.0656	-0.2987	0.3150	-	-
Bigcars	ODE	-	-	0.4913	0.3200	-	-	-	-
	Bullet	-	-	0.2608	0.2664	-	-	-	-
	DART	-	-	**0.1481**	0.1383	-	-	-	-
	MuJoCo	-	-	0.1918	**0.1126**	-	-	-	-
	PhysX	-	-	0.5350	0.4212	-	-	-	-

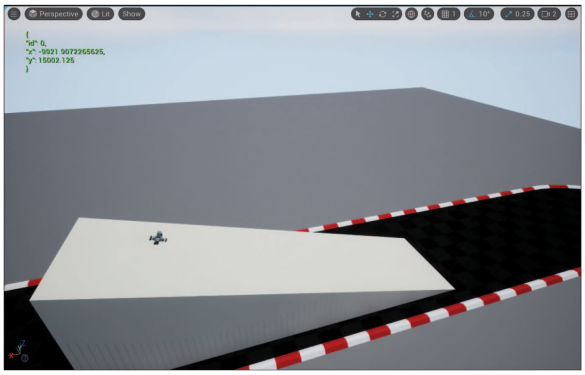


Fig. 6. In the experimental environment of the friction effectiveness tests, the robot's wheel drive shaft will be fixed, and it will fall vertically from the slope.

TABLE IV
SLIDING DISTANCE DEVIATION RATE OF THE ROBOT UNDER EACH DIFFERENT FRICTION FACTOR SETTING

Friction coefficient		0.1	0.3	0.5	0.7	1.0
d_{friction}	ODE	0.1515	**0.1856**	0.2514	0.3798	0.3185
	Bullet	0.1435	0.2774	0.3857	0.3920	0.3257
	DART	**0.1430**	0.1955	**0.1500**	**0.2905**	**0.3162**
	MuJoCo	0.7151	0.6911	0.2117	4.1781	1.1978
	PhysX	0.7860	0.0680	0.3958	0.8228	1.0933

C. Friction Effectiveness

The last experiment is about the effectiveness of friction configuration in various physics engines. We place a smooth slope at the origin of the ground coordinates of each physics engine as in Fig. 6. The slope is 3 meters high and 10 meters long in the horizontal direction. The Bigcar slides down from the slope from a fixed position and orientation. After reaching the bottom edge of the slope, the robot gains a speed and continues to slide on the ground until the friction between Bigcar and the ground makes Bigcar stop completely. Different from the earlier two experiments, here we fix the wheels of the mobile chassis robot so that the only friction force between the wheels and the ground is the sliding friction, which makes it convenient for us to calculate the sliding distance theoretically through the initial speed of the Bigcar at the bottom of the slope. By comparing the theoretical sliding distance with the actual sliding distance recorded in the experiment, we obtain the sliding distance deviation rate d_{friction} of the robot under each different friction factor setting. The performance of each physics engine under different friction factor configurations are reported in Table IV, and we conclude that DART is the best, followed by ODE, then Bullet, then PhysX, and finally MuJoCo.

V. CONCLUSION AND DISCUSSIONS

In this letter, we test the stability, accuracy and friction effectiveness of five physics engines in their default configuration using three complex multi-joint robot models. We design three experiments to evaluate the three extremely important requirements for robot simulations and obtain the ranking of each physics engine under different requirements. Although the three metrics proposed in this letter cannot cover all aspects that need to be considered in the simulation of complex multi-joint robots, our evaluations can be used as guidelines for selecting the most suitable physics engines based on specific simulation requirements.

- When the robot in the experimental scenario needs to frequently interact with objects in the environment, such as contact between omnidirectional wheels and the ground, stability should be the primary consideration.
- When the simulation experiments involve dynamics that are heavily affected by friction forces, the friction effectiveness will be more important.

TABLE V
SUMMARY OF THE TEST RESULTS

Rank	Linear stability	Angular stability	Accuracy	Friction effectiveness
1	MuJoCo	PhysX	MuJoCo	DART
2	DART	MuJoCo	DART	ODE
3	ODE	ODE	ODE	Bullet
4	Bullet	DART	Bullet	PhysX
5	PhysX	Bullet	PhysX	MuJoCo

- When an algorithm needs to be transferred from the simulation environment to the real world, using a physics engine with high accuracy for training will be more effective.

The score rankings for the three metrics are summarized in Table V. DART and MuJoCo are very good choices when dealing with scenarios that require high simulation accuracy, while MuJoCo outperforms DART in terms of stability. If the smoothness of the simulation or the stability of the simulation is more important in the robot simulation, then PhysX is the best choice (but it also has lower accuracy than other physics engines). When the effectiveness of the friction force becomes important, DART, ODE and Bullet are recommended because the overall friction bias exhibited in these three physics engines are comparatively smaller.

Roennau et al. [10] examined three physics engines (Bullet, PhysX, and ODE) using robotic models composed of simple implicit shapes, and their results are not entirely consistent with ours. In their experiments, Bullet demonstrates strong capabilities in handling friction and sliding, while it has mediocre performance in our tests. PhysX exhibits superior performance in collision detection in both studies, which indicates that PhysX provides enhanced stability in collision handling for both mesh and simple implicit shape based robotic models. For ODE, it displays average performance across all tests conducted by Roennau et al. However, in our experiments, ODE exhibits better performance over Bullet and PhysX in most aspects, which may suggest that ODE is better suitable for simulating mesh models.

REFERENCES

[1] N. Jakobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Proc. Eur. Conf. Artif. Life*, 1995, pp. 704–720.

[2] X. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3803–3810.

[3] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Proc. Conf. Robot Learn.*, 2019, pp. 317–329.

[4] A. Meier, S. Carroccio, R. Dornberger, and T. Hanne, "Discussing the reality gap by comparing physics engines in Kilobot simulations," *J. Robot. Control*, vol. 2, no. 5, pp. 441–447, 2021.

[5] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, pp. 2149–2154.

[6] E. Prado, M. Delgado, N. García-Muñoz, B. Monclús, and C. Navarro, "General-television programming in Europe (UE5): Public versus commercial channels," *El profesional de la información*, vol. 29, no. 2, 2020, Art. no. e290204.

[7] E. Rohmer, S. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2013, pp. 1321–1326.

[8] J. Hummel, R. Wolff, T. Stein, A. Gerndt, and T. Kuhlen, "An evaluation of open source physics engines for use in virtual reality assembly simulations," in *Proc. Int. Symp. Vis. Comput.*, 2012, pp. 346–357.

[9] A. Boeing and T. Bräunl, "Evaluation of real-time physics simulation systems," in *Proc. Int. Conf. Comput. Graph. Interactive Techn. Aust. Southeast Asia*, 2007, pp. 281–288.

[10] A. Roennau, F. Sutter, G. Heppner, J. Oberlaender, and R. Dillmann, "Evaluation of physics engines for robotic simulations with a special focus on the dynamics of walking robots," in *Proc. Int. Conf. Adv. Robot.*, 2013, pp. 1–7.

[11] T. Erez, Y. Tassa, and E. Todorov, "Simulation tools for model-based robotics: Comparison of bullet, havok, MuJoCo, ODE and PhysX," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4397–4404.

[12] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending open dynamics engine for robotics simulation," in *Proc. Int. Conf. Simulation, Modeling, Program. Auton. Robots*, 2010, pp. 38–50.

[13] H. He, J. Zheng, Q. Sun, and Z. Li, "Simulation of realistic particles with bullet physics engine," in *Proc. E3S Web Conf.*, 2019, vol. 92, Art. no. 14004.

[14] "Havok Physics Engine," [Online]. Available: www.havok.com

[15] A. Maciel, T. Halic, Z. Lu, L. P. Nedel, and S. De, "Using the PhysX engine for physics-based virtual surgery with force feedback," *Int. J. Med. Robot. Comput. Assist. Surg.*, vol. 5, no. 3, pp. 341–353, 2009.

[16] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.

[17] "Newton dynamics about Newton," [Online]. Available: http://newtondynamics.com/forum/newton.php

[18] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 3293–3298.

[19] "DART physics engine," [Online]. Available: http://dartsim.github.io

[20] M. A. Sherman, A. Seth, and S. L. Delp, "Simbody: Multibody dynamics for biomedical research," *Procedia Iutam*, vol. 2, pp. 241–261, 2011.

[21] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games," *PyBullet*, 2016.

[22] "MuJoCo physics engine," [Online]. Available: www.mujoco.org

[23] H. Bruyninckx and O. Khatib, "Gauss' principle and the dynamics of redundant and constrained manipulators," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2000, pp. 2563–2568.