Journal of Robotics and Control (JRC)
Volume 2, Issue 5, September 2021
ISSN: 2715-5072 DOI: 10.18196/jrc.25120

Discussing the Reality Gap by Comparing Physics Engines in Kilobot Simulations

Andreas Meier[1], Sascha Carroccio[2], Rolf Dornberger[3], Thomas Hanne[4]*
[1, 2, 3, 4] School of Business, Institute for Information Systems
University of Applied Sciences and Arts Northwestern Switzerland
Olten/Basel, Switzerland
Email: [1] andreas.meier@students.fhnw.ch, [2] sascha.carroccio@students.fhnw.ch, [3] rolf.dornberger@fhnw.ch,
[4] thomas.hanne@fhnw.ch, *Corresponding Author

***Abstract*—The difference between real experiments and their simulated counterparts, the so-called reality gap, is dependent on various factors and a challenging issue for every simulation-based robot experiment. The reality gap in robot experiments is often caused by software, which is not always able to sufficiently capture particular details and process them properly. In order to minimize this difference, this paper strives to assess different simulation physics engines in V-REP, the simulation framework for the Kilobot swarm robots. The outcome of the simulation software adaption is based on a unified multi robot experiment applied to Kilobots. This paper proposes a simulation attitude, which reflects the outcomes of a real world experiment with Kilobots accurately.**

Keywords—Kilobot, physics engines, reality gap, simulation, swarm robotic, V-REP

I. INTRODUCTION

Increasing interaction quality of robots with their environment based on automatically adapting their behavior and skills is an emerging topic in the domain of robotics [1]. Over the last years, several researchers published numerous papers proposing various optimizations of robot behaviors considering different reactive learning frameworks and self-restricting adaption in the operating environment [2]. Simulation of robotic behavior does not only enable performing a considerable number of training cycles without endangering or adapting the required physical environment, it also creates the opportunity of an enhanced reinforcement learning process in order to improve the robots' performance. A simulation is therefore helpful in order to transfer the learnings into reality.

Despite all these relevant advantages, simulation have still the restriction of not being able to fully capture the real environment a robot is finally supposed to act in. For example, there are many real factors, such as collision of robots, mass or traction [3], which might have an influence on the simulated outcomes, but, still are not, or only insufficiently, considered by the simulation software. On the other hand, there is simulation software available with the capabilities to simulate real world environments in a more precise way. Nevertheless, such software requires a high amount of computational power, which leads to hours or even days of processing time for just a few seconds of simulation [3]. In the field of robotics, according to Jakobi, it is hard to model a simulation accurately and appropriately to the applied physical environment, which becomes even more difficult with increasing complexity and the number of included robots [4]. A lot of different features and environmental circumstances can influence the robots' behavior in a real world scenario: sensor and communication behavior, noise, physical characteristics and hardware calibration.

In the specific case of the Kilobots' experiment according to Zhong et al. [5], the orbiting functionality, as example, was not noticeably performed by the robots in the real experiment as it was in the simulation. This Kilobot experiment (including simulation and real experiment) indicates a reality gap of swarm robotic behavior in the transmission of simulated training policy and real world test scenario. Christiano et al. refer to the reality gap when the main goal is accurate simulation [3]. However, in order to gain a relevant amount of training data for the robots, software simulations depict an attractive alternative to design and enhance algorithms evaluating different solution approaches [6].

II. RELATED WORK

The reality gap in robotics simulation has been studied frequently in the literature and dates back at least to the 1990s [7]. In the following we discuss some approaches for analyzing and reducing the reality gap including a few selected application areas

In [8] a concept of noise induction is used for modelling the reality gap and artificial evolution is used to develop recurrent dynamical network controllers for the simulated robot. The resulting controllers were used in a real robot leading to an almost identical robot behavior. In [9], [10], and [11] the gap between reality and simulation denoted as transferability is used an optimization criterion in a multiobjective optimization approach. The other considered objective is the control goal of the robot denoted as fitness. The problem is solved by a multiobjective evolutionary algorithm. In [12] an illusory sub-system is incorporated in the robot control for adaptation and minimizing the differences of robot behavior evaluations in reality and in simulations without explicit calibration. Authors in [13] suggest an anticipation-enabled control architecture which builds a partial model of the simulated environment within robot control. In reality, then error estimation of the model is done which is then used for improving the control.

In [14] and [15] abstraction of the sensory inputs and motor actions (i.e. preprocessed sensory inputs) is successfully used for reducing the reality gap. In recent years the applications of machine learning approaches have been used successfully for reducing the reality gap. For instance, neural networks are used in [16] and [17].

In [18] a set of small obstacles is added to the simulated environment which is shown to lead to more robust robot movements when transferred to a real world robot, especially in difficult scenarios. In [19] the reality gap is considered in the context of an evolutionary robotics approach for robots used to control the growth and motion of natural plants. The reality gap in the context of robotics and virtual or mixed reality applications is discussed in [20] and [21].

III. Problem Description

The need to effectively analyze various outcomes of potential robot behavior has led researches to apply simulated robot experiments instead of conducting real world tests [6]. Having the advantage of lower costs and faster results than real world experiments, robot simulation software is available for different platforms or distributions, for example MATLAB or SD/FAST [22]. Based on the numerous availabilities of simulation tools, it appears inviting to assume that the gained results for robot behavior in simulators perform equally in the real world [6]. Although, simulated training data seem to replicate and optimize the main functionality of the robots in an appropriate manner, the simulation software fails to consider several adjustments based on the real environment the robot is operating in [3]. For example, Christiano et al. [3] list different factors, such as exact measuring or friction. Such differences regarding the performance of robots, detected between the simulation and the applied experiment, is regarded as reality gap [1].

Zhong et al. refer to the reality gap in connection with a particular Kilobot experiment and its simulation applying noisy circumstances [5]. When several swarm robots are organizing themselves at the same time, the accuracy can be considerably decreased due to communication failures and collision issues between the robots. Considering the results with the Kilobot experiment and the comparison between simulation and real world experiment, shown in [5], several reality gap issues were identified. For example, the exploring tendencies of the Kilobots were more noticeable in the real implementation than in the simulations, conducted with the simulation framework V-REP. Another reality gap was hinted by the absent orbiting-a-fellow functionality. Although this function was running in the simulations, in the real implementation it was not noticeable, which lead to a reduced success rate of finding the target in the real experiment. An explanation for this might be that the real Kilobots are technically only able to process one infrared (IR) signal (the applied communication technology) at a time, while the simulation software does not use this limitation. Apparently, the arising time delay in communication between real Kilobots could not be considered in the simulations. In consequence, the simulations proposed that more Kilobots would reach the target than the real implementation revealed [5].

The problem, discussed in this paper, is defined as follows: The reality gap (see Fig. 1) is an inevitable disturbance factor when producing training data for robots in simulators [1]. Consequently, there are several intentions to minimize the reality gap as much as possible. Building upon the performed experiment of Zhong et al. with the Kilobots, our paper evaluates alternative settings within the simulation software by recreating and reprocessing the exact same experiment and its attributes with different physics engines. The recorded outcomes of each simulator run will be evaluated and assessed similarly as in the experiments of Zhong et al. The goal therefore is to suggest a more suitable physics engine setting for multi robot experiments in V-REP with Kilobots in order to reflect the outcomes of the real world experiment more precisely. For this reason, the relating tests, analysis and assumptions of Zhong et al. are taken as basic concept for this paper. Therefore, the assessment of different simulated outcomes is based on the comparison of the documented outcomes with V-REP and the results of the real implementation of Zhong et al. Based on that, the reality gap severity of each physics engine is determined and discussed. Recommendations are given in the conclusions.

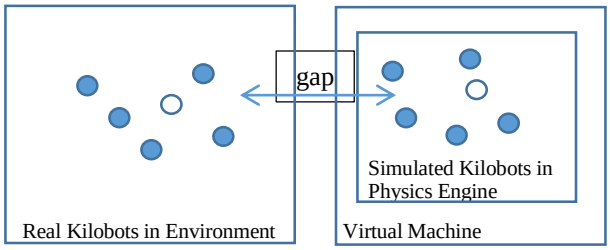

Fig. 1. The problem of the reality gap between real robot experiment and simulation of robots using physics engine.

IV. Solution Method

The reality gap between simulated outcomes and the real world experiment depends on various factors: The simulation software, computational power, which includes memory and CPU cores as well as the ability of using multiple CPU cores [23], and physics engines used within the simulation software [24, 25]. Often, a distinction is made between open source and commercial [26], which relates to both software and physics engines. It should be noted that professional software is not necessarily commercial. Pitonakova [23] compared three different simulators V-REP, Gazebo and ARGoS in regards to built-in capabilities like available physics engines, scene editor and mesh manipulations, robot libraries and other models, programming methods, and user interfaces [25]. The analysis has shown that V-REP, even though it is very complex and resource-intensive, it is the most comprehensive simulation software compared to the others. In terms of features, Gazebo is closer to V-REP than ARGoS. For example, ARGoS has no Kilobot libraries compared to V-REP.

V-REP is a multifunctional application for different robot simulation scenarios. It does not only contain an implemented Kilobot model, but also several other robot models and optimization methods, for example the Hexapod model or the Delta Arm manipulator [27]. V-REP as a framework is a standard to simulate robotic behavior. However, the variety of

possible settings inside the software, considering the simulation performance, makes its correct use very complex.

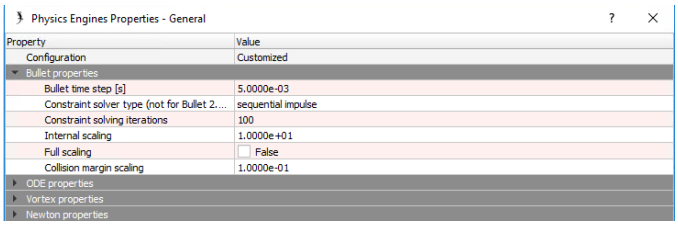

Fig. 2. Physics engines overview in V-REP [28]

The motion of rigid bodies in the physical world is increasingly often modelled and simulated by using physics engines [25]. The simulation software V-REP supports four different physics engines: Bullet, ODE, Vortex and Newton (see Fig. 2). Each physics engine has predefined standard values for e.g. static or dynamic properties, material and shape. It is possible to adjust the default parameters (e.g. friction, restitution and damping [27]) in the simulation software (see Fig. 3).

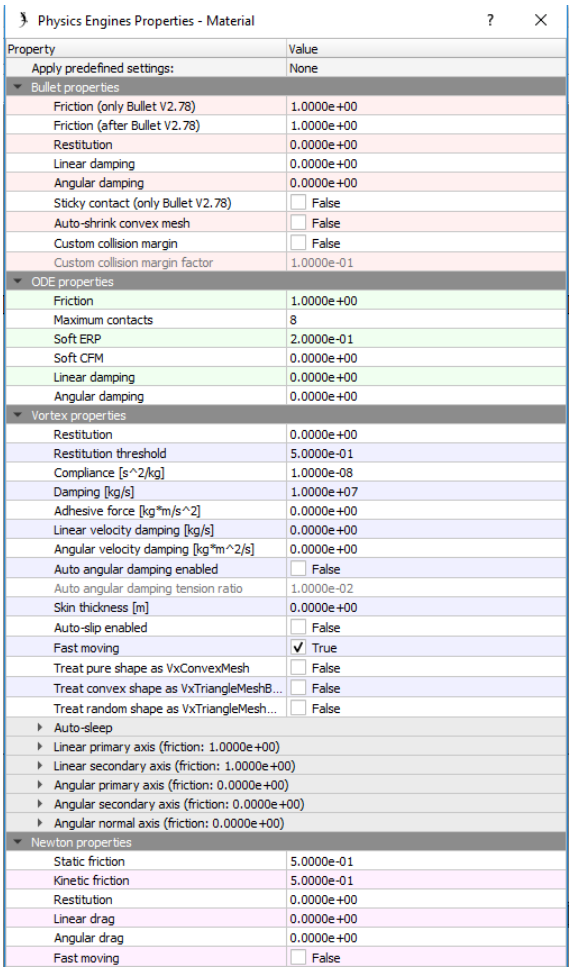

Fig. 3. Physics engines material properties [28]

Default settings are documented and can be found on the producer's website. Simulations, performed with different physics engines or customized parameters, will lead to dissimilar results. Therefore, the adjustment of the parameters per simulation needs to be done carefully in order to simulate a real behavior correctly.

A circumstance that has a major impact on such experiments is the collision behavior of two or more robots, which itself is a widely studied field [26]. The behavior of a collision in real world experiments is based on physical behavior of the objects, which is influenced by various factors such as their kinematics or the structure of their surfaces. In simulations, the collision behavior is based on algorithms [26]. As it is illustrated in the results by Zhong et al., none of the Kilobot searchers tipped over during the real world test runs as predicted in the simulation [5]. This is resulting in a huge reality gap, because a searching robot that has tipped over has no further chance to reach the target Kilobot. In order to reduce this reality gap, the parameters for the collision behavior can be adjusted in V-REP via the physics engines material properties (see Fig. 3). The question is, how?

Zhong et al. compared one particular real world experiment against two V-REP simulations [5]. The only parameter, which differed between the two simulations, was the desired distance for reaching the target. In one simulation, the value was set to 4 cm, and in the other simulation, the value was set to 6 cm. The value in the real world experiment was given with 6 cm. In order to compare the results from the experiments, the same value of 6 cm was used as configuration for the experiments with the different physics engines. Thus, our simulations were carried out and documented analogous to the procedure of Zhong et al. [5].

The simulation test runs were recorded using the integrated video recorder [30]. The properties can be adjusted in the simulation software, which are for example launching the recording at the next simulation start or define the location for the generated video files.

The simulations have been performed on a virtualized computing environment running Acropolis Hypervisor (AHV) version 20170830.166. The used virtual machine on the hypervisor was a Windows 10 Enterprise edition with 8 GB Memory and 4 CPU cores with Intel Xeon E5 2.1 GHz processor.

The simulations were carried out with V-REP PRO EDU V3.6.1 using the three physics engines Bullet, ODE and Newton. Simulation with the Vortex physics engine were not performed as the Vortex Studio Essentials software was no longer available.

V. RESULTS

Simulation I: Table I shows the test results from the simulation carried out with the physics engine Bullet 2.83 in regards to the number of searchers close to the target. Throughout the test runs, on average of 2.6 Kilobots of 10 Kilobots in total reached the target within the simulation time. In test run three, none of the searcher reached the target. Four searchers, which represents the highest number of searchers, reached the target within all test runs.

In the simulation, regarding the time the searcher needed to reach the target for the first time, the time span is between 31 and 41 seconds (see Fig. 4). The time span, a further searcher needed to reach the target is 32 and 46 seconds. In test run three, five and nine no further searcher reached the target.

Concerning collisions, an average of 2.8 Kilobots tipped over during the simulation. In the worst case, seven Kilobots tipped over in one test run.

Fig. 5, which represents the pictures taken at the end of the test runs, shows a similar situation for all test runs. Right at simulation start, some of the searchers did not move towards the target. This caused some of the searchers to stay quite far away from the target after the end of the test runs.

After 15 to 20 seconds on average, the first searcher detected another searcher, which was indicated by the white color. An exception is the first test run, where it took 30 seconds to let the first two Kilobots detect each other.

Simulation II: Table II shows the test results from the simulation carried out with the physics engine ODE in regards to the number of searchers close to the target. On average 2.3 Kilobots reached the target within the simulation time. In all test runs, a minimum of one Kilobot reached the target. In test run five, six and ten, only one searcher reached the target, which represents the lowest number of searchers reached the target. Four searchers reached the target in test run two and nine, which represents the highest number of searchers reached the target.

With respect to the time the searcher needed to reach the target for the first time, the time span is between 32 and 45 seconds (see Fig. 6). The time span, a further searcher needed to reach the target is 35 and 53 seconds. In three out of the ten test runs no further searcher reached the target.

TABLE I. NUMBER OF SEARCHERS CLOSE TO THE TARGET IN SIMULATION (BULLET 2.83)

Test run	Number of searchers close to the target				
	Green: target found	*Yellow: approach -ing target*	*White: Other searchers detected, still searching*	*Cyan: No message received, keep searching*	*Total*
1	3				3
2	3			2	5
3					0
4	2	1		1	4
5	1		2		3
6	3	1		2	6
7	3			1	4
8	3		2		5
9	1		1	1	3
10	4			1	5
Average	2,6				3,8
Median	3				4

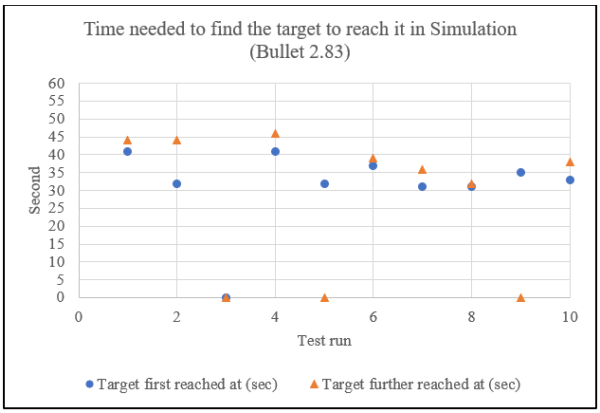

Fig. 4. Time needed to find the target and to reach it in simulation (Bullet 2.83)

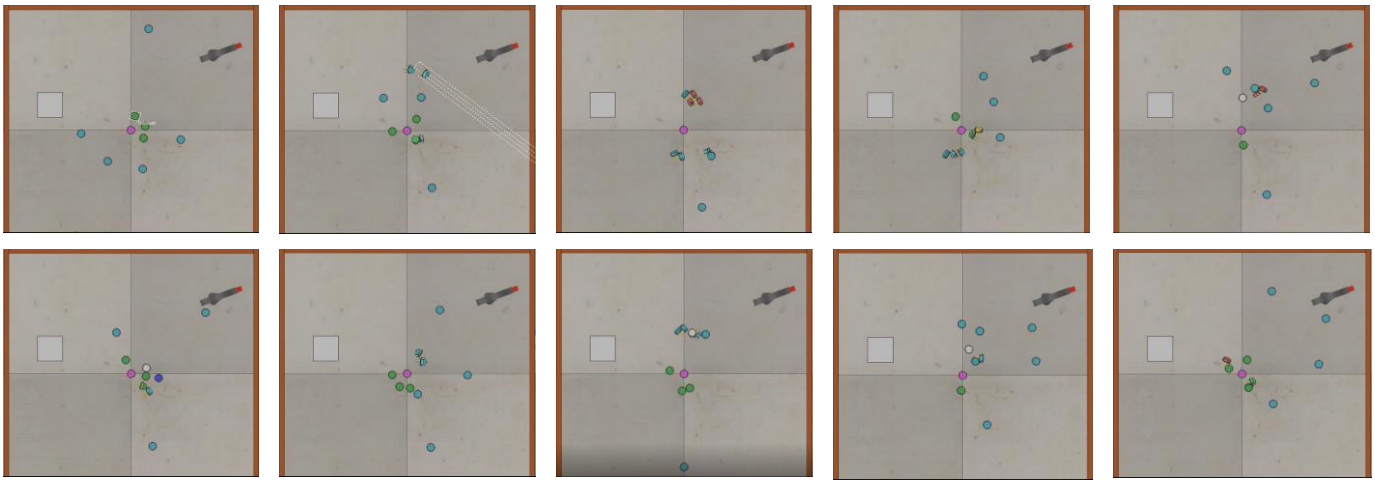

Fig. 5. Simulation results with Bullet 2.83

TABLE II. NUMBER OF SEARCHERS CLOSE TO THE TARGET IN SIMULATION (ODE)

Test run	Number of searchers close to the target				
	Green: target found	*Yellow: approach -ing target*	*White: Other searchers detected, still searching*	*Cyan: No message received, keep searching*	*Total*
1	2			1	3
2	4				4
3	2			1	3
4	3		1		4
5	1			1	2
6	1		1		2
7	2			1	3
8	3				3
9	4			1	5
10	1		1	1	3
Average	2,3				3,2
Median	2				3

Concerning collisions, an average of 3.1 Kilobots tipped over during the simulation. In one particular test run, even the target Kilobot tipped over due collision.

Fig. 7, which represents the pictures taken at the end of the test runs, shows a similar situation for all test runs. Right at simulation start, some of the searchers did not move towards the target. This caused some of the searchers to stay quite far away from the target after the end of the test runs.

After 15 to 29 seconds on average, the first searcher detected another searcher, which was indicated by the white color.

Simulation III: As shown in Fig. 8, in the simulation, none of the searchers reached the target in the test runs using the Newton physics engine.

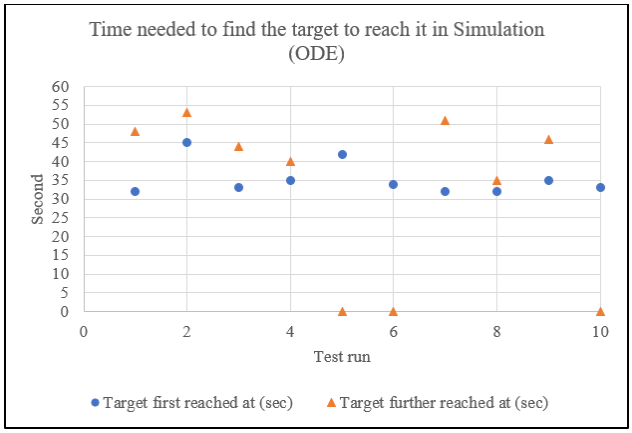

Fig. 6. Time needed to find the target and to reach it in Simulation (ODE)

Except in the first test run, at least one searcher has approached the target, however spending more than 60 seconds as upper limit, (see Fig. 8).. As indicated by the red Kilobots, some of the searchers came very close to another one in every of test runs. However, the simulation predicted that none of the searchers tipped over due to collision.

At simulation start, the searchers walked towards the target in all test runs. During the simulation, some of the searchers have changed the direction and moved away from the target (see Fig. 9).

Simulation IV: As an outlook, V-REP provides the Vortex physics engine, too. However, no simulations could be carried out, because the license was not available to us during our experiment.

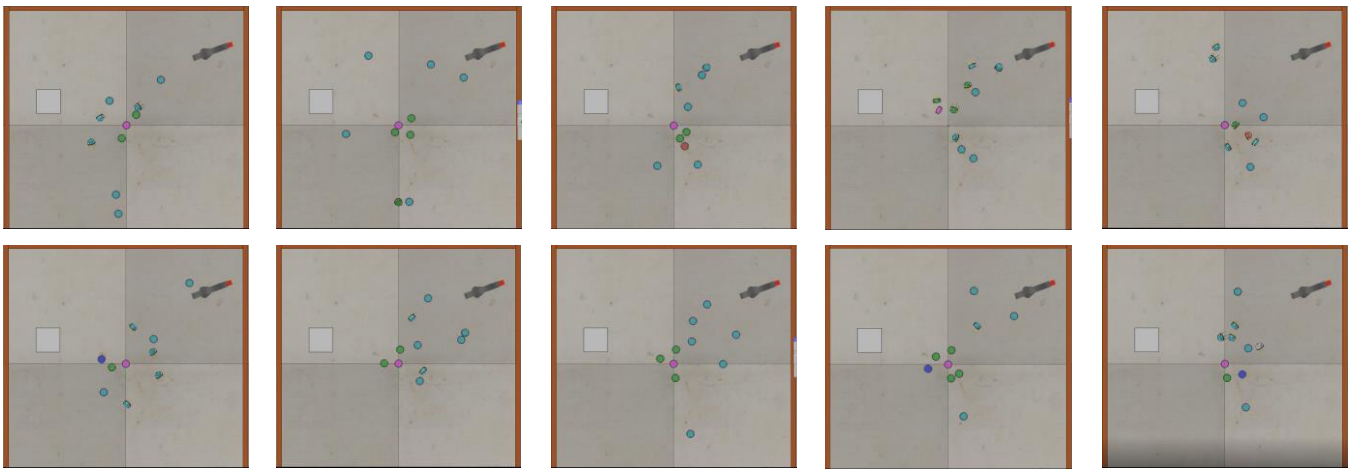

Fig. 7. Simulation results with ODE

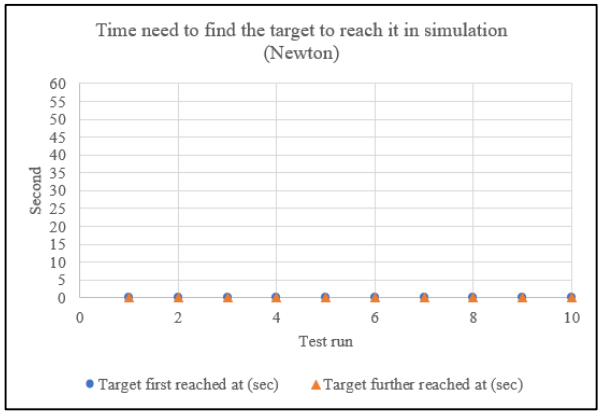

Fig. 8. Time needed to find the target and to reach it in simulation (Newton)

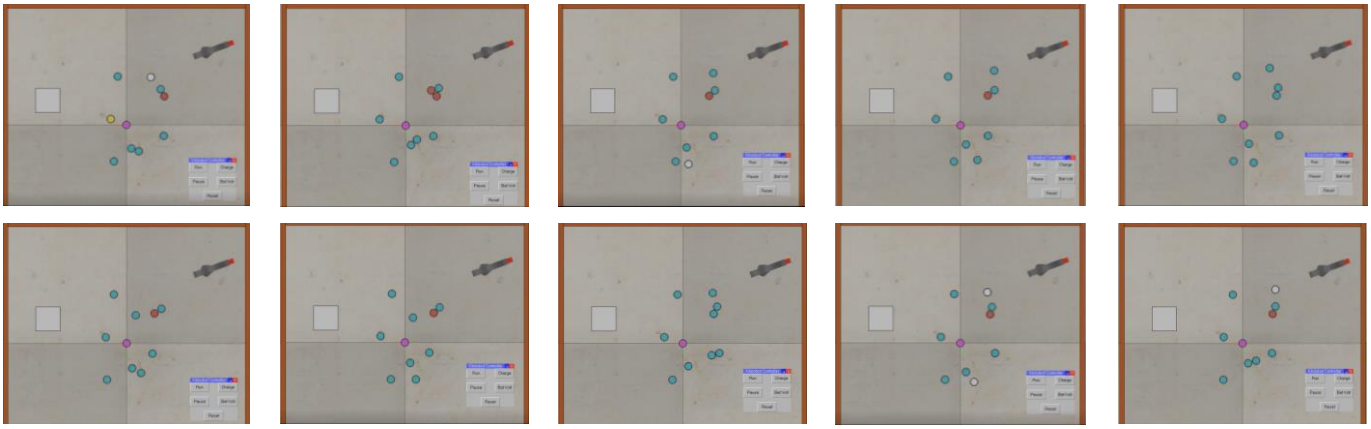

Fig. 9. Results of simulation with Newton

VI. Discussion and Analysis

As the experiments with the different physics engines have shown, there are potentially better choices than the applied standard physics engine Bullet 2.78 in the documented experiments by Zhong et al. The repeated experiments indicated that the ODE physics engine provided the most reliable results according to the real implementation with Kilobots. As considerable reasons for this suggestion is, on one hand, the average number of successful (shown in green color) searching Kilobots with ODE of 2.3. In comparison with the average value in the real implementation with 2.2, ODE reports the closest result among all tested physics engines towards the real experiment. Furthermore, as the video evaluation in the test runs with ODE revealed, the Kilobot searchers tend to move more randomly in the test field. This led to a higher distance between the searcher bots and the target bot at the end of every test run. This was also observed by Zhong et al. by performing the real implementation. In comparison to the other physics engines, using ODE, the searching Kilobots had a certain tendency to head directly toward the target Kilobot. This might be an explanation for the higher average number of Kilobots, which have found the target.

During the different simulation runs, it became noticeable that some physics engines predicted Kilobots falling over after a collision with a fellow searching bot. For example, Bullet 2.83 and ODE simulated such a behavior. Newton, in fact simulated collisions between searching Kilobots, but never a fell over of a robot. Unfortunately, there were no specific cases documented for the real experiment to draw a conclusion out of it. A more elaborated test scenario in this direction would therefore provide further insights, as well as a larger number of test runs with the different physics engines could discover specific tendencies. Additionally, the experiments with Newton indicated that none of the searching robots really reached the target. As there are no test results in the real implementation confirming this outcome, the reason for the results of the tests with the Newton physics engine should be examined in further research.

VII. Conclusion and Outlook

The tested three different physics engines Bullet, ODE and Newton in V-REP provide a broad range of attributes to simulate the behavior of Kilobots. As Fig. 3 shows, the physics engines partly use similar types of attributes. For, example both Bullet versions and ODE contain attributes called friction, linear damping and angular damping. However, other attributes are dissimilar between different physics engines, for example soft ERP or soft CFM in the ODE physics engine.

Another interesting observation is the same physics engine in different versions, such as the two mentioned Bullet versions, can have almost similar attributes, but the simulated outcomes appear very different. The Bullet 2.78 simulation, conducted by Zhong et al., has average number 4.6 Kilobots that have found the target. In comparison, the Bullet 2.83 experiment of this paper has an average number of 2.6 Kilobots that found the target using the same parameters. The reason for these different results is hidden in the physics models implemented in the software and stays unclear. One explanation for this issue is probably that the software provides continuous improvements of the included models without sufficiently informing the users about respective changes. As consequence, even a comparison of simulations using the same physics engine in different versions might lead to different results. However, the outcomes of Bullet 2.83 are the second closest outcomes compared to the real experiment after ODE results.

As a further aspect, the ODE physics engines comprises, together with the Newton physics engine, the lowest number of attributes and none of them are similar among both physics engines. In the context of the ODE physics engine, it might be interesting to learn, which specific attributes or combination of attributes are capable to create the similar results as in the real implementation.

Overall, the results show that the physics engines are difficult to compare, because they comprise many different parameters, which are difficult to adjust to the real experiment. Particularly, the correct simulation of collisions of Kilobots is difficult to compute: a small change in the initial conditions might result in a very different number of fallen robots or robots reaching the target.

The different physics engines experiments were conducted on a virtualized Windows 10 client on Acropolis Hypervisor, which is Linux-based. A further research, relying on the elaborated results, might be to check if the simulation software runs platform-independent. Repeating tests on different computing platform using the same physics engines in the same versions might if the CPU or other hardware specifications have an influence in the results.

REFERENCES

[1] J. Zagal, J. Ruiz-del-Solar and P. Vallejos, "Back to reality: Crossing the reality gap in evolutionary robotics", IFAC Proceedings Volumes, vol. 37, no. 8, pp. 834-839, 2004.

[2] J-P. Mouret and K. Chatzilygeroudis, “20 Years of Reality Gap: a few Thoughts about Simulators in Evolutionary Robotics”, GECCO ’17 Companion, Berlin, Germany, July 15-19, 2017, 4 pages

[3] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel and W. Zaremba "Transfer from simulation to real world through learning deep inverse dynamics model”, arXiv preprint arXiv:1610.03518, 2016

[4] N. Jakobi, Ph. Husbands and I. Harvey, “Noise and the reality gap: The use of simulation in evolutionary robotics”, In European Conference on Artificial Life, Springer, Berlin, Heidelberg, 704–720, 1995

[5] V. J. Zhong, R. Dornberger and Th. Hanne, "Comparison of the behavior of swarm robots with their computer simulations applying target-searching algorithms", International Journal of Mechanical Engineering and Robotics Research, 7(5). 2018

[6] S. Koos, J. Mouret and S. Doncieux, "The transferability approach: crossing the reality gap in evolutionary robotics", IEEE Transactions on Evolutionary Computation, vol. 17, no. 1, pp. 122-145, 2013

[7] J. C. Bongard, “Evolutionary robotics”, Communications of the ACM, 56(8), 74-83, 2013.

[8] N. Jakobi, P.Husbands, and I. Harvey, “Noise and the reality gap: The use of simulation in evolutionary robotics”, In European Conference on Artificial Life (pp. 704-720). Springer, Berlin, Heidelberg, 1995.

[9] S. Koos, J. B. Mouret and S. Doncieux, “Crossing the reality gap in evolutionary robotics by promoting transferable controllers”, In Proceedings of the 12th annual conference on Genetic and evolutionary computation (pp. 119-126), 2010.

[10] S. Koos, J. B. Mouret and S. Doncieux, “The transferability approach: Crossing the reality gap in evolutionary robotics”, IEEE Transactions on Evolutionary Computation, 17(1), 122-145, 2012.

[11] J. B. Mouret, S. Koos and S. Doncieux,”Crossing the reality gap: a short introduction to the transferability approach”, arXiv preprint arXiv:1307.1870, 2013.

[12] J. C. Zagal and J. Ruiz-Del-Solar, “Combining simulation and reality in evolutionary robotics. Journal of Intelligent and Robotic Systems, 50(1), 19-39, 2007.

[13] C. Hartland and N. Bredeche, “Evolutionary robotics, anticipation and the reality gap”, In 2006 IEEE International Conference on Robotics and Biomimetics (pp. 1640-1645). IEEE, 2006.

[14] K. Y. Scheper and G. C. de Croon, “Abstraction, sensory-motor coordination, and the reality gap in evolutionary robotics”, Artificial Life, 23(2), 124-141, 2017.

[15] K. Y Scheper and G. C. de Croon, ”Abstraction as a mechanism to cross the reality gap in evolutionary robotics”, In International Conference on Simulation of Adaptive Behavior (pp. 280-292). Springer, Cham, 2016.

[16] N. Cruz and J. Ruiz-del-Solar, “Closing the simulation-to-reality gap using generative neural networks: Training object detectors for soccer robotics in simulation as a case study”, In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE, 2020.

[17] F. Golemo, A. A. Taiga, A. Courville and P. Y. Oudeyer, “Sim-to-real transfer with neural-augmented robot simulation”, In Conference on Robot Learning (pp. 817-828), 2018.

[18] K. Glette, A. L. Johnsen and E. Samuelsen, ”Filling the reality gap: Using obstacles to promote robust gaits in evolutionary robotics”, In 2014 IEEE International Conference on Evolvable Systems (pp. 181-186). IEEE, 2014.

[19] M. Wahby, D. N. Hofstadler, M. K. Heinrich, P. Zahadat and H. Hamann, “An evolutionary robotics approach to the control of plant growth and motion: Modeling plants and crossing the reality gap”, In 2016 IEEE 10th International Conference on Self-Adaptive and Self-Organizing Systems (SASO) (pp. 21-30). IEEE, 2016.

[20] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas and N. Ayanian, “Mixed reality for robotics”, In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 5382-5387). IEEE, 2015.

[21] E. Freund and J. Rossmann, “Projective virtual reality: Bridging the gap between virtual reality and robotics”, IEEE transactions on robotics and automation, 15(3), 411-422, 1999.

[22] T. Erez, Y. Tassa and E. Todorov, “Simulation tools for model-based robotics: Comparison of Bullet, Havok, MuJoCo, ODE and PhysX”, IEEE International Conference on Robotics and Automation (ICRA), 4397-4404, 2015

[23] L. Pitonakova, M. Giuliani, A. Pipe and A. Winfield, "Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators", Annual Conference Towards Autonomous Robotic Systems, 2018, pp 357-368

[24] I. Bzhikhatlov and S. Perepelkina, "Research of robot model behaviour depending on model parameters using physic engines Bullet Physics and ODE", International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2017, pp. 1-4

[25] 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," 2013 16th International Conference on Advanced Robotics (ICAR), Montevideo, 2013, pp. 1-7

[26] Y. Yu, J. Yang, X. Zan, J. Huang and X. Zhang, “Research of simulation in character animation based on physics engine,” International Journal of Digital Multimedia Broadcasting, vol. 2017, Article ID 4815932, 7 pages, 2017

[27] E. Rohmer, S. P. N. Singh and M. Freese, "V-REP: A versatile and scalable robot simulation framework," 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, 2013, pp. 1321-1326

[28] CoppeliaSim, “Dynamics engines general properties”, 2020 [Online]. Available:http://www.coppeliarobotics.com/helpFiles/en/dynamicsEngineDialog.htm

[29] CoppeliaSim, “Simulation dialog”, 2020 [Online]. Available:http://www.coppeliarobotics.com/helpFiles/en/simulationPropertiesDialog.htm

[30] CoppeliaSim, “Video recorder”, 2020 [Online]. Available:http://www.coppeliarobotics.com/helpFiles/en/aviRecorder.htm