# A Roadmap for Simulation-Based Testing of Autonomous Cyber-Physical Systems: Challenges and Future Direction

Christian Birchler
christian.birchler@{zhaw,unibe}.ch
Zurich University of Applied Sciences
University of Bern
Switzerland

Sajad Khatiri
sajad.khatiri@zhaw.ch
Zurich University of Applied Sciences
Switzerland

Pooja Rani
rani@ifi.uzh.ch
University of Zurich
Switzerland

Timo Kehrer
timo.kehrer@unibe.ch
University of Bern
Switzerland

Sebastiano Panichella
sebastiano.panichella@zhaw.ch
Zurich University of Applied Sciences
Switzerland



## ABSTRACT

As the era of autonomous cyber-physical systems (ACPSs), such as unmanned aerial vehicles and self-driving cars, unfolds, the demand for robust testing methodologies is key to realizing the adoption of such systems in real-world scenarios. However, traditional software testing paradigms face unprecedented challenges in ensuring the safety and reliability of these systems. In response, this paper pioneers a strategic roadmap for simulation-based testing of ACPSs, specifically focusing on autonomous systems. Our paper discusses the relevant challenges and obstacles of ACPSs, focusing on test automation and quality assurance, hence advocating for tailored solutions to address the unique demands of autonomous systems. While providing concrete definitions of test cases within simulation environments, we also accentuate the need to create new benchmark assets and the development of automated tools tailored explicitly for autonomous systems in the software engineering community. This paper not only highlights the relevant, pressing issues the software engineering community should focus on (in terms of practices, expected automation, and paradigms), but it also outlines ways to tackle them. By outlining the various domains and challenges of simulation-based testing/development for ACPSs, we provide directions for future research efforts.

**ACM Reference Format:**
Christian Birchler, Sajad Khatiri, Pooja Rani, Timo Kehrer, and Sebastiano Panichella. 2024. A Roadmap for Simulation-Based Testing of Autonomous Cyber-Physical Systems: Challenges and Future Direction. In *Proceedings of International Workshop on Software Engineering in 2030 (SE 2030).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn


*SE 2030, November 2024, Puerto Galinàs (Brazil)*

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

In software engineering, testing stands as a cornerstone practice, essential for enhancing the reliability and robustness of software systems. Test automation techniques, including test generators, selection strategies, and prioritization methods, are pivotal in reducing the need for costly manual, error-prone testing procedures.

With the contemporary rise of *Autonomous Cyber-Physical Systems* (ACPSs), software engineers find themselves challenged by the need to evolve past/contemporary testing methodologies accordingly to such emerging systems requirements [16]. The inherent complexity of these systems is amplified by the challenge of conducting testing with appropriate input data and oracles (assertions), particularly when assessing system-level functionalities [6, 25].

A prevalent approach to testing ACPS, such as unmanned aerial vehicles (UAVs) and self-driving cars (SDCs), involves simulation environments, wherein the system under test operates within a simulated physical world [9, 27, 33]. Nevertheless, the applicability (or transferability) of traditional software testing techniques to such contexts remains unclear. Novel and open challenges arise when dealing with the simulation-based testing of ACPSs, including the level of realism of simulations, computational costs, the complexity of simulators, and the *Oracle Problem*.

Simulation-based testing research has garnered significant attention in recent years. Particularly in exploring the challenges of test generation for simulation-based tests in the domains of UAVs and SDCs [7, 17, 27, 36, 37]. This research represents a fundamental groundwork for understanding the challenges and needs inherent in simulation-based testing methodologies and testing practices for ACPSs. Search-based software testing techniques are a notable aspect of this research. These techniques have proven important results in simulation-based testing, facilitating advancements in test generation and improvement [15]. Furthermore, these search-based techniques find application beyond test generation, extending into regression testing tasks such as test minimization, selection, and prioritization [5, 9, 47].

Despite the progress made in simulation-based testing, several challenges persist. One challenge is the *Reality Gap* [1, 27, 35, 39], which refers to the disjunction between simulated environments

and real-world conditions. Additionally, issues like the *Oracle Problem* and the infinite input space for simulation-based test cases pose large obstacles to the efficacy of testing [6].

In light of these challenges, this paper proposes a roadmap for future research on simulation-based testing. Firstly, there is a pressing need to establish a clear definition and formulation of test cases tailored to simulation environments. By delineating the characteristics and requirements of simulation-based test cases, researchers can lay the groundwork for standardized testing methodologies. Formulating test cases for simulation environments necessitates careful consideration of various factors, including environmental dynamics, system behavior, and performance metrics.

Testing autonomous systems within simulation environments presents unique challenges. Addressing these challenges requires innovative approaches to test case design and evaluation methodologies. Furthermore, the paper advocates for creating and disseminating openly available benchmarks for simulation-based testing. Establishing standards can facilitate benchmarking and comparison of different testing methodologies, fostering collaboration and knowledge sharing within the research community. By making benchmarks openly available, researchers can streamline the testing process and promote transparency and reproducibility in experimental evaluations.

Overall, the field of simulation-based testing stands at a critical moment in time that needs further innovations and advancements. By addressing the challenges outlined in this paper and adhering to the proposed roadmap, researchers can contribute to this expected, relevant progress towards more robust and reliable testing methodologies for autonomous systems in simulation environments.

## 2 TEST CASE FORMULATION

In this section, we provide concise software testing definitions particularly tailored for simulation-based testing of ACPSs. Concretely, we define a test $\theta$ as a 4-tuple:

$$\theta = (S, E, T, O),$$

where $S$ reflects the test *subject* (i.e., the system under test), $E$ the subject's *environment*, $T$ the *task* for the subject, and $O$ the *oracle* that asserts the expected behavior of the subject. We describe each of these elements in detaul as follow.

**Subject.** First, let us define the set $U$ containing all elements of our universe:

$$U = \{x \mid \text{x is in the universe}\}$$

The system under test is the *Subject* of the test case. Formally, we define the subject as the following set:

$$S = \{x \in U \mid x \text{ is part of the system under test}\}$$

In traditional software testing, we have several levels of tests. For instance, *unit testing*, which tests a piece of code within a function, or *integration testing*, which tests the interaction between those units, but there are also component and system tests on higher abstraction levels. We have different test subjects on each of those levels, i.e., different code pieces.

In the case of autonomous systems, we can test not only code and its correct execution but also its AI models, the sensor interfaces, or the interaction between physical and software components. Furthermore, with different configuration options of those systems, we have various variants, which should be seen as different test subjects. It is clear that defining the test subject $S$ is more complex and not as intuitive as for traditional systems, since the test subject for general ACPSs does not merely consist on its source code.

**Environment.** Next to the test subject, we have an *environment* $E$ that embeds the test subject. We define the environment as follows:

$$E = \{x \mid x \in U \setminus S\}$$

The environment covers everything except the test subject $S$. In practice, the environment is often simplified, focusing solely on the operating system or hardware configurations. All other aspects of the universe $U$ are omitted as they are irrelevant for the test subject. However, in the case of simulation-based testing of ACPS, the environment usually has a higher cardinality (i.e., larger) as we model the physical world with simulators.

**Task.** Every test case has a *task* $T$ that the subject has to do. Formally, we define a task $T$ as a sequence of actions. Hence, we can write the following:

$$T = (a_0, a_1, a_2, ..., a_n),\ n \in \mathbb{N}_0$$

In traditional software testing, the test code (e.g., unit test) sets the environment and triggers the subject to perform an action $a$ (i.e., calling the function of interest). In simulation-based testing, defining the test and its tasks solely through code is rarely feasible. Simulators typically demand additional configuration files and scripts to interact with the simulation environment and describe tasks. Furthermore, tasks involve numerous smaller actions, such as setting waypoints to define an ACPS test track. So conceptually speaking, simulation-based tests have longer sequences defining the task than in traditional software testing, i.e., $n_{trad} < n_{sim}$.

**Oracle.** Software engineers have expectations of how the subject has to behave. We use the term *oracle* for those expectations of the subjects within a test. First, let us define $B$ as the set of all possible behaviors. Hence, the oracle $O$ is a map defined as follows:

$$O : B \times B \rightarrow \{0, 1\}$$

$$(b_{expected}, b_{actual}) \mapsto \begin{cases} 0 & \text{if } b_{expected} \neq b_{actual} \\ 1 & \text{if } b_{expected} = b_{actual} \end{cases}$$

In the case of a unit test of a function `sum` that accepts two arguments `a` and `b`, we expect to get the sum of these arguments. However, in simulation-based testing, the oracle does not check if a function returns the correct value but assesses how the subject behaves in the simulation environment. Hence, the concept of an oracle in simulation-based testing is more complex as we have to model behaviors with a sufficient abstraction to assert them with the actual observed ones [10].

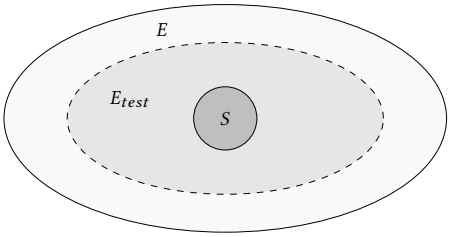


**Figure 1: The test subject $S$ is embedded in an environment $E$, from which we abstract many aspects away und only use a fraction as $E_{test}$.**

## 3 CHALLENGES

Applying concepts from Section 2 to ACPSs poses new automated testing challenges due to their complex, diverse simulation and real-world environments, as well as the test subject's complexity.

**Defining the testing task and the oracle.** In software testing, engineers execute task actions $T$ with specific argument values and observe the behavior $b_{actual}$. Well-defined metrics compare the actual behavior with the expected behavior $b_{expected}$. For instance, checking the outcome of a function on equality with the expected value. Verifying the software system behavior $b_{actual}$ against the correct behavior $b_{expected}$ is only partially automatable, i.e., automatically defining the map $O$. This challenge is known as the *Oracle Problem*. In ACPS simulation-based testing, defining task $T$ and asserting with oracle $O$ is challenging. Unlike traditional testing, simulation-based testing offers more freedom in defining $T$ and $O$. However, exploring the testing space of the physical world in simulation is computationally expensive and often not cost-effective. Moreover, simulations may lower the required computing power and the realism level of simulating the physical world. This represents a multifaced problem for simulation-based testing: On one hand, low simulation costs may lead to unrealistic actual behavior $b_{actual}$. On the other hand, expensive simulation may lead to more realistic behavior $b_{actual}$. Thus, addressing the *Oracle Problem* cost-effectively in simulation-based testing is complex and requires addressing totally new research challenges.

**Defining the environment.** In traditional software testing, engineers create test cases using the same programming language as the subject. Technologies such as *Docker* ensure a similar testing environment to production: Real-world factors can be typically ignored/overlooked in traditional testing, so engineers focus on a subset $E_{test}$, including *Docker*, OS, and hardware.

Simulation-based testing adds complexity by the need to model the entire physical world. When testing ACPS, software engineers must consider the physical world as the execution environment $E$ of an ACPS, which is the subject $S$. With this, two major challenges occur:

(1) What aspects of the environment $E$, i.e., the physical world, can be abstracted away so that we have the testing environment $E_{test}$?
(2) How do we simulate $E_{test}$ as realistic as possible to accurately verify the subject's behavior $b_{actual}$?

Software engineers and computer scientists work with abstractions to focus on specific aspects of their work. In software testing, they abstract away environmental complexity to validate system behavior reliably. Once the test environment $T_{test}$ is defined, a simulator must adequately replicate it. However, due to the nature of potentially inaccurate/simplified simulation environments, the behavior of the subject $S$ in simulation environments $E_{test}^{sim}$ may not always reflect the behavior in the real-world environment $E_{test}^{real}$; this leads to the *Reality Gap* problem.

**Reality gap.** In simulation-based testing, the *Reality Gap*[1, 27, 35] poses a critical concern. Simulated contexts often fail to faithfully mirror real-world situations due to simplifications necessary for computational feasibility. This trade-off between accuracy and computational time determines the extent to which simulations reflect reality[13]. Robotics simulations particularly struggle with accurately replicating phenomena such as actuators (e.g., torque characteristics, gear backlash), sensors (e.g., noise, latency), and rendered images (e.g., reflections, refraction, textures). Depending on the context, the reality gap can be quantified, e.g., measuring the difference a trajectory of a subject $S$ between a physical test environment $E_{test}^{real}$ and a simulated environment $E_{test}^{sim}$ This disparity between reality and simulation is known as the reality gap [13].

The reality gap has been an open research problem in robotics for years now. With the boost of *Evolutionary Robotics* and the application of reinforcement learning in designing robotic control systems in recent decades, practitioners rely more and more on simulations to evaluate their designs,i.e., test subjects [22, 40]. More specifically, a test subject's *fitness* (e.g., algorithm, trained model) is calculated based on its performance in simulation for reaching the robot goals. However, transferring robot skills acquired in a simulated environment to a physical setting, widely referred to as *Sim2Real transfer*, remains yet an open challenge [14].

**Lack of Benchmarks.** Because of the high complexity of autonomous systems and their testing infrastructure, benchmark artifacts (e.g., simulation logs, and implementations of test subjects) are rarely openly available for research. Furthermore, simulation-based testing is costly and hardly accessible to all researchers. Hence, those researchers rely on openly available datasets and benchmarks. A few recent examples in the domain of self-driving cars are SensoDat [11] and DeepScenario [32]; they provide datasets of driving scenarios and logged sensor data. However, the development and testing of ACPS rely on larger comprehensive datasets to train and evaluate the various AI technologies that are part of the ACPS.

**Need of cost-effective solutions.** As simulation-based testing for ACPS is inherently costly and non-sustainable, we require strategies to address this issue. Traditional software testing practices like agile culture, test-driven development (TDD), DevOps methodologies, and regression testing offer quick feedback loops for developers. Yet, adapting these techniques for simulation-based tests is uncertain. Another challenge is making ACPS development more agile by integrating fast feedback loops with system performance in a DevOps cycle. Undoubtedly, the research aims to make simulation-based testing more cost-effective.

## 4 AUTOMATION NEEDS & FUTURE WORK

This section discusses future research and automation needs for ACPSs, focusing on incorporating simulation-based testing into development processes and expanding the concept of test quality.

**Development and testing practices & paradigms.** Agile software development fosters iterative development and rapid feedback, aiding in adapting to new requirements. Test-Driven Development (TDD) ensures systematic testing of new requirements, where test cases precede feature implementation. Despite TDD's benefits, its applicability to simulation-based testing of ACPSs remains uncertain. Ideally, TDD bridges the reality gap, ensuring ACPS behavior aligns with requirements. While numerous test cases are generated via TDD, not all need execution for each system change, following the principle of *regression testing*. With regression testing, we run for a change to the system only relevant test cases that assess the new change's behavior and verify the existing functionality's correctness. To do so, techniques such as test minimization, selection, and prioritization are applied [43]. Applying regression testing techniques to simulation-based methods is challenging due to the requirement for computable metrics and features. Further research, as demonstrated by [8, 9], is needed to develop such metrics and features for simulations.

Effective test cases are essential for identifying bugs and ensuring the robustness and behavior of the system. In traditional software testing, *Mutation Testing* injects artificial bugs into the test subject $S$ to observe which test cases catch them [23]. However, applying mutation testing to ACPS in simulation is challenging due to the complexity of defining oracles in the simulated physical environment. Integrating these paradigms into ACPS development and testing can help in providing faster feedback loops for developers. Complementary, future work should address scalability and integration into DevOps steps to make these paradigms more practical, with specific attention on addressing bugs specific to ACPSs [41, 45].

**Representative oracle metrics.** Recent research has focused on generating or improving oracles [4, 24] for specific contexts, such self-driving cars, by simulating only the road shape. However, there is still no fully automated approach to mitigate the *Oracle Problem* for ACPS contexts. Human involvement is still necessary to evaluate safety and quality. For example, in simulation-based testing for ACPSs, metrics such those in [25, 28] are used, but recent research questions their static adoption [10]. Safety assessment via metrics may not always align with human perception due to the subjective nature of safety and its relation to realism and human experience [10]. Future research must consider what truly ensures ACPS safety and how it can be measured with quantifiable metrics.

We suggest *co-simulation* [19] to evaluate the subject's behavior, as $E_{test}$ is only approximated with simulations. Co-simulation involves multiple environments with varied physical behaviors, enhancing the robustness and determinism of oracles. Thus, test cases exhibit consistent behavior across multiple executions.

**Bridging the Reality Gap.** *Domain Randomization* techniques address the reality gap by exposing algorithms to diverse random simulation environments, assuming real-world variability. This approach aims for robustness across environments and easy transferability [26, 31]. Others advocate combining simulated evaluation with a small amount of real-world data [12, 46], typically by recalculating fitness for selected solutions in the real world and integrating their deviation into optimization processes. Some methods optimize general physics simulators based on real-world data, updating default settings using optimal values from real-world measurements [13]. Future directions may involve exploring hybrid approaches that seamlessly blend simulated and real-world data, leveraging advancements in reinforcement learning and transfer learning techniques. Advancements in hardware capabilities could enable more sophisticated simulation environments, enhancing variability and fidelity in training scenarios. Incorporating domain adaptation methods from machine learning could effectively bridge the gap between simulated and real-world environments.

Previous research has highlighted challenges with simulators for testing CPS, including the reality gap, engineering complexity of realistic environments, and replicating real-world bugs [1, 2, 42]. While solutions exist for the development phase, few address testing: *How can simulators be better utilized for CPS testing, given the reality gap?* Hildebrandt et al. [21] propose a mixed-reality method called *world-in-the-loop* simulation for UAV testing, integrating sensor data from simulated and real environments to enhance simulation realism and diversify real-world testing. Khatiri et al.'s *Surrealist* approach [27] enables realistic simulation-based UAV testing by replicating real flights and accurately reconstructing surroundings, facilitating the identification of challenging test cases resembling real-world scenarios. They also developed Aerialist [29], the first UAV test bench and test generation platform [30], offering new research opportunities in the field.

**Sustainable testing practices for ACPS.** We recognize the importance of dependable ACPSs, but must also consider the influence of quality assurance tools and technologies on climate. The ICT sector is projected to produce 5.5% of global carbon emissions and consume 20% of all electricity by 2025 [3]. With the growing adoption of data-intensive technologies and software in daily life, software energy consumption is expected to rise. Software testing consumes many resources regarding infrastructure and tools. For instance, tools and techniques for creating, prioritizing, or running tests in continuous integration and deployment pipelines [8, 9, 20]. Zaidman et al. [44] estimated the energy impact of various software projects and found that the *Elasticsearch* project was built 5025 times in 2022. Hence, it consumed 161.5 kWh of electricity for building the project, which corresponds to 9.7% of the yearly average household energy consumption of a European Union citizen [44].

Recent work in software engineering has highlighted *green coding* practices and energy patterns for source code [18, 34, 38]. However, energy patterns for software testing remain unexplored, along with the energy impact of testing practices. Questions arise: How frequently should source code be built? How can regression test suites be built energy-efficiently [38]? Furthermore, what is the awareness among ACPS developers regarding the energy consumption of their software, and what strategies can reduce energy consumption? Future research should explore questions to aid developers in reducing the energy footprint of software and hardware components in such systems.

## 5 CONCLUSION

Simulation-based testing is a standard method for evaluating the safety and quality of ACPS. While it shares similarities with traditional software testing, it's inherently more complex and can exacerbate existing issues or introduce new ones, such as the *Reality Gap* (Section 3). This paper aims to establish a unified definition and framework for testing, discussing both traditional and simulation-based approaches. It also identifies areas for future research, highlighting the challenges and automation requirements to ensure the development of safer and more sustainable/reliable ACPS for our society.

## ACKNOWLEDGMENTS

We thank the Horizon 2020 (EU Commission) support for the COSMOS project, Project No. 957254-COSMOS.

## REFERENCES

[1] Afsoon Afzal, Deborah S Katz, Claire Le Goues, and Christopher S Timperley. 2021. Simulation for robotics test automation: Developer perspectives. In *Conference on Software Testing, Verification and Validation*. IEEE, 263–274.

[2] Afsoon Afzal, Claire Le Goues, Michael Hilton, and Christopher Steven Timperley. 2020. A study on challenges of testing robotic systems. In *Intl. Conference on Software Testing, Validation and Verification*. IEEE, 96–107.

[3] Anders SG Andrae and Tomas Edler. 2015. On global electricity usage of communication technology: trends to 2030. *Challenges* 6, 1 (2015), 117–157.

[4] Aitor Arrieta, Maialen Otaegi, Liping Han, Goiuria Sagardui, Shaukat Ali, and Maite Arratibel. 2022. Automating Test Oracle Generation in DevOps for Industrial Elevators. In *Intl. Conference on Software Analysis, Evolution and Reengineering*. IEEE, 284–288.

[5] Aitor Arrieta, Shuai Wang, Goiuria Sagardui, and Leire Etxeberria. 2016. Search-based test case selection of cyber-physical system product lines for simulation-based validation. In *Intl. Systems and Software Product Line Conference*, Hong Mei (Ed.). ACM, 297–306.

[6] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *IEEE Trans. Software Eng.* 41, 5 (2015), 507–525.

[7] Matteo Biagiola, Stefan Klikovits, Jarkko Peltomäki, and Vincenzo Riccio. 2023. SBFT Tool Competition 2023 - Cyber-Physical Systems Track. In *Intl. Workshop on Search-Based and Fuzz Testing*. IEEE, 45–48.

[8] Christian Birchler, Sajad Khatiri, Bill Bosshard, Alessio Gambi, and Sebastiano Panichella. 2023. Machine learning-based test selection for simulation-based testing of self-driving cars software. *Empir. Softw. Eng.* 28, 3 (2023), 71.

[9] Christian Birchler, Sajad Khatiri, Pouria Derakhshanfar, Sebastiano Panichella, and Annibale Panichella. 2023. Single and Multi-objective Test Cases Prioritization for Self-driving Cars in Virtual Environments. *ACM Trans. Softw. Eng. Methodol.* 32, 2 (2023), 28:1–28:30.

[10] Christian Birchler, Tanzil Kombarabettu Mohammed, Pooja Rani, Teodora Nechita, Timo Kehrer, and Sebastiano Panichella. 2024. How does Simulation-based Testing for Self-driving Cars match Human Perception?. In *Intl. Conference on the Foundations of Software Engineering*.

[11] Christian Birchler, Cyrill Rohrbach, Timo Kehrer, and Sebastiano Panichella. 2024. SensoDat: Simulation-based Sensor Dataset of Self-driving Cars. In *Intl. Conference on Mining Software Repositories*.

[12] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. 2018. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *Intl. Conference on Robotics and Automation*. IEEE, 4243–4250.

[13] Jack Collins, Ross Brown, Jurgen Leitner, and David Howard. 2020. Traversing the reality gap via simulator tuning. *arXiv preprint arXiv:2003.01369* (2020).

[14] Konstantinos Dimitropoulos, Ioannis Hatzilygeroudis, and Konstantinos Chatzilygeroudis. 2022. A Brief Survey of Sim2Real Methods for Robot Learning. In *Intl. Conference on Robotics in Alpe-Adria Danube Region*. Springer, 133–140.

[15] Federico Formica, Tony Fan, Akshay Rajhans, Vera Pantelic, Mark Lawford, and Claudio Menghi. 2024. Simulation-Based Testing of Simulink Models With Test Sequence and Test Assessment Blocks. *IEEE Trans. Software Eng.* 50, 2 (2024), 239–257.

[16] Sylvain Frey, Awais Rashid, Pauline Anthonysamy, Maria Pinto-Albuquerque, and Syed Asad Naqvi. 2019. The Good, the Bad and the Ugly: A Study of Security Decisions in a Cyber-Physical Systems Game. *IEEE Trans. Software Eng.* 45, 5 (2019), 521–536.

[17] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. 2022. SBST Tool Competition 2022. In *Intl. Workshop on Search-Based Software Testing*. IEEE, 25–32.

[18] Stefanos Georgiou, Stamatia Rizou, and Diomidis Spinellis. 2019. Software development lifecycle for energy efficiency: techniques and tools. *ACM Comput. Surv.* 52, 4 (2019), 1–33.

[19] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. 2018. Co-Simulation: A Survey. *ACM Comput. Surv.* 51, 3, Article 49 (may 2018), 33 pages.

[20] Giovanni Grano, Christoph Laaber, Annibale Panichella, and Sebastiano Panichella. 2021. Testing with Fewer Resources: An Adaptive Approach to Performance-Aware Test Case Generation. *IEEE Trans. Software Eng.* 47, 11 (2021), 2332–2347.

[21] Carl Hildebrandt and Sebastian Elbaum. 2021. World-in-the-loop simulation for autonomous systems validation. In *Intl. Conference on Robotics and Automation*. IEEE, 10912–10919.

[22] Sebastian Höfer, Kostas Bekris, Ankur Handa, Juan Camilo Gamboa, Melissa Mozifian, Florian Golemo, Chris Atkeson, Dieter Fox, Ken Goldberg, John Leonard, et al. 2021. Sim2Real in robotics and automation: Applications and challenges. *IEEE Transactions on Automation Science and Engineering* 18, 2 (2021), 398–400.

[23] William E. Howden. 1982. Weak Mutation Testing and Completeness of Test Sets. *IEEE Trans. Software Eng.* 8, 4 (1982), 371–379.

[24] Gunel Jahangirova, David Clark, Mark Harman, and Paolo Tonella. 2016. Test oracle assessment and improvement. In *Intl. Symposium on Software Testing and Analysis*. ACM, 247–258.

[25] Gunel Jahangirova, Andrea Stocco, and Paolo Tonella. 2021. Quality Metrics and Oracles for Autonomous Vehicles Testing. In *Conference on Software Testing, Verification and Validation*. IEEE, 194–204.

[26] Stephen James, Andrew J Davison, and Edward Johns. 2017. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *Conference on Robot Learning*. PMLR, 334–343.

[27] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based Test Case Generation for Unmanned Aerial Vehicles in the Neighborhood of Real Flights. In *IEEE Conference on Software Testing, Verification and Validation*. IEEE, 281–292.

[28] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based Test Case Generation for Unmanned Aerial Vehicles in the Neighborhood of Real Flights. In *Intl. Conference on Software Testing, Verification and Validation*. IEEE, 281–292.

[29] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2024. Simulation-based Testing of Unmanned Aerial Vehicles with Aerialist. In *Intl. Conference on Software Engineering*.

[30] Sajad Khatiri, Prasun Saurabh, Timothy Zimmermann, Charith Munasinghe, Christian Birchler, and Sebastiano Panichella. 2024. SBFT Tool Competition 2024: CPS-UAV Test Case Generation Track. In *Intl. Workshop on Search-Based and Fuzz Testing*.

[31] Timothy E Lee, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Oliver Kroemer, Dieter Fox, and Stan Birchfield. 2020. Camera-to-robot pose estimation from a single image. In *Intl. Conference on Robotics and Automation*. IEEE, 9426–9432.

[32] Chengjie Lu, Tao Yue, and Shaukat Ali. 2023. DeepScenario: An Open Driving Scenario Dataset for Autonomous Driving System Testing. In *Intl. Conference on Mining Software Repositories*. IEEE, 52–56.

[33] Toni Mancini, Igor Melatti, and Enrico Tronci. 2023. Optimizing Highly-Parallel Simulation-Based Verification of Cyber-Physical Systems. *IEEE Trans. Software Eng.* 49, 9 (2023), 4443–4455.

[34] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. 2016. An empirical study of practitioners' perspectives on green software engineering. In *Intl. Conference on Software Engineering*. 237–248.

[35] Anthony Ngo, Max Paul Bauer, and Michael Resch. 2021. A Multi-Layered Approach for Measuring the Simulation-to-Reality Gap of Radar Perception for Autonomous Driving. In *Intl. Intelligent Transportation Systems Conference*. IEEE, 4008–4014.

[36] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. 2021. SBST Tool Competition 2021. In *Intl. Workshop on Search-Based Software Testing*. IEEE, 20–27.

[37] Samuel Parra, Argentina Ortega, Sven Schneider, and Nico Hochgeschwender. 2023. A Thousand Worlds: Scenery Specification and Generation for Simulation-Based Testing of Mobile Robot Navigation Stacks. In *IROS*. 5537–5544.

[38] Pooja Rani, Jonas Zellweger, Veronika Kousadianos, Luis Cruz, Timo Kehrer, and Alberto Bacchelli. 2024. Energy Patterns for Web: An Exploratory Study. *arXiv preprint arXiv:2401.06482* (2024).

[39] Fabio Reway, Abdul Hoffmann, Diogo Wachtel, Werner Huber, Alois C. Knoll, and Eduardo Parente Ribeiro. 2020. Test Method for Measuring the Simulation-to-Reality Gap of Camera-based Object Detection Algorithms for Autonomous Driving. In *IEEE Intelligent Vehicles Symposium*. IEEE, 1249–1256.

[40] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. 2021. Crossing the Reality Gap: a Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning. *IEEE Access* (2021).

[41] Andrea Di Sorbo, Fiorella Zampetti, Aaron Visaggio, Massimiliano Di Penta, and Sebastiano Panichella. 2023. Automated Identification and Qualitative Characterization of Safety Concerns Reported in UAV Software Platforms. *ACM Trans. Softw. Eng. Methodol.* 32, 3 (2023), 67:1–67:37.

[42] Dinghua Wang, Shuqing Li, Guanping Xiao, Yepang Liu, and Yulei Sui. 2021. An exploratory study of autopilot software bugs in unmanned aerial vehicles. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 20–31.

[43] Shin Yoo and Mark Harman. 2012. Regression testing minimization, selection and prioritization: a survey. *Softw. Test. Verification Reliab.* 22, 2 (2012), 67–120.

[44] Andy Zaidman. 2024. An Inconvenient Truth in Software Engineering? The Environmental Impact of Testing Open Source Java Projects. (2024).

[45] Fiorella Zampetti, Ritu Kapur, Massimiliano Di Penta, and Sebastiano Panichella. 2022. An empirical characterization of software bugs in open-source Cyber–Physical Systems. *J. Syst. Softw.* 192 (2022), 111425.

[46] Fangyi Zhang, Jürgen Leitner, Zongyuan Ge, Michael Milford, and Peter Corke. 2019. Adversarial discriminative sim-to-real transfer of visuo-motor policies. *The International Journal of Robotics Research* 38, 10-11 (2019), 1229–1245.

[47] Man Zhang, Shaukat Ali, and Tao Yue. 2019. Uncertainty-wise test case generation and minimization for Cyber-Physical Systems. *J. Syst. Softw.* 153 (2019), 1–21.