An Analysis of the Difficulties of Testing Robot Systems for Real World Use

Parker Hutchinson (pch6am) and Sidhardh Burre (ssb3vk)

***Abstract*—The simulation-reality gap poses a daunting challenge for robotics engineers. While simulations are relatively cheap, safe, and easy to generate, they lack the robustness and semantic meaning that real-world training offers. Conversely, real-world testing is expensive and difficult to control leading to high-costs, long timelines, and potential lives lost. While it is possible to extract data from the real-world and present in simulation via a record-and-replay system, the same problems that plague real world testing persist; expensiveness in data collection, small sample-size (which can lead to overfitting), and difficulty in generalized data. In this paper, we explore the nuances of both simulation and real-world testing examining industry best practices and challenges in the area of robotics development and testing, taking samples from papers that represent the bleeding edge of testing research. We find that methods that use a blend of both real-world and simulation testing are most effective and can lead to a robotic future.**

I. Introduction

Robot systems are becoming more and more prevalent in our daily lives. From autonomous vehicles to rudimentary packaging systems, robots are developed and deployed to the real world, and are poised to only become a larger part of our future [1]-[3]. Although the systems in our employ are primarily used only to assist or perform a variety of difficult or dangerous jobs they represent a new avenue for failure [4]-[7].

Therefore, it is essential that in the development of these robots, practical and effective quality assurance techniques are applied. Currently, modes of testing robots are limited to field testing and simulation based testing. Both of which have their own unique advantages and disadvantages.

Robot operating environments are highly variable. Robots mark the intersection between software, where we have a finite state space; and reality, where we have an unpredictable and boundless state space. Further, this state space can have extreme edge cases such as extreme temperatures or even sensor failures. Even when robots are tested in the real world, it is difficult, if not impossible, to mimic some conditions. So testing the robot extensively via real-world testing is largely infeasible. A potential solution is the use of modeling reality in a simulator but "no simulators currently exist where the information is even close to the reality, they are nowhere close to the noise and variability of real-world data" [8]. This is only furthered by the fact that bugs found in the real world are difficult if not impossible to replicate in simulation.

A potential solution to the aforementioned quandary is the use of record and replay when testing systems in complex environments. But even through the ease of collecting real-world data, issues still exist with dynamic data, where there is a feedback loop between the robot and the environment, a fact that is only worsened in the case that robot models can be overfit to representative samples of real-world data.

Therefore, there lies a clear problem in the space of training and testing robots. What is the ideal method to safely and reliably test robots prior to deployment in a resource-effective way if neither simulations nor real-world data can be used? Survey of the literature

II. Survey of Literature

In the following paper "Can robot navigation bugs be found in simulation? An exploratory study", researchers study the complex problem of navigation in a simulated environment. Specifically, they tested the navigation stack in an artificial environment to target the following questions:

Can the triggers and effects of robot navigation bugs be reproduced in low-fidelity simulation?

From the analysis of triggers, which elements are to be considered in the input model of worlds and missions used for testing?

From the analysis of triggers, which elements are to be considered in the input data configuration?

From the analysis of effects, which observation data and oracle procedures should be considered?

An environment model was set via a picture of the robot's planned environment with injected start/end points. Care was taken to control both the granularity and amplitude of irregularities within the simulated environment.

Throughout the study, it was observed that a surprising majority of the bugs (⅓) were caused by issues with configuration files and non-domain-specific bugs. This includes errors with configuring sensors and proper coding patterns (avoiding memory leaks and out-of-range array indexing). Of the domain-specific bugs that were assessed, it was found that highly-complex simulation environments were

not necessary to replicate them. An example bug was related to spot turn and map-building functions which could be fixed by taking into account inertia and braking rate. More interestingly, dangerous bugs were less prevalent. This includes robot byzantine failure bugs such as the robot allowing itself to fall into a hole/ditch. But still other errors were not considered including error triggers that combined several inputs such as mission replacement, while the robot is turning on spot, or if a narrow dead-end blocked the robot's path. Therefore, the authors recommended situation-based testing that observed the robot in a combination of world/mission elements. This suggests that although low-granularity model-based testing is sufficient for identifying primitive errors, it is not sufficient to identify situation-based errors. Which poses a significant problem in simulation testing; how can we define the upper limit on simulation specificity. Or to put it another way, how similar does simulation have to be to reality before it can catch all errors. Which leads to the following question: if simulations have to be developed that are outright mirrors of reality, isn't it more feasible to test in reality itself?

In "Real-World Testing of a Multi-Robot Team", a team of researchers from Carnegie Mellon's Robotics Institute describe how they were able to effectively train their Cooperative Robotic Watercraft (CRW) system through extensive field testing and data collection in the real world. The small CRW airboats were deployed in various environments and measured variables like temperature and dissolved oxygen in the water. To qualify, they note that many of the commonly critical robot problems are minimized on the water, because "movement is relatively simple and dangers are relatively low". The team tested the CRW boats at three separate sites, and noted that the first site alerted them to problems that were addressed prior to future deployments. Overall, the boats were tested for over 100 hours in the water, creating over one hundred thousand data points. In this specific instance, the real-world testing also showed the team that locals could operate their boats sufficiently. Real-world problems with "wireless technologies, particularly 3G" grew prominent as testing went on [11].

Clearly, real-world testing was important for this team's success. Without it, they might not have identified the necessary fix made between their first and second deployments. They also may not have predicted or been able to accurately simulate the difficulties encountered with wireless communications. But while real-world simulation was invaluable for the CRW, it was still insufficient to guarantee robot performance in the real world, a necessity for life-critical robotics.

In "Closing the Simulation-to-Reality Gap using Generative Neural Networks: Training Object Detectors for Soccer Robotics in Simulation as a Case Study", a novel paper that proposes using artificial intelligence to close the simulation/reality gap, a team of researchers utilizes Generational Neural Networks to generate test examples from existing training data[13]. Although the proposed method is traditionally employed to generate training examples for machine learning algorithms, the methodology was used in the paper to train robots to play soccer. Traditional robot training methods use simulators to train the robots for certain tasks and then attempt to apply that robots learning to the real world, a form of transfer learning. However this transfer is not always feasible due to the mismatch between simulated and real environments. A common technique pre-processes simulated/real data to reduce the mismatch between the two and then applies training to only the transformed data. So this would be all simulated data augmented with transformed real data or vice-versa. An alternate approach is to generate adversarial examples that specifically target the simulation-reality gap. However generating these adversarial examples is extremely difficult due to the need to have high domain-based knowledge of both the simulator and reality. Something that falls apart in complex systems such as self-driving vehicles. The paper employs Generational Neural Networks to transform simulated images to realistic ones during simulation and vice-versa, to create a unified domain of simulation from all testing environments. Through a series of convolutional neural networks and a novel technique, Bilinear Sampling, the paper suggests that the generative model is able to generate novel training examples solely from observed input data. The robot, after being trained on the combined data-set, was able to nearly match the performance of robots that were trained solely on real-world images within a small margin of error (2%) but with the same number of initial samples. This indicates that the Generational Neural Network represents a real avenue for using machine learning to bridge the simulation-reality gap.

Some developers may look to more extensive testing of their own codebase as a way to ensure the efficacy of their systems. Numerous tools exist to aid in this pursuit, with Integrated Development Environments (IDEs) allowing programmers to add automatically generated test cases for functions in a variety of circumstances, cutting out the need for an extensive amount of time to be spent on writing unit tests. In "Do Automatically Generated Unit Tests Find Real Faults? An Empirical Study of Effectiveness and Challenges", Shamshiri and her peers analyze how well such "state-of-the-art" tools for unit testing, including "Randoop, EvoSuite, and Agitar" perform in the broader scope of finding faults in the overall system they function in, the "Defects4J dataset". The authors noted that automatically-generated test suites had an abysmal success rate in their trials: only 19.9% of such tests managed to find real faults in the system. The team said that these tools created tests that were lacking in "code coverage", with 16.72% of the given faults never being executed by the automatically created tests. However, 63.3% of the faults were "covered by automatically generated tests at least once", but remained undetected overall. Problems were also noted with some of the generated tests themselves; 15.2% of the tests were "flaky", or unstable. Such tests may depend on external variables, and thus could pass at one time but fail at another, with nothing in the actual code changing.

1. Clearly, there is an issue with relying on the automatic generation of unit tests to ensure the competency of a codebase; developers must take a more thorough, holistic look at their codebase, using metrics including code coverage and others to guide their testing philosophy. As discussed previously, empirical testing using real world data is likely another great source to draw from for developers who hope to build robust computer systems that retain their functionality from simulated environments in reality. Developers should use automatically generated unit tests to supplement their testing, but must keep in mind that faults are often undetected by these systems and thus their performance in the real world may have unexpected errors not detected during the testing stage.

III. Analysis and Discussion

In "Can robot navigation bugs be found in simulation? An exploratory study" researchers found that simulation was in-fact useful for avoiding significant bugs. Bugs that would represent difficulty to test and find in the field. Particularly, the usefulness of simulation was marked when it came to identifying trivial bugs that don't come from robot behavior but instead from robot configuration settings. However, in "Real-World Testing of a Multi-Robot Team" it was found that real-world testing was invaluable to testing the hardware of the robot, the key part of a Cyber-physical system (CPS) placing researchers at a crossroads. While both simulation and reality-based testing are valuable, how can a researcher consolidate the two such that the validity of neither is lost.

Reality often disappoints, but it might be the only way to accurately test robot systems. CPS can be efficiently trained and tested in simulated environments, where the setup and teardown costs are minimal and trials can be repeated thousands of times with little effort. Unfortunately, building a simulation system that parallels reality sufficiently is a daunting task, however, and failure to bridge the simulation-reality gap appears to result in a plethora of verification issues for the CPSs that use the system to train. Real-world data is extremely noisy and variable, in a way that current simulators cannot replicate [1]. Accounting for human mental and emotional state is another challenge for simulators seeking to clone the real world, although they may be approximated in some scenarios, as in the pilot behavior example from Human-in-the-Loop Issue in Context of the Cyber-Physical Systems. For these reasons, and others, it's easy to understand why the majority of CBPS developers believe that simulation alone is insufficient for supporting the verification and validation of their machines. These developers specifically noted a lack of a formal connection to physics models as a key issue for simulation veracity [21].

Challenges occur even when testing robots in the real world. Bugs that occur when robots are tested in physical environments can be hard to reproduce, because of the variety of variables in reality. These fleeting bugs are called Heisenbugs. Record and replay, the process of recording sensor data for future use in simulation trials, helps to address this problem, but developers must be sure to use a wide variety of data when employing it lest their robots become overfitted to a small subset of recorded test data. Notably, record and replay can't be used if the robot requires a feedback loop with its environment, since the 'record' aspect of the process only attains values for a specific set of actions taken by the robot.

One important example of real world data that CBPS designers must address is the failure of sensor data during robot performance. To illustrate this importance, one might note that the failure of a single sensor in the Boeing 737 Max caused catastrophic plane crashes on two separate occasions [12]. Although they may not be able to be completely simulated, teasing out dependencies on sensors using the record and replay method could significantly aid in the verification process, allowing for these problems to be detected and then tested more thoroughly in the real world.

But if neither reality-based testing nor simulation-based testing can fulfill the stringent requirements that must be met to deploy robots into the real world, then there must be an alternative solution that can be employed. The paper "Closing the Simulation-to-Reality Gap using Generative Neural Networks: Training Object Detectors for Soccer Robotics in Simulation as a Case Study"[13] suggests that we use AI to generate simulations from real-world scenarios which presents a path to bridge the gap between simulation and reality but currently Generative Neural Networks are in their infancy and lack the power to generate highly complex simulations that preserver the semantic meaning of real world examples.

IV. Conclusions

Aligning computer models with the real world can be a daunting task. Try as engineers may, difficulties abound in attempting to build the 'perfect robot', and thus they must settle for doing the best they can. After analyzing the available literature, it appears that the best strategy for developers trying to simulate the noise of the real world for their systems is to emphasize the importance of retrieving actual data from physical data, and using strategies like 'record and replay' to extend the usefulness of some data. Relying on simulated data alone for testing robot systems appears to be insufficient for creating resilient, realistic solutions.

Although robotics currently represent a large portion of manual work such as assembly lines, surveillance, and shipping, they can continue to expand into more domains. The research presented here has high relevance to many major emerging technologies. This includes automated driving, surgery, surveillance, manual work, and exploration. By solving the problems outlined in this paper, it becomes possible to cheaply and efficiently step into a future where humanity can employ robotics and leaps and bounds.

References

[1] R. Cellan-Jones, "Robots 'to replace up to 20 million factory jobs' by 2030," BBC News. [Online]. Available: https://www.bbc.com/news/ business-48760799

[2] R. Heilweil, "Networks of self-driving trucks are becoming a reality in the US," Vox. [Online]. Available: https://www.vox.com/recode/2020/7/ 1/21308539/self-driving-autonomous-trucks-ups-freight-network

[3] Hyperdrive, "The state of the self-driving car race 2020," Bloomberg. [Online]. Available: https://www.bloomberg.com/features/ 2020-self-driving-car-race

[4] S. O'Kane, "Boeing finds another software problem on the 737 Max," The Verge. [Online]. Available: https://www.theverge.com/2020/ 2/6/21126364/boeing-737-max-software-glitch-flaw-problem

[5] M. Wall, "European Mars lander crashed due to data glitch, ESA concludes," Space. [Online]. Available: https://www.space.com/ 37015-schiaparelli-mars-lander-crash-investigation-complete.html

[6] R. N. Charette, "Nissan recalls nearly 1 million cars for air bag software fix," IEEE Spectrum. [Online]. Available: https://spectrum.ieee.org/riskfactor/transportation/ safety/nissan-recalls-nearly-1-million-cars-for-airbag-software-fix

[7] P. McCausland, "Self-driving Uber car that hit and killed woman did not recognize that pedestrians jaywalk," NBC News. [Online]. Available: https://www.nbcnews.com/tech/tech-news/ self-driving-uber-car-hit-killed-woman-did-not-recognize-n1079281

[8] A. Afzal, C. L. Goues, M. Hilton, and C. S. Timperley, "A Study on Challenges of Testing Robotic Systems," in 2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST), Oct. 2020, pp. 96–107. doi: 10.1109/ICST46399.2020.00020.

[9] A. Afzal, D. S. Katz, C. Le Goues, and C. S. Timperley, "Simulation for Robotics Test Automation: Developer Perspectives," in 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST), Apr. 2021, pp. 263–274. doi: 10.1109/ICST49551.2021.00036.

[10] T. Sotiropoulos, H. Waeselynck, J. Guiochet, and F. Ingrand, "Can robot navigation bugs be found in simulation? An exploratory study," in 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS2017), Prague, Czech Republic, Jul. 2017, p. 10p. Accessed: Oct. 31, 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01534235

[11] S. Paul, V. Prasanna, K. Balajee, V. Abhinav, T. Christopher, D. John, S. Adrian, S. Kumar, B. Luis, and K. George (2012, June). ReaReal-World Testing of a Multi-Robot Team. The Robotics Institute. Retrieved October 31, 2022, from https://www.ri.cmu.edu/pub_files/2012/6/main.pdf

[12] Gates, D. (2020, November 22). Q&A: What led to Boeing's 737 max crisis. The Seattle Times. Retrieved October 31, 2022, from https://www.seattletimes.com/business/boeing-aerospace/what-led-to-boeings-737-max-crisis-a-qa/#:~:text=Two%20crashes%20of%20virtually%20new,jet%20into%20a%20nose%20dive.

[13] 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), Jul. 2020, pp. 1–8. doi: 10.1109/IJCNN48605.2020.9207173.

[14] S. Shamshiri, R. Just, J. Rojas, G. Fraser, P. McMinn, and A. Arcuri (2015). "Do Automatically Generated Unit Tests Find Real Faults? An Empirical Study of Effectiveness and Challenges" in 2015 30th IEEE/ACM International Conference on Automated Software Engineering.