Åbo Akademi

OPEN ACCESS

This is an electronic reprint of the original article. This reprint may differ from the original in pagination and typographic detail.

Adaptive test generation for unmanned aerial vehicles using WOGAN-UAV

Winsten, Jesper; Soloviev, Valentin; Peltomäki, Jarkko; Porres, Ivan

Published in:
Proceedings - 2024 IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2024

DOI:
10.1145/3643659.3648603

Published: 14/04/2024

Document Version
Final published version

Document License
CC BY

Link to publication

Please cite the original version:
Winsten, J., Soloviev, V., Peltomäki, J., & Porres, I. (2024). Adaptive test generation for unmanned aerial vehicles using WOGAN-UAV. In *Proceedings - 2024 IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2024* (pp. 43-44). (Proceedings - 2024 IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT 2024). ACM. https://doi.org/10.1145/3643659.3648603

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

This document is downloaded from the Research Information Portal of ÅAU: 15. Mar. 2025

Adaptive test generation for unmanned aerial vehicles using WOGAN-UAV

Jesper Winsten, Valentin Soloviev, Jarkko Peltomäki, Ivan Porres
Information Technology, Faculty of Science and Technology
Åbo Akademi University, Turku, Finland
name.surname@abo.fi

ABSTRACT

WOGAN-UAV is a test generation tool based on Wasserstein generative adversarial networks. In this paper, we present how WOGAN-UAV works and how it can be applied in the Unmanned Aerial Vehicle Testing Competition held as a part of SBFT 2024 workshop.

CCS CONCEPTS

• **Software and its engineering** → **Search-based software engineering**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

ACM Reference Format:
Jesper Winsten, Valentin Soloviev, Jarkko Peltomäki, Ivan Porres. 2024. Adaptive test generation for unmanned aerial vehicles using WOGAN-UAV. In *2024 ACM/IEEE International Workshop on Search-Based and Fuzz Testing (SBFT '24), April 14, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3643659.3648603

1 INTRODUCTION

Cyber-physical systems (CPSs) such as self-driving cars, autonomous robots, and unmanned aerial vehicles (UAVs) combine physical components with embedded computing elements enabling advanced perception, planning, and control functions. The complexity and safety-critical nature of these systems make rigorous testing and validation a necessity before they are used in the real world.

This article describes the WOGAN-UAV tool, an entry in the Unmanned Aerial Vehicle Testing Competition held as a part of the Search-Based and Fuzz Testing (SBFT) 2024 workshop [4]. In the competition, the open source tool Aerialist [5, 6] is used to simulate UAVs. We consider it as our system under test (SUT).

The goal of the competition is to develop a tool that places obstacles on the flight path of an UAV in such a way that the UAV flies too close to an obstacle or even crashes into one. The safe distance is set to 1.5 m. An example test scenario and the UAV flight path are depicted in Figure 1.

This research work has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007350 AIDOaRT.

This work licensed under Creative Commons Attribution International 4.0 License.

SBFT '24, April 14, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0562-5/24/04.
https://doi.org/10.1145/3643659.3648603

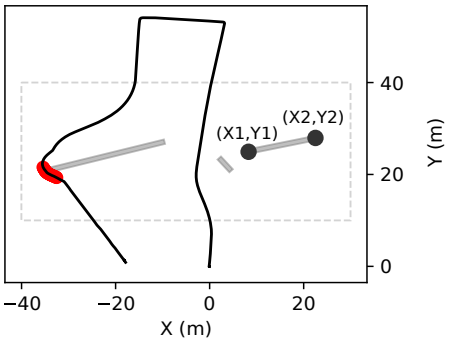

Figure 1: The obstacles (gray boxes) and the flight path of the UAV (black line). The dashed line indicates the area where obstacles can be placed and red indicates UAV flying too close to an obstacle.

2 WOGAN-UAV TEST GENERATION TOOL

The WOGAN-UAV tool uses the WOGAN algorithm [7] for the UAV problem. This algorithm is implemented in the STGEM tool[1] and has also been used to develop other testing tools [8, 9] that participated in previous editions of the SBFT CPS competition [2, 3]. Compared to WOGAN-UAV, the previous tools generate test suites to validate the lane-keep assist system of a self-driving car. The WOGAN hyperparameter setup used for the UAV problem is identical to the setup of [9].

2.1 The WOGAN Algorithm

The WOGAN algorithm uses Wasserstein generative adversarial networks (WGANs) [1] to automate test generation for CPSs. WOGAN is applicable to any black-box SUT that is reasonably deterministic and accepts vector or signal inputs and outputs. To use WOGAN, an objective function for minimization is required. In the context of the UAV problem, the input encodes the obstacles as a vector (see Subsection 2.2), and the output is a signal of distances to each obstacle during the simulation. The objective function returns the minimum distance between the UAV and all obstacles.

The aim of WOGAN is to train a generator $\mathcal{G}$ that can synthesize low-objective inputs from noise. The generator is trained as a WGAN using training data created online during the algorithm execution. The result of the algorithm is a test suite of inputs executed during the training and the trained generator $\mathcal{G}$. The test suite already contains many low-objective inputs, and $\mathcal{G}$ can be further sampled for inputs by computing $\mathcal{G}(x)$ for noise points x.

[1] https://gitlab.abo.fi/stc/stgem

In Section 3, we analyze the test suite, not the inputs obtained from $\mathcal{G}$ after the training has finished.

The training data for WOGAN is obtained as follows. First, an initial random search is performed to be able to train $\mathcal{G}$. After this, $\mathcal{G}$ itself is used for augmenting the training data. The idea is that $\mathcal{G}$, even if partially trained, can generate novel inputs that have lower objective than inputs obtained via random search. These novel inputs are executed on the actual SUT, and the results are used for further training of the WGAN. Since $\mathcal{G}$ is initially not necessarily good, WOGAN also trains an analyzer which estimates input objective function values without evaluating them on the SUT. The analyzer is used to filter the inputs generated by $\mathcal{G}$: only inputs with low estimated objective are executed. The WGAN itself is trained on a subset of the collected training data. This subset is sampled by preferring low-objective inputs over ones that have high objective. The sampling changes during the algorithm execution: at the end low-objective tests are preferred more aggressively than in the beginning. For the details, see [7].

2.2 Input Representation

A scenario for the Aerialist tool [5, 6] is determined by one to four obstacles that are rectangular parallelepipeds. They are described by their widths, lengths, heights, rotations, and the locations of their bottom area centroids. We fix the height to 20 m as suggested by the competition organizers. In a valid scenario, no obstacles can collide, and they must fit into a predetermined rectangular test area. In addition, it is expected that the scenario is feasible so that the UAV can find a path. No SUT output is available for nonfeasible scenarios, and no method to test feasibility is provided in Aerialist.

Using the encoding of the obstacles as four vectors of the form $(x, y, \text{width}, \text{length}, \text{rotation})$ is straightforward, but the scenarios are frequently invalid. This poses a problem for WOGAN as it must learn to generate valid inputs. To aid the WOGAN test generation and to reduce input dimensionality, we opted to use the following alternative representation. Given two points (x_1, y_1) and (x_2, y_2) (belonging to the test area), we form a rectangle whose middle divider corresponds to the line segment determined by the two points. The width of the rectangle is fixed to 1 m. The corresponding obstacle has the rectangle as its base, and its height is set to 20 m. We have decided to always use exactly three obstacles in a scenario. Hence a scenario, or input, is determined by 12 parameters. An input is not automatically valid, but the chosen representation reduces the proportion of invalid inputs significantly.

3 WOGAN'S PERFORMANCE

We evaluate the performance of WOGAN-UAV by comparing it to a uniform random search using the same input representation. The test suites generated by WOGAN-UAV consist of 200 inputs of which the first 50 correspond to the initial random search. As the UAV scenario evaluation is slow (especially when an input is nonfeasible), running WOGAN-UAV can take as much as 21 h of real time. Due to this, we replicated the WOGAN-UAV execution only 12 times with different random seeds. Additionally we executed 700 uniformly randomly sampled inputs. Of the 700 random inputs, 26.6% revealed a fault (UAV flew closer than 1.5 m to an obstacle). The proportion of nonfeasible inputs was 27.1%. The WOGAN input generation and training time amounts to roughly 30 s per replica which amounts to 0.0% of the total time used. For WOGAN, the proportion of failing inputs in the 12 replicas was 33.7%.

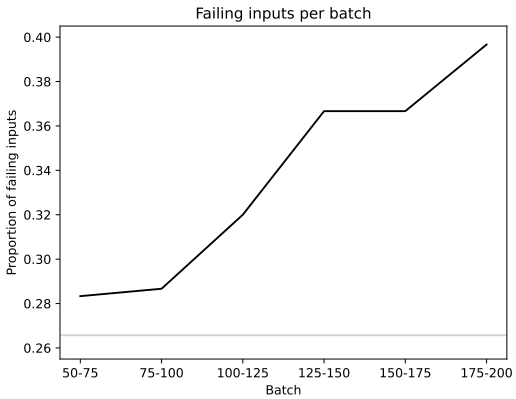

Figure 2: Proportions of failing inputs produced by WOGAN per batch. The gray line represents the proportion of failing tests for uniform random search.

To further analyze WOGAN's performance, we studied the proportion of failing inputs over batches of size 25. Our hypothesis was that the later batches, which correspond to later inputs during the WOGAN execution, have higher proportion as WOGAN learns over time to generate failing inputs to the UAV problem better. Figure 2 agrees with our hypothesis. This figure displays the observed batch proportions averaged over the 12 replicas. Initially, WOGAN generates failing inputs almost as frequently as random search, but later the frequency has clearly increased. In the batch 175–200, the proportion of failing inputs is 39.7%.

REFERENCES

[1] M. Arjovsky, S. Chintala, and L. Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 214–223.

[2] M. Biagiola, S. Klikovits, J. Peltomäki, and V. Riccio. 2023. SBFT tool competition 2023 - Cyber-physical systems track. In *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing. SBFT 2023.*

[3] A. Gambi, G. Jahangirova, V. Riccio, and F. Zampetti. 2022. SBST tool competition 2022. In *15th IEEE/ACM International Workshop on Search-Based Software Testing. SBST 2022.* 25–32.

[4] S. Khatiri et al. 2024. SBFT tool competition 2024 - CPS-UAV test case generation track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing. SBFT@ICSE 2024.*

[5] S. Khatiri, S. Panichella, and P. Tonella. 2023. Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights. In *16th IEEE International Conference on Software Testing, Verification and Validation. ICST 2023.*

[6] S. Khatiri, S. Panichella, and P. Tonella. 2024. Simulation-based testing of unmanned aerial vehicles with Aerialist. In *International Conference on Software Engineering. ICSE 2024.* To appear.

[7] J. Peltomäki, F. Spencer, and I. Porres. 2022. Wasserstein generative adversarial networks for online test generation for cyber physical systems. In *15th IEEE/ACM International Workshop on Search-Based Software Testing. SBST 2022.*

[8] J. Peltomäki, F. Spencer, and I. Porres. 2022. WOGAN at the SBST 2022 CPS tool competition. In *15th IEEE/ACM International Workshop on Search-Based Software Testing. SBST 2022.* 53–54.

[9] J. Winsten and I. Porres. 2023. WOGAN at the SBFT 2023 CPS tool competition - Cyber-physical systems track. In *16th IEEE/ACM International Workshop on Search-Based and Fuzz Testing. SBFT 2023.* 43–44.