SBFT Tool Competition 2024 - CPS-UAV Test Case Generation Track

Sajad Khatiri
Zurich University of Applied Sciences
Switzerland

Prasun Saurabh
Zurich University of Applied Sciences
Switzerland

Timothy Zimmermann
Verity AG
Switzerland

Charith Munasinghe
Zurich University of Applied Sciences
Switzerland

Christian Birchler
Zurich University of Applied Sciences
Switzerland

Sebastiano Panichella
Zurich University of Applied Sciences
Switzerland

ABSTRACT

While simulation-based testing is critical for ensuring the safety of autonomous Unmanned Aerial Vehicles (UAVs), it has not been adequately researched yet. The UAV Testing Competition organized by the Search-Based and Fuzz Testing (SBFT) workshop is an initiative designed to inspire and encourage the software testing Community to direct their attention toward UAVs as a rapidly emerging and crucial domain. It provides a simple software platform and case study to facilitate their onboarding in the UAV domain and help them develop their first test generation tools for UAVs.

In this first edition of the competition, 7 tools were submitted, evaluated, and compared extensively against each other and the baseline approach. We evaluated their test generation performance for 6 different case studies using our novel benchmarking infrastructure. The generated test suites were scored and ranked based on the number and severity of the revealed faults, and the complexity, diversity, and execution time of the test cases. This paper describes the competition context, its platform, the competing tools, and the evaluation process and results.

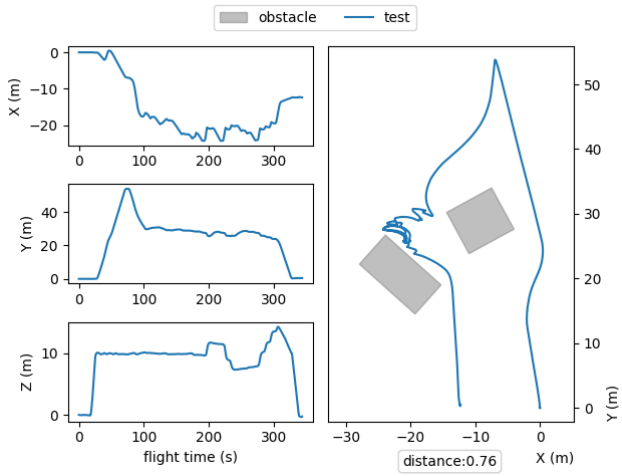

Figure 1: A failing test case

KEYWORDS

Tool Competition, Software Testing, Test Case Generation, Unmanned Aerial Vehicles, Search Based Software Engineering

1 INTRODUCTION

Simulation-based testing is critical for ensuring the safety of autonomous unmanned aerial vehicles (UAVs). Over the years, support for UAV developers has increased with open-access projects for software and hardware such as the autopilot support provided by PX4 [2, 10]. However, despite the necessity of systematically testing such complex and automated systems to ensure their safe operation in real-world environments, there has been relatively limited investment in this direction so far.

Previous studies have shown that many UAV bugs can be potentially detected before field tests if proper simulation-based testing is in place [13]. This suggests the need for further research on setting up simulation environments that test UAVs' behavior in diverse, complex, and realistic scenarios [2, 15]. However, the engineering complexity of UAVs and the physical test environment, as well as the challenges in setting up realistic simulation environments that effectively capture the same bugs, represent relevant obstacles [6, 7].

In the first edition of the UAV Testing Competition, we aim to provide software testing researchers with a simple platform to facilitate their onboarding in the UAV domain. Using the provided platform and case studies, the goal is to use search-based techniques for generating challenging test cases for autonomous vision-based UAV navigation systems. Inspired by previous SBFT tool competitions [1, 5, 11], we invited researchers to participate in the competition with their system-level test generation tools for UAVs and compete against the current state-of-the-art approach [6].

2 COMPETITION DESIGN

The objective of the competitors is to develop a test generator for our UAV system under test, a vision-based autonomous flight system called PX4. Participants are required to submit a robust test generation tool capable of generating diverse and effective test suites to uncover vulnerabilities within the system. This involves manipulating obstacle sizes and placements within the test environment, with the ultimate goal of either causing the UAV to crash or taking an unsafe path, as demonstrated in Figure 1. The effectiveness of the generated tests are measured based on the minimum distance of the drone to the obstacles during the flight.

To guarantee a fair comparison among the competing tools and to ease their development, we provided the participants with an open-source, extensible test infrastructure [7] hosted on GitHub:

https://github.com/skhatiri/UAV-Testing-Competition

2.1 Benchmarking Platform

The software under test in the competition is PX4 [10], a vision-based UAV autopilot system that has been studied in previous research [6] while the testing process has been facilitated by Aerialist [7], a UAV test bench developed on top of PX4.

2.1.1 PX4 Platform. PX4 is an open-source autopilot software stack widely used in the UAV industry for a range of vehicle types, from quadcopters to VTOL vehicles. It offers capabilities for navigation, stabilization, and autonomous mission planning and is compatible with various hardware platforms. An integral part of this ecosystem is the PX4 Avoidance module, which equips UAVs with autonomous obstacle detection and avoidance capabilities, crucial for safe operation in complex environments. PX4 internally logs comprehensive operational data and telemetry including GPS coordinates, sensor data, and flight modes which can be used for flight analysis. It also supports multiple software in the loop simulation environments (*e.g.,* Gazebo) to allow a safe and controlled development and testing environment for novel UAV control systems including mission planning and obstacle avoidance.

2.1.2 Aerialist. Aerialist [1] (unmanned AERIAL vehIcle teST bench) is a novel test bench for UAV software that automates all the necessary UAV testing steps: setting up the test environment, building and running the UAV firmware code, configuring the simulator with the simulated world properties, connecting the simulated UAV to the firmware, applying proper UAV configurations, scheduling and executing runtime commands, monitoring the UAV at runtime for any issues, and extracting the flight log after the test completion [7]. With Aerialist, competition participants have an easy-to-use platform to automate tests on the simulated UAVs, allowing them to conduct the experiments required in their test generation approach.

Aerialist models a UAV test case with a set of test properties and uses a YAML structure to describe the test including the *drone* properties (software configurations, mission plan, etc.), *simulation* properties (simulator, environment, obstacles, etc.), and the *commands* sent at runtime to the drone. It also supports large-scale experiments by deploying the test execution in a Kubernetes cluster.

2.2 Rules and Restrictions

The competition participants were expected to submit a test generator that generates challenging test cases for a given case study. The case studies are simple Aerialist test description YAML files including predefined drone configurations and a mission plan, as well as simulation environment settings without any obstacles. The test generators are only allowed to manipulate the simulation environment by adding obstacles. For simplicity, we only consider up to 4 box-shaped obstacles. An obstacle is defined by its size (length, width, height) and position in the simulation environment (x,y,z) in meters and its rotation angle (r) in degrees.

The drone is expected to safely navigate all possible environments, avoid any obstacles on the path, and complete the mission plan. Given a case study, the competition goal is to generate alternative variants of the surrounding environment (*i.e.,* obstacles) to make it more challenging for the drone to safely navigate the environment. The introduced obstacles may force the UAV to fly unsafely close to the obstacles (less than the 1.5m safety threshold investigated in previous work [6]) while still completing the mission, or even crash into them; a failing test that needs proper investigation by the UAV developers (as demonstrated in Figure 1).

Participants are expected to use search-based methods to find challenging obstacle configurations and the generated test cases (following Aerialist modeling) must respect these considerations:

- The obstacle configurations are expected to keep the mission physically feasible. The test cases that make it impossible for the UAV to find its path (*e.g.,* by creating a long wall on the path) without any safety violation are considered invalid.
- All obstacles are expected to fit in a given rectangular area as stated in the case study.
- Maximum four obstacles can be placed in the environment. They must be placed directly on the ground ($z = 0$), be taller than the UAV flight height ($h > 10m$), and must not overlap.

3 EXPERIMENTS AND RESULTS

3.1 Competing Tools

Seven tools were submitted to the competition, and with one retracted tool, six tools competed against the baseline in the first edition of the UAV Testing Competition: AmbieGen [4], CAMBA [9], DeepHyperion-UAV [17], TAIiST [16], TUMB [12], and WOGAN-UAV [14]. Similar to the previous SBFT competitions [1, 3, 5, 11], we used the state-of-the-art UAV test generation approach, Surrealist [6], as the baseline.

AmbieGen [4] leverages a surrogate (approximate) fitness function to represent system behavior and guides the test generation with an evolutionary algorithm, prioritizing test scenarios using an RRT* path planning algorithm. Intuitively, test cases with longer paths to the target should be more challenging for the UAV.

CAMBA [9] employs a cost-aware, mutation-based test case generation algorithm that sequentially learns from the previous flight logs and positions two obstacles in the environment to make the drone crash.

DeepHyperion-UAV [17] leverages the key advantages of Illumination search to find diverse misbehavior-inducing test cases (obstacle configurations) spread across the cells of a map representing the feature space of the system.

TAIiST [16] leverages the capabilities of LLMs to intelligently simulate a wide range of real-world scenarios and interactions that UAVs may encounter. This includes interpreting and responding to dynamic environmental changes, unexpected obstacles, and real-time decision-making processes.

TUMB [12] relies on Monte Carlo Tree Search (MCTS) to explore different placements of obstacles in the environment. Increasing the tree depth corresponds to adding a new obstacle to the environment, while adding a new node in the current tree level corresponds to optimizing the placement and dimensions of the last added obstacle.

WOGAN-UAV [14] is an online test generation tool based on Wasserstein generative adversarial networks. The WOGAN algorithm is a general-purpose black-box algorithm, and as such can be used on any given deterministic system that has real-valued signal inputs and outputs.

[1] https://github.com/skhatiri/Aerialist

Surrealist [6] employs an iterative adaptive greedy search approach to generate test cases that maximize a difficulty measure defined based on the drone's minimum distance to the obstacles.

3.2 Evaluation Process

The evaluation of the tools was done in two steps: First, each of the tools was used to generate a test suite for our evaluation case studies. Then, the generated test suites were evaluated and scored.

3.2.1 *Test Generation Phase.*

Case Studies. To ensure the generalizability of the test generation approaches, we evaluated the tools with 6 different case studies with increasingly challenging settings. 2 of them were derived from the sample case studies provided in the competition call (CS2, CS3) while 4 new ones were specifically created for the evaluation. The case studies vary in the number of waypoints (3-6), the shape of the mission plan (triangular, rectangular, diamond, *etc.*), and their overlap with the valid obstacle area. Figure 1 plots a sample flight in CS2 with 3 waypoints and a triangular shape.

Tool Execution. We employed various measures to ensure an identical execution environment for all the tools. We Dockerized and deployed the tools in our evaluation Kubernetes cluster with fixed resources (1.5 vCPUs, 15 GB RAM). Each tool was given a simulation budget of 200 (maximum allowed test cases to simulate) for each case study and the test cases were simulated in separate isolated containers under fixed resource allocations (6 vCPUs, 4 GB RAM). A 500-second timeout was enforced for each simulation, after which the simulation was interrupted, and logs were extracted. After their execution, the tools were supposed to output a test suite, consisting of the failing tests they found during their execution. The reported test cases were then used for the final evaluation.

3.2.2 *Test Suite Evaluation Phase.*

Due to time and computational constraints, the high number of competing tools (7), and the large size of some of the generated test suites (up to 151 tests), we limited the evaluations to 20 test cases per tool per case study (*eval_set*), randomly selected from the test suites. These test cases were first checked for compliance with the competition rules (*e.g.,* allowed obstacle area, obstacle size). Each valid test case was then executed (simulated) 5 times to minimize non-determinism effects and all of the executions were pointed independently based on the minimum distance of the drone to the obstacles (*min_dist*) during the flight according to Formula 1.

$$point(sim) = \begin{cases} 5, & \text{if } min\_dist(sim) < 0.25m \\ 2, & \text{if } 0.25m \leq min\_dist(sim) < 1m \\ 1, & \text{if } 1m \leq min\_dist(sim) < 1.5m \\ 0, & \text{if } min\_dist(sim) \geq 1.5m \end{cases} \quad (1)$$

The average point (*avg_point*) of each test case is then used to formulate its *test_score* in Formula 2. Here, we estimate the complexity of the test cases using the number of obstacles (*#obst*) in the environment and the average test execution time (*avg_time*).

$$test\_score(t) = \frac{avg\_point(t) \times 10}{\#obst(t)^2 \times avg\_time(t)} \quad (2)$$

To estimate the scores for the non-simulated test cases in suites containing more than 20 tests, we assign half of the average value of the evaluated cases (*avg_test_score*) to each of the remaining tests. This accounts for the increased likelihood of encountering similar and lower-quality test cases in larger test suites.

$$rest\_score(s) = \begin{cases} (\#tests(s) - 20) \times \frac{avg\_score(s)}{2}, & \text{if } \#tests(s) > 20 \\ 0, & \text{if } \#tests(s) \leq 20 \end{cases} \quad (3)$$

To encourage test diversity and ensure fairness in the scoring, *too similar* test cases were penalized. Two test cases are considered too similar if their flight trajectories are almost identical, calculated using the Dynamic Time Warping (DTW) distance of their average trajectories (among the 5 simulations).

$$sim\_pen(s) = 1 - \frac{\#similar\_tests(s)}{\#evaluated\_tests(s))} \quad (4)$$

The above-calculated metrics are then integrated into a final score for the whole test suite, generated by the given tool for the given case study.

$$suite\_score(s) = (\textstyle\sum_t test\_score(t) + rest\_score(s)) \times sim\_pen(s) \quad (5)$$

3.3 Results and Ranking

Table 1 summarizes the total number of reported test cases, and the obtained *suite_score* of the tools for each case study. The case study scores are summed up for each tool, and used to form the final, overall ranking of the tools. The sum of the individual case study rankings, and the number of obstacles used in test cases generated by each tool, are also reported. More details including the case study definitions, evaluated test suites, flight plots, and scoring details are included in the evaluation artifacts [8].

All of the evaluated tools were able to generate valid failing test cases ($score > 0$) for at least 3 case studies. All competitors outperformed the baseline [7] in this matter, with 3 tools being successful in all 6 case studies, and the remaining 3 tools in 5 of them. The sum of the individual tool scores for each case study can give us some insights into their difficulty. Lower values for CS4 and CS5 suggest that they were the hardest, while CS3 and CS6 with the highest sum seem to be the easiest. CS3 and CS4 report also the highest number of unsuccessful tools (2) as well.

The competition saw a tight race between the top tools, with *WOGAN-UAV* and *TUMB* leading the pack. *WOGAN-UAV*, securing the first position, achieved an overall score of 53.69 points and dominated in three of the case studies. *TUMB*, closely trailing and clinching the second overall spot with 52 points, demonstrated its strength by ranking first in two of the other case studies.

The next tools in order, *CAMBA* and *DeepHyperion-UAV*, differ from the other competitors by putting only two obstacles in the environment, making the test cases more realistic and less complicated to set up in the real world. *CAMBA* achieves a competitive performance with a score of 41.11 points, while failing in just one case study, and ranking 2 or 3 in all the others. Interestingly, *DeepHyperion-UAV* was ranked first in CS4; the hardest case study where most of the other tools scored poorly. *AmbieGen* with a close overall performance of 18.47 points, was successful in all the case

Table 1: Ranking of the tools

Tool Name	#Obst.	CS2		CS3		CS4		CS5		CS6		CS7		Rank Sum	Score Sum	Final Rank
		#	score	#	score	#	score	#	score	#	score	#	score			
WOGAN-UAV	3	61	12.55	72	14.00	81	2.35	39	8.40	71	4.81	90	11.57	12	53.69	**1**
TUMB	[1-4]	69	0.12	113	15.59	135	7.12	114	2.73	151	15.32	125	11.12	16	52.00	**2**
CAMBA	2	36	11.84	30	8.50	33	0	11	3.16	102	12.92	22	4.69	18	41.11	**3**
DeepHyperion	2	2	1.22	28	8.31	10	7.74	22	1.08	7	0	14	2.96	28	21.31	**4**
AmbieGen	[1-4]	30	2.82	46	2.00	36	0.86	65	1.22	151	10.07	30	1.51	26	18.47	**5**
Surrealist	2	10	5.03	1	0	10	0	1	0.75	19	3.88	2	0	34	9.67	**6**
TAIiST	[2-4]	7	0.81	13	0	6	0.43	11	1.57	29	1.67	22	1.33	33	5.81	**7**
SUM		215	34.40	303	48.39	311	18.51	263	18.91	530	48.67	305	33.19			

studies, succeeded by the baseline tool, *Surrealist*, with 9.67 and *TAIiST* with 5.81 points.

4 CONCLUSION AND FINAL REMARKS

This year marks the first edition of the CPS-UAV Testing Competition. We evaluated and compared six tools namely AmbieGen, CAMBA, DeepHyperion-UAV, TAIiST, TUMB, and WOGAN-UAV, and used Surrealist as a baseline. As per the results of this year, the best-performing tool is WOGAN-UAV closely followed by TUMB.

This year's competition was constrained to the deployment of box-shaped obstacles. For the upcoming years, we envisage introducing other kinds of obstacles such as trees, and buildings, as well as other environmental factors including wind and lighting conditions. Furthermore, we anticipate adjustments to the evaluation process and criteria, such as introducing a size limit for the test suites to make sure we can evaluate all the test cases. The participants will then be required to select only the top test cases.

ACKNOWLEDGMENTS

We thank the participants of the competitions for their invaluable contribution. We thank the Horizon 2020 (EU Commission) support for the project COSMOS (DevOps for Complex Cyber-physical Systems, Project No. 957254-COSMOS).

REFERENCES

[1] Matteo Biagiola, Stefan Klikovits, Jarkko Peltomäki, and Vincenzo Riccio. 2023. SBFT Tool Competition 2023 - Cyber-Physical Systems Track. In *International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE*. IEEE, 45–48. https://doi.org/10.1109/SBFT59156.2023.00010

[2] Andrea Di Sorbo, Fiorella Zampetti, Corrado A. Visaggio, Massimiliano Di Penta, and Sebastiano Panichella. 2022. Automated Identification and Qualitative Characterization of Safety Concerns Reported in UAV Software Platforms. *ACM Trans. Softw. Eng. Methodol.* (Sept. 2022). https://doi.org/10.1145/3564821 Place: New York, NY, USA Publisher: Association for Computing Machinery.

[3] Nicolas Erni, Al-Ameen Mohammed Ali Mohammed, Christian Birchler, Pouria Derakhshanfar, Stephan Lukasczyk, and Sebastiano Panichella. 2024. SBFT Tool Competition 2024 - Python Test Case Generation Track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*

[4] Dmytro Humeniuk and Foutse Khomh. 2024. AmbieGen at the SBFT 2024 Tool Competition - CPS-UAV Track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*

[5] Gunel Jahangirova and Valerio Terragni. 2023. SBFT Tool Competition 2023 - Java Test Case Generation Track. In *International Workshop on Search-Based and Fuzz Testing*. IEEE, 61–64. https://doi.org/10.1109/SBFT59156.2023.00025

[6] Sajad Khatiri, Sebastiano Panichella, and Paolo Tonella. 2023. Simulation-based test case generation for unmanned aerial vehicles in the neighborhood of real flights. In *International Conference on Software Testing, Verification and Validation.*

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

[8] Sajad Khatiri, Prasun Saurabh, Timothy Zimmermann, Charith Munasinghe, Christian Birchler, and Sebastiano Panichella. 2024. SBFT 2024 UAV Test Case Generation Competition: Evaluation Artifacts". https://doi.org/10.5281/zenodo.10605583

[9] Marco De Liso and Zhi Wen Soi. 2024. CAMBA CPS-UAV at the SBFT Tool Competition 2024. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*

[10] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. 2015. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *international conference on robotics and automation*. IEEE, 6235–6240.

[11] Sebastiano Panichella, Alessio Gambi, Fiorella Zampetti, and Vincenzo Riccio. 2021. SBST Tool Competition 2021. In *International Workshop on Search-Based Software Testing*. IEEE, 20–27. https://doi.org/10.1109/SBST52555.2021.00011

[12] Shuncheng Tang, Zhenya Zhang, Ahmet Cetinkaya, and Paolo Arcaini. 2024. TUMB at the SBFT 2024 Tool Competition – CPS-UAV Test Case Generation Track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*

[13] Christopher Steven Timperley, Afsoon Afzal, Deborah S Katz, Jam Marcos Hernandez, and Claire Le Goues. 2018. Crashing simulated planes is cheap: Can simulation detect robotics bugs early?. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 331–342.

[14] Jesper Winsten, Valentin Soloviev, Jarkko Peltomäki, and Ivan Porres. 2024. Adaptive test generation for unmanned aerial vehicles using WOGAN-UAV. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*

[15] Fiorella Zampetti, Ritu Kapur, Massimiliano Di Penta, and Sebastiano Panichella. 2022. An empirical characterization of software bugs in open-source Cyber–Physical Systems. *Journal of Systems and Software* 192 (2022), 111425. https://doi.org/10.1016/j.jss.2022.111425

[16] Taohong Zhu, William Newton, Suzanne Embury, and Youcheng Sun. 2024. TAIiST CPS-UAV at the SBFT Tool Competition 2024. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*

[17] Tahereh Zohdinasab and Andrea Doreste. 2024. DeepHyperion-UAV at the SBFT 2024 Tool Competition - CPS-UAV Test Case Generation Track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2024.*