SBFT Tool Competition 2025 – UAV Testing Track

Sajad Khatiri
University of Bern & ZHAW
Switzerland

Tahereh Zohdinasab, Prasun Saurabh
Università della Svizzera italiana
Switzerland

Dmytro Humeniuk
Polytechnique Montréal
Canada

Sebastiano Panichella
University of Bern
Switzerland

***Abstract*—Unmanned Aerial Vehicles (UAVs) are complex and critical cyber-physical systems, making simulation-based testing essential for ensuring their safe operation. Despite its importance, this field remains relatively unexplored, offering significant opportunities for further research and advancement. The UAV Testing Competition strives to actively engage the software testing community in establishing UAV testing as a rapidly emerging and indispensable domain. The second edition of the tool competition was organized jointly by SBFT and ICST, and received five submissions in total. Two of the submitted tools competed in the context of SBFT, and their test generation capabilities were assessed across three case studies and compared against each other and the baseline approach. The final ranking was determined based on the tools' effectiveness in failure detection and the diversity of the generated test cases.**

***Index Terms*—Tool Competition, Simulation-based Testing, Unmanned Aerial Vehicles, Software Testing, Test Case Generation, Search Based Software Engineering**

I. Introduction

Autonomous Unmanned Aerial Vehicles are classified as complex cyber-physical systems, and simulation-based testing is essential to guarantee their safety. Several open-access hardware and software projects, such as Ardupilot and PX4 [12], have emerged and matured over time to support the development of autonomous UAVs and their functionalities. However, testing critical functionalities in these complex systems, essential for ensuring safe and reliable operation, has lagged behind.

When conducted effectively, simulation-based testing can accurately identify bugs and issues in UAV systems, thereby significantly enhancing the safety and reliability of field testing [15]. However, accurately replicating the physical testing environment and the inherent complexity of detecting bugs that may arise in real-world scenarios remains a significant challenge [7], [8].

Software testing researchers participating in the UAV Testing Competition are provided with a structured platform designed to facilitate simulation-based UAV testing. This motivates and allows researchers to apply search-based test generation techniques in a novel and challenging domain: *autonomous vision-based UAV navigation systems*. Building upon the success of earlier testing tool competitions [4], [13], we invited researchers to take part in the competition by developing their own system-level test generation tools for UAVs and compete against each other and the state-of-the-art approach [7].

The inaugural edition of the UAV tool competition [9] in 2024 saw six teams submitting their UAV testing tools. Reflecting on the feedback and insights gained from the first edition, we made several constructive changes to make this year's competition more significant. This edition of the UAV Testing Competition (2025) was jointly organized by the SBFT and ICST communities [11]. In total, five tools were submitted to the joint call: three competing in the context of ICST (detailed in the respective competition report [11]) and two in SBFT (detailed in the remainder of this paper).

II. Competition Design

As part of the competition, participants are expected to develop a test generator for the given system under test, PX4 [12], a vision-based autonomous flight system. The submitted tool should be capable of generating diverse and effective test cases to identify underlying vulnerabilities in the system.

To identify vulnerabilities in the system under test, the submitted tool should modify the placement and size of obstacles within the simulated environment. A test will be considered valid if the UAV deviates onto an unsafe trajectory or collides with an obstacle, as illustrated in Figure 1. The generated tests are evaluated by analyzing the minimum distance between the trajectory of the UAV and the obstacles present in the environment.

Participants were given a test infrastructure tool [8] to develop their own test generator tool over the top of that:

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

A. Benchmarking Platform

PX4 [12], a vision-based UAV autopilot system, is the software under test. This open-source platform has been thoroughly examined in previous research [3], [6], [7]. The UAV test bench Aerialist [8], built on the top of PX4, facilitates the testing process.

1) PX4 Platform: PX4 is an open-source Professional Autopilot that has an active community from around the world to support it. It can control various vehicles like ground vehicles, aircraft (multi-rotor copters, fixed wing, etc.), and underwater vehicles. By providing support for different kinds of sensors, it supports navigation, autonomous mission planning, and stabilization of the UAV.

PX4-Avoidance module enables the UAV to avoid obstacles during its navigation by providing a safe path. PX4 provides in-depth flight analysis by recording comprehensive operational data and telemetry, including sensor readings, GPS

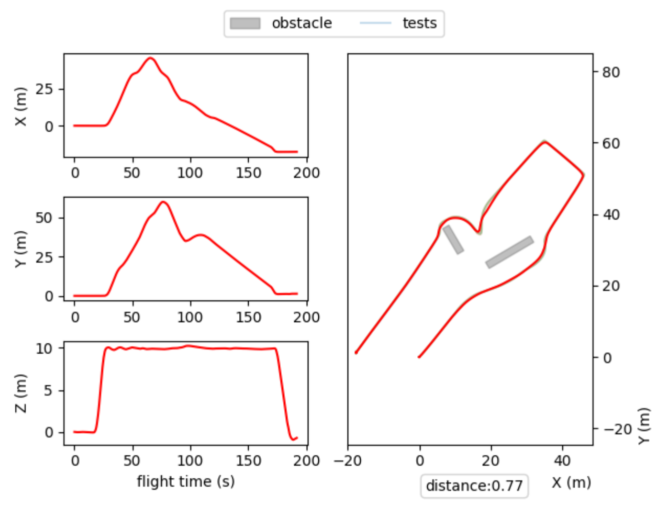

Fig. 1: A failing test case

coordinates, and flight modes. To facilitate the development and testing of features such as obstacle avoidance, advanced UAV control systems, and bug detection, PX4 supports multiple software-in-the-loop (SITL) simulation environments like Gazebo and JMAVSim.

2) Aerialist: Aerialist [1] (unmanned AERIAL vehIcle teST bench) is an advanced UAV testing platform designed to streamline and automate the testing workflow. Aerialist offers participants an intuitive and efficient platform that simplifies UAV test execution [8], allowing them to focus primarily on developing their test generation tool.

Aerialist represents a UAV test case as a structured set of test properties formatted in YAML. This structure includes *drone* properties (such as software configurations and mission plans), *simulation* properties (covering the simulator, environment, and obstacles), and *commands* executed during runtime. Aerialist can also be executed in a containerized environment using Docker, whether locally or in the cloud, with Kubernetes supporting test execution on a large scale.

B. Rules and Restrictions

The submitted test generators by the participants should generate challenging test cases and find vulnerabilities in the UAV system for a given case study.

A case study, represented in YAML format, is created by specifying an Aerialist test description, *i.e.,* a mission plan, a predefined UAV configuration, and a simulation environment without obstacles. Test generators can introduce **up to three obstacles** into the simulator. Each obstacle is defined by its dimensions (length, width, height in meters), position within the simulation environment (x, y, z), and rotation angle (r) in degrees.

The autonomous UAV should complete the mission by avoiding the obstacles placed along the trajectory in the environment. The test generator is expected to generate tests that make it hard for the UAV to navigate correctly in the environment and increase the chance of collision or following an unsafe trajectory. The obstacles introduced by the tool may cause the UAV to deviate from its planned trajectory, bringing it dangerously close—within the 1.5m safety threshold [7]—while still completing the mission, or potentially lead to a crash. These failing test cases reveal potential vulnerabilities that warrant further investigation by the UAV developers (as demonstrated in Figure 1).

Participants are required to apply search-based techniques to discover complex obstacle configurations. The test cases they generate, following Aerialist's modeling framework, must adhere to the following points:

- Any test case will be deemed invalid if it creates an impossible navigation scenario, such as placing an obstacle as significant as a long wall that prevents the mission from being completed. The path planning should remain physically feasible.
- All obstacles must be confined within the designated rectangular area defined in the case study.
- Unlike last year's edition, only up to three obstacles are allowed in the environment at a time, increasing the challenge of finding failing test cases. They must be placed directly on the ground ($z = 0$), have a height greater than the UAV's flight altitude ($h > 10m$), and must not overlap.

III. Experiments and Results

A. Competing Tools

Pseudo-random [14], and *OptObstacle* [5] are the two test generation tools that were submitted for the competition. Surrealist [7], the state-of-the-art UAV test generation approach, was used as the baseline. ICST competition report [11] introduces three other tools submitted in the joint call.

Pseudo-Random [14] leverages predictions from a lower-fidelity simulator to estimate the UAV's distance from an obstacle, running the full simulation with Gazebo only for the most promising tests.

OptObstacles [5] leverages multi-objective meta-heuristic search to generate challenging test scenarios. It optimizes the scenarios based on (1) the geometric validity of obstacle placement, (2) the position of the obstacle with respect to the expected trajectory, and (3) the distance between obstacles and UAV.

Surrealist [7] 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.

B. Evaluation Process

The evaluation of the submitted tools was conducted in two main stages: First, for each case study, the tool generated a test suite within a budget of 100 evaluations. Subsequently, the generated test suites were assessed and scored..

1) Test Generation Phase:

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

a) Case Studies: We used three case studies(CS) of increasing difficulty for evaluating the test generation approaches. They were adapted from the most challenging case studies of the previous edition [9] and vary in the number of waypoints (3,4), the degree to which they overlap with the designated obstacle area, and the mission plan shape (rectangular, triangular). A sample flight with four waypoints, a rectangular shape, and two obstacles is plotted in Figure 1.

b) Tool Execution: All submitted tools were containerized using Docker and deployed in our Kubernetes cluster with fixed resources (1.5 vCPUs, 15 GB RAM) to ensure a consistent evaluation environment for all tools. For each case study, the simulation budget for each tool was set to 100, representing the maximum number of test cases allowed for simulation. To ensure consistency, the test cases were executed in separate, isolated containers with fixed resource allocations (6 vCPUs, 4 GB RAM). For each simulation, a timeout of 500 seconds was allocated, after which the simulation was terminated, and logs were extracted. After each test case execution, the tools are expected to generate a test suite containing the failing tests found during the execution and ordered based on their relevance and criticality. These failing tests were then used for the final evaluation.

2) Test Suite Evaluation Phase: Due to time and computational constraints, evaluations were restricted to 20 generated tests per tool for each case study. These 20 tests were selected sequentially based on the output of the test generator. The selected test cases were first verified for compliance with competition rules, including obstacle size and permitted obstacle placement. The valid test cases were then executed three times to tackle the non-determinism behavior, and each of the executions was scored based on the minimum distance between the flight trajectory of the UAV and the obstacles (*min_dist*) as per the below Formula.

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

The *test_score* for each test case was calculated using its average point (*avg_point*), as shown in Formula 2. To encourage scenarios with fewer obstacles—a more challenging task—we penalized test cases with more obstacles and longer test execution times (*avg_time*) in minutes.

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

The average score $Failure\_score$ of a test suite s was calculated as the sum of all the test scores (excluding duplicates):

$$Failure\_score(s) = \sum_{t \in s} test\_score(t) \quad (3)$$

We further define the similarity of test cases based on the area covered by obstacles, as specified in Formula 4 ($area(t)$ represents the total area occupied by obstacles in a given test case).

$$Similarity(t_i, t_j) = \frac{|area(t_i) \cap area(t_j)|}{|area(t_i) \cup area(t_j)|} \quad (4)$$

The similarity between test cases is measured on a scale from 0 to 1. A similarity score of 1 indicates identical obstacle placements where the union and overlap of the areas are equal. Lower scores correspond to differing obstacle configurations. Before the final scoring, test case pairs with a similarity of 1 were marked as duplicates, and only one of them was included in the further evaluations.

We established a diversity metric based on the above test similarity to encourage test diversity. The overall test suite diversity was estimated as the average dissimilarity across all pairs n of test cases (excluding duplicates):

$$Diversity\_score(s) = 1 - \frac{1}{n} \sum_{t_i, t_j \in s} Similarity(t_i, t_j) \quad (5)$$

For each case study CS_i, tools were ranked independently as per *Failure* (R_F) and *Diversity* (R_D) scores of their generated test suites as reported in Table I. The individual ranks R_F and R_D were then aggregated to find the top-performing tool in each category (failure detection and test diversity). The final ranking was determined using Formula 6, which assigns three times more weight to the failure detection performance, as uncovering high-severity failures (i.e., those involving closer proximity to obstacles) remains the competition's primary focus.

$$R = 0.75 \times R_F + 0.25 \times R_D \quad (6)$$

C. Results and Ranking

Table I summarizes our evaluation metrics. For each case study, the total number of reported test cases, the number of failed test cases among the 20 evaluated ones (in parenthesis), the failure and diversity scores, and their ranking (in parenthesis) are reported. Individual scores and rankings are summed in the final columns, and the final tool ranking is calculated. More details, including the case study definitions, evaluated test suites, flight plots, and scoring details, are included in the evaluation artifacts [10].

All evaluated tools could generate valid failing test cases ($\#tests > 0$) for all the 3 case studies. At the same time, due to the non-determinism involved in the performance of the PX4-Autopilot agent and the simulation environment, not all reported tests that were failing during the test generation phase (simulated once) also failed in the evaluation phase (simulated three times). This remarks the importance of considering non-determinism in the test generation approaches.

Overall, *Pseudo-Random* performed well in terms of the severity of revealed failures as well as the test diversity across all case studies. It was consistently ranked first in terms of the failure score and second in terms of the diversity of the reported tests. *OptObstacles* was consistently ranked first in diversity across all case studies, making it the best tool in

TABLE I: Ranking of the tools

Tool Name	#Obst.	CS2			CS4			CS5			SUM		Final	
		#tests	failure	diversity	#tests	failure	diversity	#tests	failure	diversity	failure	diversity	Rank	
Pseudo-Random	2	31 (7)	**5.25 (1)**	0.86 (2)	66 (12)	**9.3 (1)**	0.87 (2)	17 (5)	**2.71 (1)**	0.82 (2)	**17.26 (3)**	2.57 (6)	3.75	**1**
Surrealist	2	52 (11)	4.16 (2)	0.21 (3)	9 (3)	1.93 (2)	0.20 (3)	9 (1)	0.11 (2)	0.18 (3)	6.21 (6)	0.60 (9)	6.75	**2**
OptObstacles	[2-3]	3 (1)	0.44 (3)	**0.93 (1)**	7 (0)	0.00 (3)	**0.92 (1)**	2 (0)	0.00 (3)	**0.89 (1)**	0.44 (9)	**2.75 (3)**	7.5	**3**
SUM		86 (19)	9.85	2	82 (15)	11.23	2	28 (6)	2.82	1.89				

this regard, closely followed by *Pseudo-Random*. At the same time, it only produced a few failing tests, which resulted in a low final failure score. *Surrealist* showed good performance in failure scores for *CS2* and *CS4*, securing second place in all case studies. However, it consistently ranked last in terms of test diversity.

Among the case studies, *CS2* seems to be the easiest with the highest number of reported tests and a high sum of the failure scores, while all tools had the most difficulty finding failing tests for *CS5*. In the final ranking with the weighted sum of the failure and diversity rankings, *Pseudo-Random* obtained the first place, followed by *Surrealist* and *OptObstacles* in order.

IV. Conclusion and Final Remarks

This year marks the second edition of the UAV Testing Competition, jointly organized at ICST and SBFT 2025. This year, we evaluated and compared two tools at SBFT—Pseudo-Random and OptObstacles—using Surrealist as a baseline. Based on the results, Pseudo-Random emerged as the top-performing tool. In this edition, we introduced a size limit for test suites, requiring participants to select only the most effective test cases. Additionally, we implemented a new input diversity metric based on the shape similarity of obstacles in test cases. The 2025 competition was limited to box-shaped obstacle deployments. Looking ahead, we plan to introduce new metrics [1], [2] and expand the competition to include diverse obstacle types, such as trees and buildings, and environmental factors like wind and lighting conditions.

Acknowledgments

We thank all the participants for their contribution to the competition. We thank the Horizon 2020 (EU Commission) support for the project InnoGuard, Marie Sktodowska-Curie Actions Doctoral Networks (HORIZON-MSCA-2023-DN), the Hasler Foundation for the project (Project No. 23064) entitled ”Bridging the Reality Gap in Testing Unmanned Aerial Vehicles”, and the SNSF for the project entitled ”SwarmOps: Human-sensing based MLOps for Collaborative Cyber-physical systems” (Project No. 200021_219732).

References

[1] Birchler, C., Klikovits, S., Fazzini, M., and Panichella, S. ICST tool competition 2025 - self-driving car testing track. In *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025* (2025).

[2] Birchler, C., Mohammed, T. K., Rani, P., Nechita, T., Kehrer, T., and Panichella, S. How does simulation-based testing for self-driving cars match human perception? *Proc. ACM Softw. Eng. 1*, FSE (2024), 929–950.

[3] Di Sorbo, A., Zampetti, F., Visaggio, C. A., Di Penta, M., and Panichella, S. Automated Identification and Qualitative Characterization of Safety Concerns Reported in UAV Software Platforms. *ACM Trans. Softw. Eng. Methodol.* (Sept. 2022).

[4] Gambi, A., Jahangirova, G., and Riccio, V. Message from sbft 2024 program chairs. In *2024 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT)* (2024), pp. ix–ix.

[5] Jiang, Z., and Babikian, A. A. Optobstacles at the sbft 2025 tool competition - uav testing track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2025* (2025).

[6] Khatiri, S., Mohammaid Amin, F., Panichella, S., and Tonella, P. When uncertainty leads to unsafety: Empirical insights into the role of uncertainty in unmanned aerial vehicle safety. *arXiv preprint arXiv:2501.08908* (2025).

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

[8] Khatiri, S., Panichella, S., and Tonella, P. Simulation-based testing of unmanned aerial vehicles with aerialist. In *International Conference on Software Engineering (ICSE)* (2024).

[9] Khatiri, S., Saurabh, P., Zimmermann, T., Munasinghe, C., Birchler, C., and Panichella, S. ”sbft tool competition on testing unmanned aerial vehicles”, 2023.

[10] Khatiri, S., Zohdinasab, T., Prasun, S., Humeniuk, D., and Panichella, S. ICST 2025 uav testing competition: Evaluation artifacts”. https://doi.org/10.5281/zenodo.14714469, 2025.

[11] Khatiri, S., Zohdinasab, T., Saurabh, P., Humeniuk, D., and Panichella, S. ICST tool competition 2025 - uav testing track. In *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025* (2025).

[12] Meier, L., Honegger, D., and Pollefeys, M. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *international conference on robotics and automation* (2015), IEEE, pp. 6235–6240.

[13] Panichella, S., Gambi, A., Zampetti, F., and Riccio, V. SBST tool competition 2021. In *International Workshop on Search-Based Software Testing* (2021), IEEE, pp. 20–27.

[14] Shrinah, A., and Eder, K. Pseudo-random at the sbft 2025 tool competition - uav testing track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2025* (2025).

[15] Timperley, C. S., Afzal, A., Katz, D. S., Hernandez, J. M., and Le Goues, C. Crashing simulated planes is cheap: Can simulation detect robotics bugs early? In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)* (2018), IEEE, pp. 331–342.