# ICST 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*—Simulation-based testing plays a crucial role in ensuring the safety of autonomous Unmanned Aerial Vehicles (UAVs); however, this area remains underexplored. The UAV Testing Competition aims to engage the software testing community by highlighting UAVs as an emerging and vital domain. This initiative offers a straightforward software platform and representative case studies to ease participants' entry into UAV testing, enabling them to develop their initial test generation tools for UAVs. In this second iteration of the competition, three tools were submitted, assessed, and thoroughly compared against each other, as well as the baseline approach. Our benchmarking framework analyzed their test generation capabilities across three distinct case studies. The resulting test suites were evaluated and ranked based on their failure detection and diversity. This paper provides an overview of the competition, detailing its context, platform, participating tools, evaluation methodology, and key findings.**

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

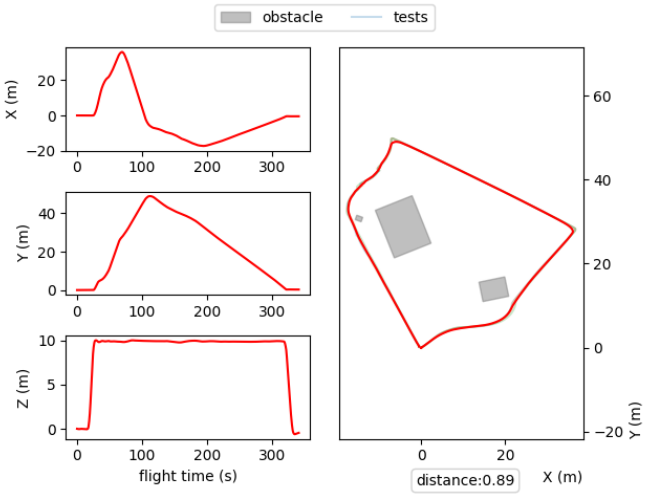


Fig. 1: A failing test case

## I. Introduction

Simulation-based testing plays a vital role in guaranteeing the safety of autonomous unmanned aerial vehicles (UAVs). Over time, support for UAV developers has expanded through open-access software and hardware projects, including autopilot support from PX4 [13] and Ardupilot. However, despite the critical need for systematically testing these complex and automated systems to ensure their safe operation in the real world, investment in this area has remained relatively limited.

Previous studies have demonstrated that many UAV bugs could potentially be detected before field testing if effective simulation-based testing is implemented [17]. However, the engineering complexity of UAVs and the physical testing environment, and the challenges of creating realistic simulations that accurately capture the same bugs pose significant obstacles [7], [8].

In this context, the UAV Testing Competition offers a simple platform for software testing researchers to ease their entry into the UAV domain. The competition aims to motivate them to apply search-based test generation techniques in a new interesting domain: *autonomous vision-based UAV navigation systems*.

Inspired by previous testing tool competitions [4], [14], we invited researchers to participate in the competition with their system-level test generation tools for UAVs and compete against the state-of-the-art approach [7].

In the first edition of the competition [9] in 2024, the initiative attracted six teams to submit their UAV testing tools and got positive feedback from the community. The UAV Testing Competition 2025 is organized jointly by the SBFT and ICST community [11], and we have put efforts into improving the competition from previous experience and feedback.

## II. 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 known as PX4. Participants must submit a robust test generation tool capable of producing diverse and effective test suites to identify potential vulnerabilities in the system.

This process involves adjusting obstacle sizes and placements within the test environment, aiming to either cause the UAV to crash or take an unsafe path, as illustrated in Figure 1. The effectiveness of the generated tests is evaluated based on the minimum distance of the flying drone to the obstacles.

To ensure a fair comparison among competing tools and facilitate their development, we have provided participants with an open-source, extensible test infrastructure [8]:

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

### A. Benchmarking Platform

The software under test is PX4 [13], a vision-based UAV autopilot system that has been extensively analyzed in previous

research [3], [6], [7]. The testing process is streamlined by Aerialist [8], a UAV test bench built on top of PX4.

*1) PX4 Platform:* PX4 is a widely adopted open-source autopilot software stack in the UAV industry, supporting various vehicle types, from quadcopters to VTOL aircraft. It provides essential functionalities such as navigation, stabilization, and autonomous mission planning while maintaining compatibility with multiple hardware platforms. A key component of this ecosystem is the PX4 Avoidance module, which enables UAVs to detect and avoid obstacles autonomously—critical for ensuring safe operation in complex environments. PX4 also logs extensive operational data and telemetry, including GPS coordinates, sensor readings, and flight modes, facilitating in-depth flight analysis. Additionally, it supports multiple software-in-the-loop (SITL) simulation environments, such as Gazebo, providing a safe and controlled setting for developing and testing innovative UAV control systems, including mission planning and obstacle avoidance.

*2) Aerialist:* Aerialist [1] (unmanned AERIAL vehIcle teST bench) is an advanced test bench for UAV software that fully automates the UAV testing process. It streamlines key steps, including setting up the test environment, compiling and running the UAV firmware, configuring the simulator with appropriate world properties, linking the simulated UAV to the firmware, applying necessary UAV settings, scheduling and executing runtime commands, monitoring the UAV for potential issues, and extracting flight logs upon test completion [8]. By using Aerialist, participants benefit from a user-friendly platform that simplifies UAV test execution, allowing them to focus mainly on their test generation approaches.

Aerialist defines a UAV test case using a structured set of test properties and represents it in a YAML format. This structure encompasses the *drone* properties (such as software configurations and mission plans), *simulation* properties (including the simulator, environment, and obstacles), and the *commands* issued to the drone at runtime. Additionally, Aerialist facilitates large-scale experimentation by enabling test execution deployment within a Kubernetes cluster.

### B. Rules and Restrictions

Competition participants were tasked with developing a test generator capable of producing challenging test cases for a given case study. Each case study consists of a basic Aerialist test description in YAML format, including predefined drone configurations, a mission plan, and simulation environment settings without obstacles. Test generators can only modify the simulation environment by introducing up to three box-shaped obstacles (for simplicity). Obstacles are defined by their dimensions (length, width, height) in meters, position in the simulation environment (x, y, z), and rotation angle (r) in degrees.

The drone is expected to navigate safely through various environments, avoiding obstacles along its path while successfully completing the mission plan. For a given case study, the objective is to generate alternative variations of the surrounding environment (i.e., obstacles) to increase the challenge for the drone's navigation. The introduced obstacles may push the UAV to fly dangerously close—within the 1.5m safety threshold [7]—while still completing the mission, or even result in a crash. Such failing tests highlight potential vulnerabilities that require further investigation by UAV developers (as demonstrated in Figure 1).

Participants are expected to employ search-based methods to identify challenging obstacle configurations. The generated test cases, structured according to Aerialist's modeling framework, must adhere to these considerations:

- Obstacle configurations must ensure the mission remains physically feasible. Test cases that make navigation impossible (e.g., by placing a long wall across the path) without violating safety constraints are invalid.
- All obstacles must fit within the specified rectangular area in the case study.
- A maximum of three obstacles can be placed in the environment (one obstacle less than the previous year to make it more challenging). They must be positioned directly on the ground ($z = 0$), exceed the UAV's flight height ($h > 10m$), and must not overlap.

## III. Experiments and Results

### A. Competing Tools

Three test generation tools named *Evolv-1* [12], *PALM* [16] and *TGen-UQ* [5] were submitted to the competition. Additionally, we used the state-of-the-art UAV test generation approach, Surrealist [7], as the baseline.

**Evolv-1** [12] employs the (1+1) Evolutionary Algorithm with customized mutation operators adjusting obstacles' position or rotation. Moreover, the Fibonacci Spiral Point Selection methodology is used to prioritize the tests for evaluation.

**PALM** [16] relies on Monte Carlo Tree Search 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. Authors have extended their previous tool [15] that was ranked second last year [9].

**TGen-UQ** [5] leverages Q-learning combined with the Upper Confidence Bound (UCB) exploration-exploitation strategy to search for challenging obstacle configurations in a simulated environment.

**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 tools was done in two steps: First, each tool generated a test suite for our evaluation case studies with a budget of 100 evaluations. Then, the generated test suites were evaluated and scored.

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

*1) Test Generation Phase:*

*a) Case Studies:* To ensure the generalizability of the test generation approaches, we evaluated the tools with three different case studies (CS) with increasingly challenging settings. They were derived from the most challenging case studies of the previous competition [9] and vary in the number of waypoints (3,4), the shape of the mission plan (triangular, rectangular), and their overlap with the valid obstacle area. Figure 1 plots a sample flight in CS4 with four waypoints and a triangular shape with three obstacles.

*b) 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 100 (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, ordered according to their criticality and importance. The reported test cases were then used for the final evaluation.

*2) Test Suite Evaluation Phase:* Due to time and computational constraints, the high number of competing tools, and the large size of some of the generated test suites, we limited the evaluations to 20 test cases per tool per case study, selected in sequential order, according to the output of the test generator. 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) 3 times to minimize non-determinism effects, and all executions were scored 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 was used to formulate its *test_score* in Formula 2. Here, we take into account the complexity of the test cases using the number of obstacles (*#obst*) in the environment and the average test execution time (*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 the test cases based on the area covered by the obstacles according to Formula 4 ($area(t)$ is the total area covered by the obstacles in the test case).

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

Where $0 \leq Similarity \leq 1$: test cases with identical placement of obstacles get a similarity of 1 (union and overlap of the areas are equal), and they get less value if the obstacles cover different areas. Before the final score calculation, the test case pairs for which $Similarity = 0$ were marked as duplicates, and one of them was excluded from evaluation.

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):

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

Tools were first ranked independently for their performance in each case study $CS_i$ according to *Failure* ($R_F$) and *Diversity* ($R_D$) scores of their generated test suites as reported in Table I. The ranks range from 1 to 4, with 1 corresponding to the best performance. Individual $R_F$ and $R_D$ ranks were then summed up to recognize the best overall tool in each category (failure detection and test diversity). The overall ranking was aggregated according to Formula 6, giving three times more importance to the failure detection performance, as revealing the failures of high severity level (closer distance to the obstacle) remains the priority of the competition.

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

### 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 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), failed also in the evaluation phase (simulated three times). This remarks the importance of considering non-determinism in the test generation approaches.

Overall, *Evolv-1* performed well in terms of the severity of revealed failures as well as their diversity across all case studies, apart from *CS4*, where it obtained the lowest failure score. It consistently scored second in terms of the diversity of the reported tests. *PALM* obtained one of the highest

TABLE I: Ranking of the tools

| Tool Name | #Obst. | CS2 | | | CS4 | | | CS5 | | | SUM | | Final Rank | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #tests | failure | diversity | #tests | failure | diversity | #tests | failure | diversity | failure | diversity | | |
| **Evolv-1** | 2 | 57 (12) | **9.02 (1)** | 0.88 (2) | 22 (2) | 1.08 (4) | 0.82 (2) | 37 (6) | **2.93 (1)** | 0.85 (2) | **13.04 (6)** | 2.57 (6) | **6** | **1** |
| **PALM** | [2-3] | 58 (1) | 0.14 (4) | 0.58 (3) | 82 (20) | **3.58 (1)** | 0.42 (3) | 76 (19) | 0.80 (2) | 0.46 (3) | 4.52 (7) | 1.47 (9) | 7.5 | **2** |
| **TGen-UQ** | 2 | 5 (1) | 0.50 (3) | **1.00 (1)** | 6 (1) | 1.44 (3) | **0.94 (1)** | 6 (0) | 0.00 (4) | **0.89 (1)** | 1.95 (10) | **2.84 (3)** | 8.25 | **3** |
| **Surrealist** | 2 | 52 (11) | 4.16 (2) | 0.21 (4) | 9 (3) | 1.93 (2) | 0.20 (4) | 9 (1) | 0.11 (3) | 0.18 (4) | 6.21 (7) | 0.60 (12) | 8.25 | **3** |
| SUM | | 172 (25) | 13.83 | 2.56 | 119 (26) | 8.05 | 2.36 | 128 (26) | 3.85 | 2.20 | | | | |

failure scores in CS4 and CS5. Moreover, *PALM* reported the biggest number of failing tests compared to the rest of the tools. At the same time, the average diversity of *PALM* tests was lower compared to the other tools, surpassing only the *Surrealist* baseline in this metric. Regarding diversity, *TGen-UQ* consistently ranked first across all case studies. At the same time, it only produced a few failing tests, which resulted in a lower final failure score. *Surrealist* showed strong performance in failure scores for *CS2* and *CS4*, securing second place in both 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 the highest sum of the failure scores, while the tools had the most difficulty finding failing tests for *CS5*. In the final ranking with the weighted sum of the failure and diversity rankings, *Evolv-1* obtained the first place, followed by *PALM*, while *TGen-UQ* and *Surrealist* share the third place with the same scores.

## IV. Conclusion and Final Remarks

This year marks the second edition of the UAV Testing Competition. We evaluated and compared three tools, namely Evolv-1, PALM, and TGen-UQ, and used Surrealist as a baseline. As per the results of this year, the best-performing tool is Evolv-1 followed by PALM. In this edition, we introduced a size limit for the test suites and required the participants to select only the top test cases. Moreover, we introduced a new input diversity metric based on the shape similarity of obstacles in test cases. This year's competition was constrained to the deployment of box-shaped obstacles. For the upcoming years, we envisage introducing new metrics [1], [2], obstacle types such as trees, buildings, and environmental factors, including wind and lighting conditions.

## Acknowledgments

We thank the participants of the competition for their invaluable contribution. 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] Javadi, A., and Birchler, C. Tgen-uq at the icst 2025 tool competition - uav testing track. In *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025* (2025). ICST Tool Competition 2025 - UAV Testing Track.

[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. SBFT tool competition 2025 - uav testing track. In *IEEE/ACM International Workshop on Search-Based and Fuzz Testing, SBFT@ICSE 2025* (2025).

[12] Lechthaler, P., Prandi, D., Kifetew, F. M., and Susi, A. Evolv-1 at the icst 2025 tool competition - uav testing track. In *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025* (2025).

[13] 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.

[14] 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.

[15] Tang, S., Zhang, Z., Cetinkaya, A., and Arcaini, P. 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* (2024).

[16] Tang, S., Zhang, Z., Cetinkaya, A., and Arcaini, P. Palm at the icst 2025 tool competition - uav testing track. In *IEEE/ACM International Conference on Software Testing, Verification and Validation, ICST 2025* (2025). ICST Tool Competition 2025 - UAV Testing Track.

[17] 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.