PhysCov: Quantifying Physical Coverage for Autonomous Drones

CS6888 - Software Analysis and Applications

Shreyes Bhat

University of Virginia, shreyes@virginia.edu

Zoraiz Qureshi

University of Virginia, zoraizq@virginia.edu

Sebastian Elbaum

University of Virginia, selbaum@virginia.edu

Thorough testing of autonomous systems for UAVs before their deployment is crucial to guarantee safety and mitigate costly failures. Existing software testing coverage metrics lack interpretability and these test adequacy metrics are not effective for complex AI driven autonomous vehicles. We introduce PhysCov, a metric to measure coverage on a Software-In-The-Loop simulation tool like Microsoft AirSim. PhysCov is measured as a proportion of geometrically unique physical scenarios or test cases obtained in the form of numeric reachability signatures recorded against the LIDAR sensor on autonomous vehicles in simulation over the total number of possible signatures. We test various scenarios on AirSim using a quadrotor equipped with a LIDAR sensor under certain defined survey tasks. Based on our evaluation, we propose that gathering high coverage with PhysCov indicates testing of an adequate number of distinct behaviors for an autonomous flying system, including uncovering potential failures such as collisions.

CCS CONCEPTS • Computer systems organization → Robotics; • Software and its engineering → Software testing and debugging

Additional Keywords and Phrases: Software Testing Metrics, Robotics, Simulation

1 Introduction

Autonomous drones have numerous applications in automated surveillance, parcel delivery, disaster response, and more. Testing for autonomous vehicles is a trending research topic, as the onboard algorithms are responsible for collision avoidance and the overall safety of the vehicle and its environment. Autonomous vehicles fall broadly into two categories, controller-driven (sensor-based actuation) and AI-driven (using image and video recognition).

Established ways of testing autonomous vehicles apart from end-to-end manual testing in the real world which is expensive in nature, include the use of software test adequacy metrics. These metrics assess the extent to which a test suite can exercise system behaviors appropriately. Some examples are code coverage of an onboard software controller and neuron coverage for deep neural network (DNN) driven vehicles [1]. Tian et al. proves that neuron coverage as a metric can be used to measure the accuracy of the decision taken by the autonomous vehicle, postulating that similar decisions fire similar neuron activations in a DNN [2]. Code coverage includes testing the code for completeness using general testing strategies like unit testing or static analysis to expose failures [3]. However, these metrics do not guarantee completeness in a real world use case, and cannot effectively test system behaviors that may lead to failures e.g. crashes from collisions. Another approach to this would be scenario testing which can better group tests that give rise to similar system behaviors, where hypothetical scenarios can be tested in simulations, but not much work has been done on this in the domain of autonomous drones.

To gauge coverage physically, we would require extensive testing with humans in the loop, and failures can be expensive since hardware may need replacing if the drone crashes. Moreover, it is also a challenge to appropriately limit any gain in physical coverage as it may not be finite in nature given continuous readings can be obtained at very small timepoints and distance measures such as from LIDAR (Light Detection and Ranging) sensors in the case of UAVs in the real world which is only limited by hardware precision in measurement and storage. However, if certain approximations are imposed over these continuous ranges it could be not only limited, but it should also be possible to group test cases together based on scenarios.

Test cases for autonomous drones can be designed and run in software-in-the-loop (SITL) and hardware-in-the-loop (HITL) simulations, where a multirotor drone can be simulated in a well-defined world with objects. However, when it comes to testing drones, collision avoidance is given importance without quantifying the area

coverage of the drone in the world. We extend the quantification of the existing physical coverage metric PhysCov to the domain of UAVs using signatures representing the sensed reachability of the drone with the help of simulated 3D LIDAR sensors. We hypothesize that a high physical coverage without collisions in the simulation translates to safer autonomous flying in the real world, Hence, we propose examining the following research questions - "*Is it possible to compute quantifiable physical coverage by scaling the PhysCov measure to the domain of UAVs in real-time? Is this optimized test adequacy measure significantly effective in assessing the effectiveness of a real-time simulation run in testing an autonomous UAV flying system with obstacle avoidance? Can it effectively gauge that distinct system behaviors that may lead to actual failures e.g. collisions from crashes have been tested in the simulation?"*

2 APPROACH

To enable PhysCov generation, we project geometrically equidistant rays from the UAV origin which can go upto the maximum range of this reachability set of the drone or get obstructed by obstacles, forming a combination of 3D rays of different lengths which is denoted as an 'RSR' (Reduced Sensed Reachability) signature - each potentially identifying a unique physical coverage scenario or test case encountered. A LIDAR sensor generates the drone's sensed environment - a spherical subset E' of the complete world environment E. For an autonomous UAV, a test suite would consider not only the external environment it is exposed to obtained using the LIDAR but also the internal state of the vehicle represented by its kinematics-dynamics state. Since the drone can only reach a certain set of given coordinates in a specific time frame from its original position given its initial kinematics state due to its physical or software-configured hardware limitations, the actual reachability set R is computed as a further smaller subset of the sensed environment E'.

$$PhysCov = \frac{N}{(R/g)^x}$$

PhysCov is computed as a ratio of the number of unique RSR signatures found *N*, at a specific time point during the simulation, to the total number of possible signatures, computed given granularity *g* and projected beams count *x* which is also the length of the RSR signature (as in RSR-x) - both of which are configurable parameters. In the equation, granularity g identifies the number of approximated distance steps the maximum beam possible length (same as LIDAR range) is divided into and R defines the max reachable range of the drone given one timestep based on its reachability set. Therefore, this fraction generally represents the number of unique physical scenarios (or test cases) the drone has covered over the total number of possible physical scenarios, based on RSR signatures.

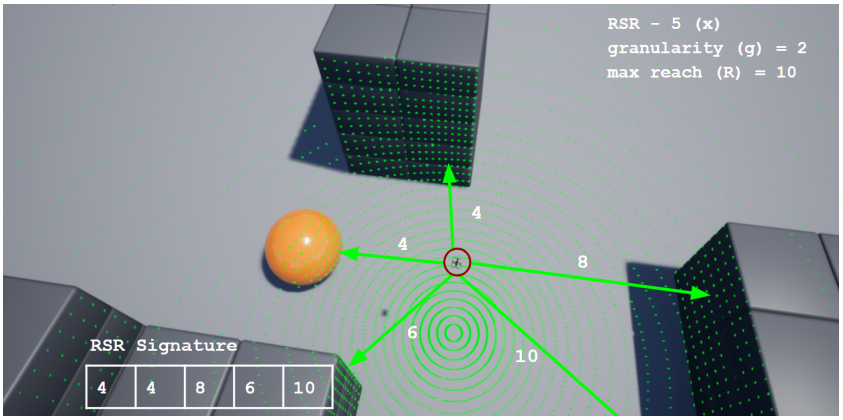

Fig 2.1. Sample RSR signature with RSR-5, granularity g=2 and reachable range = 10 in AirSim

For example, in Figure 2.1, if the max reachable range (R) is 10m and a granularity (g) of 2 is selected then the steps [2, 4, 6, 8, 10] would be used as bins to approximate each of the beam lengths generated by rounding them to the nearest step based on granularity (0 is excluded as that is not physically possible, being the drone center). Note that here we assume that R is always constant, because we over-approximate the drone's reachability set to a simpler spherical range, assuming the drone moves at a constant speed e.g. 10 m/s to any point in a spherical radius in one timestep (1s). Reachability sets for UAVs have been computed using several existing methods, e.g. getting kinematics state data such as position, velocity, altitude and angular velocity of the drone at a specific time to compute resultant planar velocities using the Kinematics-Dynamics (KD) model, then taking the convex hull of all possible future coordinates [5]. However, for our simplicity, we make this assumption of over-approximation which is also a limiting factor, as by doing this, an excess number of RSR signatures will be found that are not meaningful given that they lie outside of the reachability set of the

drone, thereby, reducing the overall accuracy of PhysCov gained during a simulation in estimating the actual proportion of potential test cases covered.

Since there is a lot of data collection involved using different environments to verify the effectiveness of such coverage metrics and streamlined APIs to test, we use software-in-the-loop SITL based simulation in AirSim - an open-source cross-platform simulator maintained by Microsoft [4], using the default 'simple_flight' flight controller. To ensure that our measurements are scalable to actual hardware, for the LIDAR sensor, we configure specific parameters for this sensor such as range, points per second, channels and and field-of-view in order to depict actual survey-grade sensor hardware present currently - the 360 degrees field-of-view CL-360 compact LIDAR scanner [13].

To generate geometrically equidistant 3D direction vectors projected on our reachability set, which would represent as our sensing rays or RSR beams, terminating at the instance a physical obstacle is detected. We assume that an autonomous drone system can move with max degrees of freedom so these rays must be spatially distributed over a sphere. For this task, we utilize the help of a prior work that does something similar for the case of generating a simulation of flocks of boids with autonomous collision avoidance [7]. It first generates a specified number of these beam direction vectors using angle increments with the golden-ratio, also known as the golden spiral method or fibonacci sphere [8]. The next step is to truncate these rays up to any intersecting segmentations of objects sensed by the LIDAR within the reachability set of the drone, using 3D sphere casting, to detect collisions ahead of a ray by investigating a specified spherical radius for neighboring physical obstacles (obtained as 3D points from the LIDAR representing a point-cloud) along the ray, using a radius of 1m [9]. This was done with a vectorized approach in Python but the computation of an RSR signature at a certain time step still takes 250ms on average, which is a limitation as it is very slow compared to LIDAR scan frequency (50 Hz) in an actual environment. However, we assume that there are no environmental changes in smaller intervals than this computation time that impact our PhysCov collection accuracy significantly.

3 Evaluation

Potentially, the best case scenario to evaluate such a test adequacy metric would be evaluated on an actual autonomous system on similar or a wider variety of survey tasks in SITL and then tested live on HITL simulation to verify that the results replicate appropriately - if sufficient resources and time are available. We hypothesize that under the same configurations for RSR and sensors such as LIDAR for a given task, if 100% PhysCov is achieved for an autonomous UAV system, without any collisions inside software then the results should replicate in an actual real setting on hardware - indicating the effectiveness of PhysCov as a test adequacy metric. However, the AirSim quadrotor does not provide its own sample collision avoidance or autonomous system. We attempted using existing implementations in the domain that were already integrated with this simulator e.g. prior projects using deep reinforcement learning and convolutional neural networks, however, none of them showed promising results - they were either overfitted to their environments and obstacle types or could not handle dynamic goals such as a survey path. Hence, we resorted to manual runs for gathering real-time coverage data and an autonomous playback mode to replay the recorded 3D coordinate path from a manual run.

Fig 3.1. High altitude road survey (S1)

Fig 3.2. Low altitude road survey (S2)

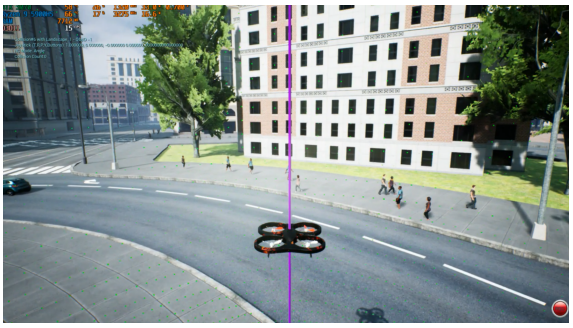

Fig 3.3. Repeated take offs and landing with varying altitudes (S3)

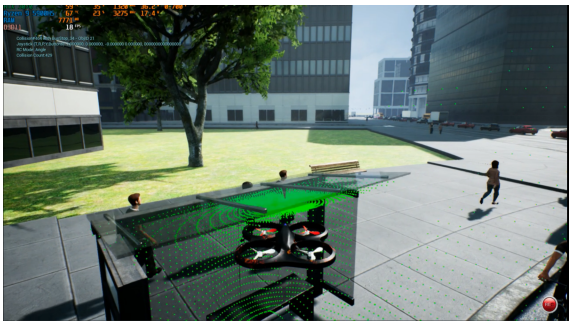

Fig 3.4. Low altitude flying with collisons (S4)

Firstly, to test the effectiveness of PhysCov as a test adequacy metric, we used four distinct simulation scenarios in the predefined City environment in AirSim, with densely packed objects simulating an urban area. We tested the following for scenarios: (S1) High altitude map survey along roads without collisions (Fig 3.1), (S2) Low altitude map survey along roads without collisions (Fig 3.2), (S3) Repeated take-offs and landings at the same position with varying altitude (Fig 3.3) and (S4) Low altitude map survey along roads with collisions with nearby objects (Fig 3.4). The controlled configuration used for these experiments were drone velocity set to 6 m/s (yielding a 6m reachable range), survey time period to 1000 simulation timesteps (based on PhysCov scans), beam count set to 5 (RSR-5), granularity set to 2 and map density kept constant.

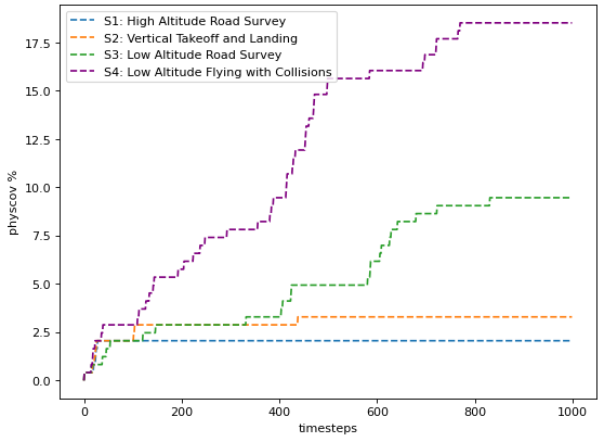

Fig 3.5. Comparing PhysCov in Scenario S4 with number of unique collisions

Fig 3.5 shows the trends in coverage as they were continuously measured during the scenario testing. Surveying at high altitude (S1) resulted in the lowest total coverage, peaking at 2.3% at the final time point as the drone encountered no objects in its reachable range at any point during the simulation except the initial take-off. Repeated take-offs and landings (S2) yields marginally better coverage at 3.5% as the drone encounters more objects in its immediate surroundings. However, the curve flattens as it does not find enough unique RSR signatures, as in such a scenario only the 2 beams projected at the bottom (out of 5) are affected, producing unique signatures over timesteps with varying altitude. The low altitude survey along roads while avoiding collisions (S3) resulted in a peak coverage of 9.8%, as the drone encounters unique RSR signatures from the road and objects such as vehicles, buildings and pedestrians in its reachable range, and repeating this survey while permitting collisions with nearby objects (S4) yields much higher PhysCov at 18.2% as the drone records a larger number of unique RSR signatures from passing close to or colliding with objects in its surroundings so more RSR beams are affected. The drone was flown manually in these simulations and only covered the roads which form a small portion of the map that consists of buildings, parks and parking lots. However, it is intuitive to observe that a drone that can survey the entire map autonomously, flying close to obstacles without colliding into them could have a 100% Physcov peak value, indicating that the autonomous drone has a perfect collision avoidance system (given an appropriate configuration of parameters such as beam count and granularity according to the task).

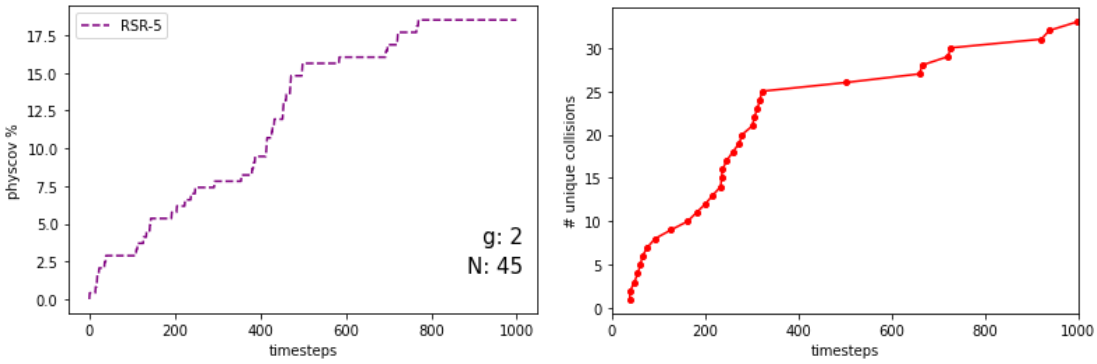

Fig 3.6. Comparing PhysCov in Scenario S4 with number of unique collisions

To further validate whether the metric can approximately test failures - specifically collision-based crashes, we test possible failure cases in our last scenario S4. We assume that uniqueness of the collision can be simply identified by the angle of the collision with respect to the drone; In our analysis, a collision is termed as unique if a vector drawn from the center of the drone to the point of impact of the collision with an object - is at least 20° apart from unique collision vectors already recorded in the 3D space. We posit that this is a reasonable threshold since the drone could register numerous collisions with the same object at small variation in angles, but a steering actuation of 20° in any direction by an autonomous steering system can avoid obstacles in most cases. Coverage is also observed to increase before the collision actually happens, implying that collisions are not necessary to obtain higher coverage and it only signifies that getting too close to an object provides certain distinct RSR signatures with smaller values and these are scenarios where a collision would be imminent. Fig 3.6 shows the correlation between PhysCov and unique collisions observed in S4. It is observed that increase in coverage is proportional to increase in number of unique collisions, as each unique collision results in a unique RSR signature.

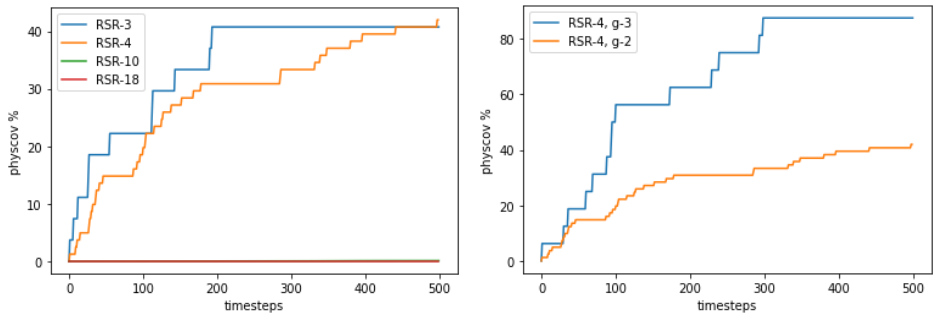

Fig 3.7. Left (a): Varying beam count for different RSR signatures for autonomous playback runs. Right (b): Varying granularity for RSR-4.

Finally, varying both beam count (x) and granularity (g) as parameters offers more flexibility, but optimum values must be found so that the coverage metric represents testing real-life situations and edge cases for the decision-making algorithms that power such autonomous drones. Figure 3.7 shows the impact of different x and g values for these parameters in each session. We utilized RSR-5 with a granularity of 2 for the previous experiments were obtained from empirical analysis, iteratively testing different values and this proved to show the most interpretable and adequately significant coverage gains given that our simulation runs were limited in timesteps. It is evident that with higher values of x as more beams are being utilized and lower values of g as smaller and more precise distance steps are used for rounding to mark uniqueness of an RSR signature, the total number of possible RSR signatures increases. Hence, we can see in Figure 3.7.b that RSR-4 with a granularity of 2 only gains less than half of the final coverage (~40%) compared to granularity of 3 (~85%), after the same number of timesteps at the timesteps as there is a larger number of possible unique RSR signatures that can be generated so the search space is larger and takes more time and distinct physical scenarios to exhaust. Given the flexibility of this configuration, it should also be possible to utilize these RSR generation parameters to generate custom test cases dynamically and automatically (e.g. moving 3D obstacles) to stimulate PhysCov gains all the way to achieving 100% coverage and exhaustively testing autonomous UAV systems e.g. for obstacle avoidance.

4 Conclusion

In terms of the methodology, our reachability set computation was over-approximated to a constant spherical region which affects the precision of PhysCov in predicting actually possible test cases. For future work, efficiently computing the precise reachability set dynamically in real-time using the Kinematics-Dynamics state of the drone will be another challenge to solve this issue. Secondly, not all of these RSR beams are utilized properly in most autonomous UAV tasks - especially the ones projected towards the back of the drone (opposite to motion) especially if the drone is configured to move in a forward-only direction. It should be possible to project only forward-oriented RSR beams for such scenarios - some angular threshold apart in the direction vector of the drone's movement limiting the search space, to focus only on obstacles ahead. The current scenarios would take a longer time to test under simulation, so we cannot expect any high saturation in PhysCov using them. Hence, automated dynamic test generation aimed specifically at increasing coverage could be done based on RSR configurations used in a 3D simulator to efficiently test how an autonomous system distinctly behaves against all possible RSR signatures. Finally, PhysCov could be integrated with the C++ AirSim API right into AirSim to reduce computational overhead for more accurate live coverage.

AirSim is designed to test autonomous vehicles in a setting as close as possible to real world physics and conditions. Our test runs for the scenarios were manual so we could not evaluate over a longer period of timesteps under the simulation. Even though future evaluation on an autonomous system under longer periods of simulation and more diverse scenarios is necessary, under the same configurations for RSR and sensors such as LIDAR for a given task, if high PhysCov is achieved for an autonomous UAV system without any collisions in the simulator, then the results should replicate in a real setting - indicating the effectiveness of PhysCov as a test adequacy metric. Specifically, the scenarios showcase that PhysCov increases proportionally with the presence of a more diverse test suite, showcasing that it can assess the extent to which it exercises distinct behaviors from an autonomous UAV system.

REFERENCES

[1] Sun, Youcheng & Huang, Xiaowei & Kroening, Daniel. (2018). Testing Deep Neural Networks.

[2] Tian, Yuchi, Kexin Pei, Suman Jana, and Baishakhi Ray. "Deeptest: Automated testing of deep-neural-network-driven autonomous cars." In Proceedings of the 40th international conference on software engineering, pp. 303-314. 2018.

[3] Sykora, Krystof & Ahmed, Bestoun & Bures, Miroslav. (2020). Code Coverage Aware Test Generation Using Constraint Solver.

[4] Shah, Shital & Dey, Debadeepta & Lovett, Chris & Kapoor, Ashish. (2017). AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles.

[5] Hildebrandt, Carl & Elbaum, Sebastian & Bezzo, Nicola & Dwyer, Matthew. (2020). Feasible and stressful trajectory generation for mobile robots. 349-362. 10.1145/3395363.3397387.

[6] Hildebrandt, Carl & Elbaum, Sebastian & Bezzo, Nicola. (2020). Blending kinematic and software models for tighter reachability analysis. 33-36. 10.1145/3377816.3381730.

[7] Reynolds, Craig. (1987). Flocks, Herds and Schools: A Distributed Behavioral Model. Computer Graphics. 21. 25-34. 10.1145/37402.37406.

[8] González, Álvaro. (2010). Measurement of Areas on a Sphere Using Fibonacci and Latitude–Longitude Lattices. Mathematical geosciences. 42. 49-64. 10.1007/s11004-009-9257-x.

[9] Goslin, Mike & Mine, Mark. (2004). The Panda3D Graphics Engine. Computer. 37. 112 - 114. 10.1109/MC.2004.180.

[10] QuadrotorAirsimDRL. “sunghoonhong/QuadrotorAirsimDRL: Deep Reinforcement Learning for Airsim Environment” GitHub, https://github.com/sunghoonhong/AirsimDRL.

[11] AirSimNeuralNet. “guillemhub/AirSimNeuralNet: Collision Avoidance Using RL.” GitHub, https://github.com/guillemhub/AirSimNeuralNet.

[12] Bondi, Elizabeth & Joppa, Lucas & Tambe, Milind & Dey, Debadeepta & Kapoor, Ashish & Piavis, Jim & Shah, Shital & Fang, Fei & Dilkina, Bistra & Hannaford, Robert & Iyer,tArvind. (2018). AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs. 1-12. 10.1145/3209811.3209880.

[13] CL-360 Compact LIDAR Scanner, Teledyne Optech, https://www.teledyneoptech.com/en/products/compact-lidar/cl-360/