# An Experience Report on Requirements-Driven Model-Based Synthetic Vision Testing

**Markus Murschitz** and **Oliver Zendel** and **Martin Humenberger** and **Christoph Sulzbachner** and **Gustavo Fernández Domínguez** [1]

**Abstract.** In this work we show how to specifically sample domain parameters - for a certain system under test (SUT) - to create corresponding test data in order to find the system's limits of operation and discover its flaws. The SUT is part of an aerial sense and avoid system that performs aerial object detection in a video stream. In order to generate synthetic test data, we first define the variability range of the test data based on real-world observations, as well as the problem specification as requirements. Then synthetic test data with explicit situational and domain coverage is generated to show how it can be used to identify problems within the tested system. Next, we show how to specifically sample domain parameters to create corresponding test data which allows us to find the operation limits of the system under test. Finally, we verify the gained insights and therefore the methodology in two ways: (i) By comparing the evaluation results to results obtained with real-world data, and (ii) by identifying the reasons for certain shortcomings based on tested SUT internals.

## 1 INTRODUCTION

Nowadays, computer vision (CV) systems are increasingly used in applications, which can either be potentially harmful to humans or are a safety measure to prevent accidents. CV systems are considered as hard to test, due to high complexity of the algorithms, variety of inputs, as well as large numbers of possible results and internal states. It is of utmost importance that the CV and testing community settle on a common standard regarding testing-procedures and concepts to safely open new application fields.

In our opinion one of the most neglected points in vision testing is that CV applications have to be tested as a whole: the system and the environment it is working in. Normally, there are two ways to improve the result quality of any real-world CV application: (i) to optimize the vision system, and (ii) to increasingly control the scene. While the first goal is concerned with camera optics components and algorithms, the second goal restricts the number of possible inputs (i.e. the domain) and, thus, reduces applicability. As a result, the requirements that are to be satisfied by a CV application have to include both: domain aspects as well as functional aspects.

During the course of this work we will focus on the specific application of aerial obstacle detection as system to be tested (see Figure 1): A single camera mounted on a small propeller-driven light aircraft is producing a continuous image stream. The SUT's goal is to detect other planes in this image stream, which are on a potential impact course. Further detections are the input for a *sense and avoid* [1] controller. In order to formulate the requirements, it is important to analyze the system with its full purpose in mind even though this work is only testing the sense part.

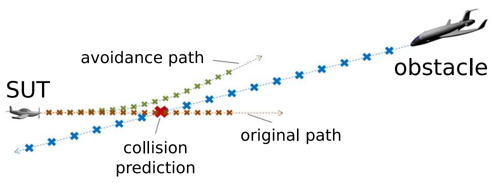


**Figure 1.** The SUT : A sense and avoid algorithm for aerial vehicles

Naturally, a failure of such a system leads to dangerous situations, hence rigorous testing is obligatory. To follow the idea of test driven development (see *e.g.* Boehm *et al.*[5]), test data is being designed in an early stage, when a complete system is unavailable (neither hardware nor software). At this early development stage, exists a general idea and a basic set of requirements about what the system should be capable of, and some potential internals (used methods of the algorithm) are already determined. To test a system based on such definitions is called *Gray Box Testing*, in contrast to *White Box Testing*, where the entire code of the system is available, and *Black Box Testing*, where only the general functionality of the algorithm is available to the testers.

The research objectives of this work are:

- to design synthetic test data based on the SUT specification together with domain restrictions in order to reveal possible flaws of the SUT and to answer system design relevant questions[2],
- to analyze the SUT's performance for certain partitions of this test data, and derive a number of insights into the SUT's weaknesses,
- to show how operational limits can be determined by the approach,
- and to verify if the approach is feasible, and if it leads to reasonable results by comparing the gained insights to real-world examples and analyzing SUT's internals.

The paper is organized as follows: Section 2 summarizes the related work and discusses how the proposed procedure differs from other approaches. Section 3 presents the approach for designing test data based on requirements of a concrete system. Section 4 describes the general procedure to create the actual test data and Section 5 presents obtained test results. Section 6 shows how the presented approach is validated while Section 7 summarizes the findings and finally, Section 8 concludes the topic and gives an outlook.

[1] AIT Austrian Institute of Technology, Austria, email: [Markus.Murschitz.fl, Oliver.Zendel, Martin.Humenberger, Christoph.Sulzbachner, Gustavo-Javier.Fernandez]@ait.ac.at

[2] The concrete formalization of the domain model written in a specific domain language, and many details on the creation of test data from this model are omitted, because it would exceed its scope.

## 2 RELATED WORK

Many advances in computer vision are closely related to available labeled data. For example, increasingly difficult test data allowed the community to analyze new approaches and rate them. Moreover, training data is constantly growing in variability, which is driving the currently impressive advances in deep-learning-based vision. Increasingly complex and larger real-world test data sets require, dependent on the ground truth type, either a considerable amount of annotation work or complex vision-independent measurements (e.g. Light Detection And Ranging (LiDAR)). To manage manual annotation effort, several strategies have been developed: many works approach the issue by crowd sourcing technologies [8, 20, 10, 16, 19] and, or semi-supervised methods [3, 29]. Other works fully concentrate on synthetic test data, due to the reduced effort in ground truth generation [27, 13, 12, 24, 4, 6].

There is an ongoing discussion about real-world versus synthetic test datasets, since it is not entirely clear if synthetic test data can replace real-world test data. Naturally, on the one hand, testing is aiming to reflect real-world behavior as close as possible. On the other hand, today's demand for test data and especially for training data often cannot (at least with manageable effort) be fulfilled by real-world data sets [22]. Biedermann *et al.* [4] present evaluations of Advanced Driver Assistance Systems (ADAS) on their fully synthetic, physically accurate rendered COnGRATS dataset and claim that their synthetic data is not ”simpler” than real-world data.

Ros *et al.* [27] generate synthetic data for semantic segmentation of urban scenes that can be used to test and train systems. Their test data design is based on the requirements of visual ADAS working in an urban environment. It is especially tailored for training deep convolutional neural networks, which need data that is sufficiently diverse to learn many parameters. The authors argue, that pixel-based human-made annotations (e.g. ImageNet [28]) are still a driving factor in system development, but are also simply too expensive for more complex applications, such as those required for ADAS.

Regarding the domain of aerial vehicles (AVs), Ribeiro and Oliveira [26] show a system to test a roll attitude autopilot system, by controlling a flight simulator, named X-Planes with Matlab/Simulink, but they do not deal with domain or situational coverage.

### 2.1 Vision Testing Requirements

Konderman [17] first analyzes how ground truth is currently generated and further elaborates on the current shortcomings of ground truth data design. He sees the aim of performance analysis in understanding, under which circumstances a given algorithm is suitable for a given task and finds requirements for engineering to be the key for better suited ground truth and test data. Finally, he presents a requirements analysis for stereo ground truth.

Regarding computer vision robustness, an extensive list of situational requirements has been presented by Zendel *et al.* [31]. They perform a risk analysis method called Hazard and Operability Study (HAZOP) and apply it to CV for the first time. This method is employed to generate a checklist for test data and includes many potentially performance hampering vision situations (criticalities). Therefore, the checklist constitutes a robustness coverage metric, which can be used to validate test data.

Model-based testing is a well established method to generate a suite of test cases from requirements [7, 9]. It aims at automatically generating input and expected results (ground truth) for a given system or software under test, such that a certain test purpose can be achieved. This purpose defines the capabilities or properties of the system to be tested. In order to enable a detailed specification of what has to be tested and to measure test progress, a test metric or coverage criterion is introduced.

### 2.2 Performance Metrics

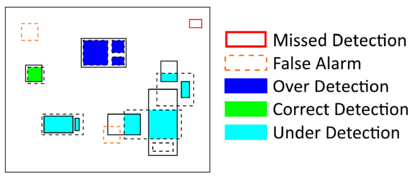


**Figure 2.** Multi Object Maximum Overlap Matching Instances (dashed strokes are the SUT result, solid ones the GT)

In evaluations on computer vision a performance metric has to be chosen dependent on the application type (*e.g.* object detection (OD), object tracking (OT) or event detection (ED)). Several sets of performance measures were proposed for different applications: VACE [15] (OD and OT), CLEAR [2] (OD and OT), CLEAR MOT [2] (OT), and the information theoretic measures [11] (OT) and CREDS [32] (ED). These metrics consist of different scores evaluating the performance with respect to the number of true positives, true negatives, false positives (or false alarms), false negatives (or missed detections), deviation errors or fragmentation depending on the target to be evaluated. Examples for more exact pixel-based comparisons besides the simple bounding box overlap measure include the Hoover Method [14] as well as the Multi Object Maximum Overlap Matching (MOMOM) of Özdemir *et al.* [25]. The Hoover Method [14] was originally proposed to calculate a performance measure for correct detections, over-segmentation, under-segmentation, missed detections and noise. MOMOM solves the basic problem of finding the best way to assign target object instances in the GT to detection instances found in the SUT's output, by modelling this as an optimization problem. The underlying assumption is that the best matching is the one that globally maximizes the overlapping regions of GT and SUT result. The results of MOMOM are a classification of detections into correct, under, over, and missed detections (see Figure 2). It also reports false alarms, *i.e.* SUT results that do not have overlapping pixels with any GT object instance. After classification, the actual performance measure based on the matches can be calculated. Özdemir *et al.* [25] use a performance measure that is sensitive to object shape and boundary fragmentation errors.

## 3 TEST DESIGN

To test a vision system, we first specify its objective (the task to be solved) and its domain (the world it has to operate in) as specific as possible. Potential sources of input for such design phase are standards, functional descriptions, requirements for the SUT, and known issues. In this case study, a group of testers and developers agreed on such definitions and decided on the type of ground truth, and therefore on what constitutes a test case (see Figure 4). Then the group defined a number of performance influencing questions which are desired to be answered by the evaluation. The most relevant, and therefore presented questions are:

- How do different backgrounds influence the number of false detections? (answered in Section 5.1)
- How does the type of an observed approaching aerial object influence the missed detections? (answered in Section 5.2)
- How do detections depend on the approaching aerial flight path and distance? (answered in Section 5.3)

One reoccurring key principle in deriving insights from evaluations is to find equivalence classes (of objects and situations) that can easily be analyzed. One possible data partition into equivalence classes of situations are scenarios. In this case, they are based on flight maneuvers (see Section 3.3).

In the following sub sections we show the exemplary definition of the SUT and its objective respectively (Section 3.1), the domain it has to operate in (Section 3.2) and the maneuver-based scenarios (Section 3.3).

## 3.1 System under Test

The purpose of the SUT to be tested in this work is a collision avoidance system to assist pilots in order to increase safety in public airspace (usually referred to as "Sense and Avoid" see Figure 1). The role of this collision avoidance system is to identify non-cooperative airborne objects (gliders, ultralights etc.) which cannot be detected by existing collision avoidance systems. Before designing the test data, various general methodologies have been discussed on how to tackle the problem: Statistical models of sky region intensity can reveal anomalous objects in sky regions. Gradient based methods can hint the detection of initial candidate flight objects. By determining the ego-motion, objects that move in relation to the camera and background can be revealed. Finally, also template based trained aerial object detectors are a possible solution. The results of any of these methods are regions within the image that can be tracked.

This work concentrates on testing the detector step only, since evaluations of tracking are more complicated (see e.g. Kristan *et al*. [18]), and would exceed the scope of this short paper. The SUT used for our evaluation is a prototype in development and states one possible way to solve the problem. The evaluation of this early prototype is a good showcase for the testing approach. However, the proposed strategies and methods for test data generation can be applied to other detectors, as well.

## 3.2 Domain

A domain definition specifies what is allowed and what is excluded within the closed world segment the algorithm has to operate in. Vision systems of today often have to operate in rather uncontrolled outdoor environments. A closed world description might not be able to include the entire range of possible situations that can occur. However, by explicitly defining the closed world, we at least exactly know the limitations of the test data. Also no coverage estimation is possible without defining the variables (their number) and their maximum and minimum bounds.

We define such world by two main ingredients: (i) Rules of existence and co-existence: which classes (e.g. which objects) exist, and by what parameters they are described. (ii) Relation rules (between object classes or instances) are usually represented as constraints on their parameters.

Part of this definition is that we define limits for parameters. In this case study we derive the following limits from the requirements of the SUT itself:

- Altitude range: 200 – 2.000m, hence no takeoff and landing

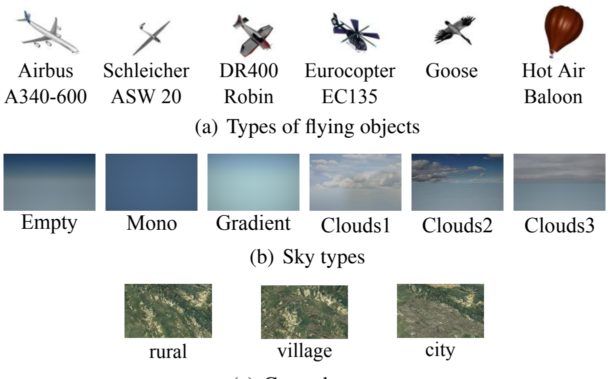


**Figure 3.** Considered entities for the generation of test cases

- Flight speed: 100 - 200km/h
- Good sight: daylight only, no flights during thunderstorms and rain, no flights within clouds

For the purpose of this work, a rather limited domain was defined: It constitutes of all possible combinations of

- 0 to 3 aerial objects (out of six possible types, see Figure 3(a))[3],
- the sky is characterized by one out of six sky types (see Figure 3(b)),
- the ground type is one out of three types (see Figure 3(c)).

Every type of aerial object is not only characterized by its appearance, but also by the rules it is subject to (*e.g*. all planes can only fly in directions similar to their forward orientation and not backwards[4]). They all have speed ranges characteristic to the respective object. Parameters such as direction, actual speed of flying objects, and speed of the SUT-plane are free, but bounded within known limits. Finally, in order to generate each test case, the entities are all integrated into a scene by combining a skydome (to represent the sky) a more or less flat ground model and the respective flying objects (see Figure 4).

## 3.3 Scenarios

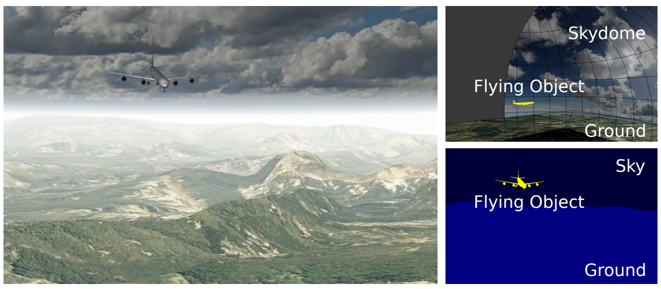


**Figure 4.** Example synthetically generated input image, scene and image ground truth of the *Head On* scenario

A scenario is a definition of how the previously defined closed world works in a specific situation. As such, a scenario can be employed to explicitly restrict the scenes (which are generated by it) to

[3] The element depicted as single goose is only one representative of the used swarm of geese.
[4] Extreme winds are excluded.

a certain subset of possible scenes.

The scenarios considered in this case study are maneuver-based:

- Head on: AV is on a head-on collision course with SUT-plane
- Converging: flight path of AV and SUT-plane intersect
- Take Over: AV is taking over SUT-plane
- Following: SUT-plane follows AV
- Parallel: AV and SUT-plane have parallel flight paths
- Null: No AVs are visible

Within a scenario, the constraints on entities are constant for all test cases. For example, in a takeover scenario the starting point of any AV has to be behind the starting point of the SUT. The flight direction is limited as well, so that the path of the AV crosses the field of view of the SUT-plane. The coverage in respect to these situational aspects is by definition guaranteed simply by defining numerous of these scenarios such as takeover, crossing, heading etc. and generating test data for them[5].

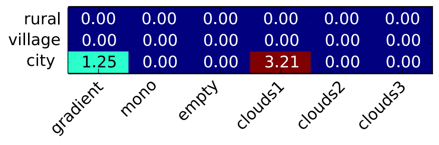
A scenario only limits the possibilities according to certain constraints, but it still contains numerous variable parameters such as: which AV is used, where it starts exactly, or the exact direction of movement. All these free parameters span a high dimensional parameter space. Many of those parameters are continuous; a test case on the other hand is only a discrete sample of the parameter space.

## 4 TEST DATA GENERATION

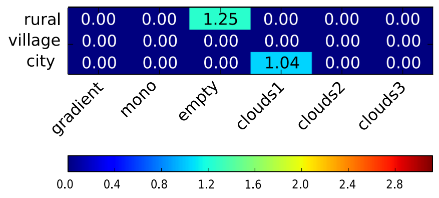
Describing the entire procedure of test data generation from a domain description and respective models in detail would exceed the scope of this work. Some general methodologies are discussed in the following (for more details see Zendel *et al.* [30]). The variables of scene elements (defined in Section 3.2) span a parameter space. Since some variables are parameters over a continuous range, sampling is required. With the objective of optimal parameter coverage in mind, a smart sampling is advantageous. Such sampling is accomplished by employing low discrepancy (see e.g. Matousek *et al.* [21]) as a sampling method. The problem is stated as follows: With a given number of sampling points (=single test cases), how is it possible to minimize the volumes of untested regions in the multi-dimensional parameter space while observing varying conditions for each of the parameters (e.g. minimum/maximum limits and different data types) with respect to the domain model rules. This is accomplished by applying the following generation strategy:

1. The domain definition is modeled in description logic as a Satisfiability Modulo Theory (SMT) model. With an SMT model it is possible to check for satisfiability, find solutions, test solutions, and evaluate if they fit the model (see e.g. Moura *et al.* [23]).
2. Samples are taken in the parameter space according to low discrepancy sampling.
3. The samples are evaluated for validity by testing them in conjunction with the SMT model of the domain.
4. Steps 2 and 3 are repeated until a valid solution is found. If this fails for a certain number of times, a solution is generated from the SMT model. If even this fails, the domain model is invalid and not satisfiable under the requested circumstances.
5. The test case is created with the chosen selection of parameter values. It represents the initial setup of a scene as well as all variables needed to simulate its progression within the time frame. A dedicated rendering and post-processing pipeline generates images and GT labelings for specified moments in time (e.g. a sequence of 100 frames at 20 fps). An example is given in Figure 4.

[5] If there is at least one test case for each scenario.



(a) after 5 initialization frames



(b) after 5 frames with atmospheric effects

**Figure 5.** Percentage of frames with false alarms

## 5 TEST RESULTS

In order to compare the SUT output and the ground truth we need an appropriate metric. The choice of metric must be based on the task and the corresponding requirements. We chose an adaption of Multi Object Maximum Overlap Matching (MOMOM) of Özdemir *et al.* [25] tailored to the needs of aerial object detection applications. Not all of the detections' instance classes distinguished by MOMOM (correct-, missed-, over-, under-detections and false alarms; see Figure 2) are relevant for sense and avoid. Over-detections are not considered a problem, only missed detections and false alarms are erroneous outputs. Also, in comparison to the original MOMOM, the actual shape of the detected objects bears no relevance, but position and size are important.

In the following, evaluations and corresponding datasets for this case study are presented: the background evaluation (Section 5.1), the foreground / target object evaluation (Section 5.2), and finally an example for parametric analysis (Section 5.3).

### 5.1 Background Dependency Evaluation

To evaluate how the background influences the number of false alarms, we designed a test data set that does not show any aerial objects (Null test) and it varies ground and sky types (see Figure 3(c) and Figure3(b)). For each ground-sky combination 5 different flight paths, each consisting of 40 consecutive frames from a moving virtual camera with 10 frames per second, are generated. For an example see Figure 4.

In a first preliminary experiment, the initialization time is determined by analyzing the false alarm dependency on the frame number. It reveals that the number of false alarms significantly drops after four frames. Thus, we decided not to evaluate any frames with an index lower than five in the following evaluations to avoid overemphasizing the startup phase.

In a second evaluation (for the same test data), we analyze the number of false alarms per ground-sky combination, without considering initialization artifacts (see Figure 5(a)). The following observations are made:

(O1) In general, the background model of the SUT is capable of modeling backgrounds with the tested variance.

(O2) False alarms occur mostly in scenarios where the ground model contains a city scene.

(O3) The number of false alarms is reduced if simulated atmospheric effects are present (see Figure 5(b) versus Figure 5(a)).
(O4) The remainder of the false alarms are located at hard edges of mountains in the terrain.

The question arising from Observation (O2) is: why does the city lead to significantly higher false alarms compared to the other terrain elements? There are two possible reasons: (i) the SUT's background model is sensitive to elements of unexpected high frequencies or significant change in appearance in the background. (ii) The SUT benefits from two effects, which in real-world data reduce the impact of the ground on the captured image: Firstly, ground is blurred due to limited depth of field. And secondly, light-scattering particles in the atmosphere reduce image saturation. The first reason (i) would mean that this is a shortcoming of the SUT, while the second reason (ii) means the test data is in that respect "harder" than real-world data.

However, to get a less biased test result, the remaining dataset was computed with a simulated atmospheric effect (results are shown in Figure 5(b)).

From the perspective of hazard analysis, Observation (O4) exhibits no problematic behavior, but a positive side effect. Mountains and their hard edges are definitely dangerous objects for an airplane.

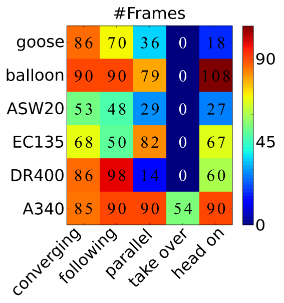
## 5.2 Flying Object Dependency Evaluation

The influence of the flying object's shape and characteristics is analyzed by investigating their effect on the number of missed detections. The therefore synthesized test set is in general comparable to the previous one, but has a single object within the field of view (FOV) of the camera in at least one frame of each test sequence. In each sequence, the object is entering the FOV and leaving the FOV over time. The variables are the starting and end position of the objects and their speed. Due to the nature of this test set (objects inside and outside of the FOV), it must be filtered before analysis: (a) the frame number must be greater or equal to five (see previous section), (b) the flying object must cover a minimal amount of pixels. The corresponding requirements to pass the filter are: a minimal number of pixels of $15^2 = 225$ which corresponds to 0.062% of the image (image size 752x480), and a frame number larger than 5. Figure 6(a) shows the remaining number of frames after filtering for each object-scenario combination.

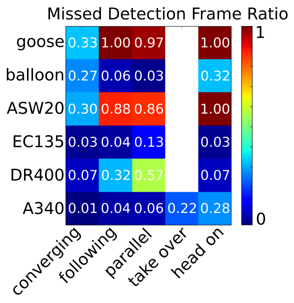
The zeros in the takeover scenario in Figure 6(a) show that not all scenarios are possible with all flying objects. They have different speeds, which approximate their real-world cruising speeds. The A340 plane is the only one faster than the SUT, hence the only one that can pass the SUT in a takeover scenario. Furthermore, in order to analyze the effect of object type on missed detections, we analyze the relative number of frames with at least one missed detection[6] (see Figure 6(b)). The following observations can be made:

(O5) In general, slower objects are less likely to be detected than faster ones.
(O6) Thin structures can be missed: The object "goose", which actually consists of several geese flying in a swarm, is not detected at all. The individual geese are usually very small. The ASW20 states a similar problem: it has a thin wing profile and therefore its silhouette has thin parts.
(O7) The big and slow balloon confuses the system. It is the slowest of all these objects.
(O8) The SUT performs much better on converging than on the other domain models.

[6] According to the MOMOM matching optimization results.



(a) Coverage: # of frames per combination



(b) Relative # of frames with missed detections

**Figure 6.** Evaluation results for all scenario-object combinations

Our initial hypothesis defining parallel flights as problematic did not prove to be correct. There is a minor reduction in performance, but it is not significant in comparison to the other scenarios. Regarding Observation (O8), it is the least restricted scenario and the most probable[7] (see coverage in Figure 6(a)). It only requires the flight paths to intersect within the SUT's field of view.

## 5.3 Parametric Analysis

Along with coverage of situations and entities (as described in the previous sections), our tool chain also supports parametric analysis. Since the head on situation is the most critical, it was decided to do a parametric evaluation on the flight path for such a scenario. Test data was generated containing a single plane of the type Airbus A340. It was set up to fly towards the SUT with constant speed from various directions (determined by low discrepancy sampling). In Figure 7, the dependency of missed detections in relation to the flight path is visualized. Obviously, the missed detections increases with distance; as can be seen in both XZ-plane and XY-plane projections. One cannot draw reliable conclusions from the YZ-plane, but the XY-plane clearly shows decreased detection rates close to the Y=0 line.

(O9) Planes heading directly towards the SUT-AV state an additional difficulty for the SUT.

## 6 VALIDATION OF THE APPROACH

In the previous section, we established a list of 9 observations about the system's behavior in the defined domain. The question remains, whether those results are only valid for the synthetic test cases or do they generalize to real-world situations. In the following, we show two methods of evaluating the testing approach for the applicability to the SUT at hand: (i) Find similar situations in real-world data recorded from an actual aerial vehicle and compare the results, (ii) find out details in the SUT's algorithm that undermine the validity of the observations.

Regarding the general observation judging the background as sufficient (Observation (O1)): the available real-world data contains various cloud formations that did not influence the performance significantly.

Regarding Observation (O2) and (O4), where mountains and cities have been detected as flying objects: Unfortunately the available captured test data does not contain any sequences with mountains, but

[7] At least within our definition of the domain.

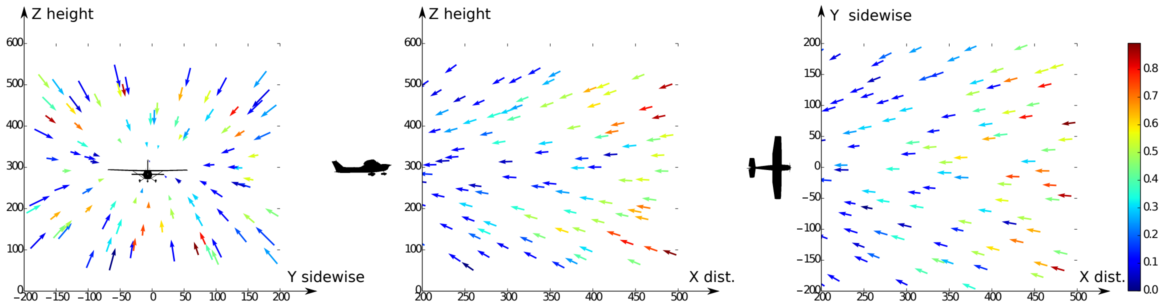
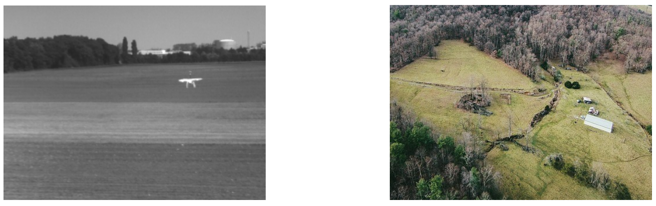


**Figure 7.** Missed detections in dependency of the flight path (the arrows signify the flight speed [m/s] and its starting point [m], the black plane is the SUT)



(a) hovering drone example

(b) house example

**Figure 8.** Real-world examples for observations on synthetic test data behavior[9]

it contains houses (see Figure 8(b)), which state a similar problem in low height flights. The SUT reacted with false positives for those scenes.

Observing that low speed AVs are a potential problem (Observation (O5)), a real-world situation was sought out in captured video material. The phenomenon could be found in a video segment showing a hovering drone (see Figure 8(a)). This video material was not annotated since the segment was not considered critical before the synthetic test run. It would have been difficult to find the segment of some hundred frames in a couple of gigabytes of data.

The problem with thin and/or disconnected silhouettes (Observation (O6)) can be explained by the internals of the SUT: in order to make it more robust, the SUT has a filtering step for small noise, which ignores such small elements. Therefore the goose swarm, which is not connected, and where each individual goose is too small to lead to detection on its own, is considered to be noise. This is simply a camera resolution dependent effect, hence the testers recommended to the system developers to increase the camera resolution.

Observation (O7), huge slow objects like the balloon can confuse the system, can be explained by the general approach of the SUT treating the task as foreground background segmentation. If an object covers the majority of the pixels of the camera image, it is assumed to be background. This causes a missed detection as well as a number of false alarms in the background.

Observation (O9), where one plain flies toward the SUT on head-on collision course, could not be found in the real-world data, but it can be explained similarly to the previous Observation (O6).

Observation (O3), concerning the atmospheric effects was already discussed in the previous section. Observation (O8) establishing that the system performs best in the converging scenario, could not be verified by any of the two means, which made even more evident why synthetic test data is needed.

## 7 LESSONS LEARNED

Whenever we test with synthetic test data we initially have to make sure that the data is ”sufficiently realistic”, meaning that the behavior of the algorithm with synthetic data is similar to real world behavior. We propose to compare results for synthetic and real-world data in an initial phase and adapt the synthetic data until similar behavior is reached (See Figure 5(b)).

We also have to ensure that we measure performance fairly and according to the test objectives (*e.g.* do not measure initialization phases).

What became clear during the evaluation of the approach is, that is time-consuming, often infeasible, and sometimes even impossible (without risk to human life) to find specific situations in real-world data. Even if the data is available, it can mean that one has to sift through gigabytes of mostly unannotated video material.

## 8 CONCLUSION & OUTLOOK

During the course of this work we depicted a case study on how to design test data based on SUT requirements and a definition of the environment the tested system has to operate in. One key element was to identify several performance-influencing questions that allow for deep insights into the shortcomings of SUT. For each of these questions test data was created, evaluations have been performed, and observations on the tested system’s behavior have been made. Finally, the validation of this testing approach could verify 8 of 9 observations with either real-world examples that cause the same erroneous behavior, or explanations based on the SUT’s internals. Therefore this test case generation procedure, combined with an appropriate evaluation method, leads to interpretable and valid results. During the course of this work the completeness of the generated test data could not be determined. The question remains: how many erroneous behaviors have not been found by this synthetic-test-data-based evaluation?

In our opinion, carefully planned test data (that includes functional and domain aspects) is vital to any complete assessment about a CV application and synthetic test data was the most feasible solution for the application at hand.

[9] The original image can not be published for legal reasons, therefore the image Figure 8(b) shows an exemplary similar scene.

## ACKNOWLEDGEMENTS

Special thanks go to our recently retired project leader Wolfgang Herzner. Financial support was provided by the Austrian Ministry for Transport, Innovation and Technology (BMVIT) in the scope of the TAKE OFF program within the DVKUP and RPA-AI projects.

## REFERENCES

[1] P. Angelov, *Sense and Avoid in UAS: Research and Applications*, John Wiley & Sons, 2012.

[2] K. Bernardin and R. Stiefelhagen, 'Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics', *EURASIP Journal on Image and Video Processing*, (2008).

[3] V. Biaud, V. Despiegel, C. Herold, O. Beiler, and S. Gentric, 'Semi-supervised Evaluation of Face Recognition in Videos', in *Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications*, VIGTA '13, New York, NY, USA, (2013). ACM.

[4] D. Biedermann, M. Ochs, and R. Mester, 'Evaluating visual ADAS components on the COnGRATS dataset', in *2016 IEEE Intelligent Vehicles Symposium (IV)*, (2016).

[5] B. Boehm, D. Rombach, H., and V. Zelkowitz, M., *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili*, Springer Science & Business Media, 2005.

[6] J. Butler, D., J. Wulff, B. Stanley, G., and J. Black, M., 'A naturalistic open source movie for optical flow evaluation', in *Computer Vision–ECCV 2012*, Springer, (2012).

[7] S. Dalal, A. Jain, N. Karunanithi, J. Leaton, C. Lott, G. Patton, and B. Horowitz, 'Model-based testing in practice'. ACM, (1999).

[8] R. Di Salvo, D. Giordano, and I. Kavasidis, 'A Crowdsourcing Approach to Support Video Annotation', in *Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications*, VIGTA '13, New York, NY, USA, (2013). ACM.

[9] C. Dias Neto, A., R. Subramanyan, M. Vieira, and H. Travassos, G., 'A Survey on Model-based Testing Approaches: A Systematic Review', in *Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies: Held in Conjunction with the 22Nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, WEASELTech '07, New York, NY, USA, (2007). ACM.

[10] A. Donath and D. Kondermann, 'How Good is Crowdsourcing for Optical Flow Ground Truth Generation?', *submitted to CVPR*, (2013).

[11] K. Edward, K., D. Matthew, P., and H. Michael, B., 'An information theoretic approach for tracker performance evaluation', in *2009 IEEE 12th International Conference on Computer Vision*, (2009).

[12] R. Haeusler and D. Kondermann, 'Synthesizing Real World Stereo Challenges', in *Pattern Recognition*, eds., Joachim Weickert, Matthias Hein, and Bernt Schiele, Lecture Notes in Computer Science, Springer Berlin Heidelberg, (2013).

[13] V. Haltakov, C. Unger, and S. Ilic, 'Framework for Generation of Synthetic Ground Truth Data for Driver Assistance Applications', in *Pattern Recognition*, eds., Joachim Weickert, Matthias Hein, and Bernt Schiele, Lecture Notes in Computer Science, Springer Berlin Heidelberg, (2013).

[14] A. Hoover, G. Jean-Baptiste, X. Jiang, J. Flynn, P., H. Bunke, B. Goldgof, D., K. Bowyer, W. Eggert, D., A. Fitzgibbon, and B. Fisher, R., 'An experimental comparison of range image segmentation algorithms', *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1996).

[15] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, R. Bowers, M. Boonstra, V. Korzhova, and J. Zhang, 'Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2009).

[16] Á. Kiss and T. Szirányi, 'Evaluation of Manually Created Ground Truth for Multi-view People Localization', in *Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications*, VIGTA '13, New York, NY, USA, (2013). ACM.

[17] D. Kondermann, 'Ground Truth Design Principles: An Overview', in *Proceedings of the International Workshop on Video and Image Ground Truth in Computer Vision Applications*, VIGTA '13, New York, NY, USA, (2013). ACM.

[18] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Cehovin, 'A Novel Performance Evaluation Methodology for Single-Target Trackers', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2016).

[19] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and L. Zitnick, C., 'Microsoft COCO: Common Objects in Context', in *Computer Vision – ECCV 2014*, eds., David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Lecture Notes in Computer Science, Springer International Publishing, (2014).

[20] L. Maier-Hein, S. Mersmann, D. Kondermann, C. Stock, G. Kenngott, H., A. Sanchez, M. Wagner, A. Preukschas, A. Wekerle, S. Helfert, and others, 'Crowdsourcing for reference correspondence generation in endoscopic images', in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*, Springer, (2014).

[21] J. Matousek, *Geometric discrepancy: An illustrated guide*, Springer, 1999.

[22] S. Meister and D. Kondermann, 'Real versus realistically rendered scenes for optical flow evaluation', in *2011 14th ITG Conference on Electronic Media Technology (CEMT)*, (2011).

[23] L. Moura and N. Bjørner, 'Z3: An Efficient SMT Solver', in *Tools and Algorithms for the Construction and Analysis of Systems*, eds., C. R. Ramakrishnan and Jakob Rehof, Lecture Notes in Computer Science, Springer Berlin Heidelberg, (2008).

[24] N. Onkarappa and D. Sappa, A., 'Synthetic sequences and ground-truth flow field generation for algorithm validation', *Multimedia Tools and Applications*, (2013).

[25] B. Özdemir, S. Aksoy, S. Eckert, M. Pesaresi, and D. Ehrlich, 'Performance measures for object detection evaluation', *Pattern Recognition Letters*, (2010).

[26] R. Ribeiro, L. and M. F. Oliveira, N., 'UAV autopilot controllers test platform using Matlab/Simulink and X-Plane', in *2010 IEEE Frontiers in Education Conference (FIE)*, (2010).

[27] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, 'The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes', in *CVPR*, (2016).

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, C. Berg, A., and L. Fei-Fei, 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision*, (2015).

[29] R. Socher and L. Fei-Fei, 'Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora', in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (2010).

[30] O. Zendel, W. Herzner, and M. Murschitz, 'VITRO - Model based vision testing for robustness', in *2013 44th International Symposium on Robotics (ISR)*, (2013).

[31] O. Zendel, M. Murschitz, M. Humenberger, and W. Herzner, 'CV-HAZOP: Introducing Test Data Validation for Computer Vision', in *Proceedings of the IEEE International Conference on Computer Vision*, (2015).

[32] F. Ziliani, S. Velastin, F. Porikli, L. Marcenaro, T. Kelliher, A. Cavallaro, and P. Bruneaut, 'Performance evaluation of event detection solutions: the CREDS experience', in *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, (2005).