Test Input Partitioning for Automated Testing of Satellite On-board Image Processing Algorithms

Ulrike Witteck[1], Denis Grießbach[1] and Paula Herber[2]
[1]*Institute of Optical Sensor Systems, German Aerospace Center (DLR), Berlin-Adlershof, Germany*
[2]*Embedded Systems Group, University of Münster, Germany*

Keywords: Image Processing, Software Testing, Input Partitioning, Satellite Systems.

Abstract: On-board image processing technologies in the satellite domain are subject to extremely strict requirements with respect to reliability and accuracy in hard real-time. Due to their large input domain, it is infeasible to execute all possible test cases. To overcome this problem, we define a novel test approach that efficiently and systematically captures the input domain of satellite on-board image processing applications. To achieve this, we first present a dedicated partitioning into equivalence classes for each input parameter. Then, we define multidimensional coverage criteria to assess a given test suite for its coverage on the complete input domain. Finally, we present a test generation algorithm that automatically inserts missing test cases into a given test suite based on our multidimensional coverage criteria. This results in a reasonably small test suite that covers the whole input domain of satellite on-board image processing applications. We demonstrate the effectiveness of our approach with experimental results from the ESA medium-class mission PLATO.

1 INTRODUCTION

Various on-board image processing applications are subject to strict requirements with respect to reliability and accuracy in hard real-time. Due to the large input domain of such applications, testing the systems manually is error-prone and time-consuming. Hence, a test approach is needed that automatically and systematically generates test cases for testing such applications. However, the automated test generation for on-board image processing applications poses a major challenge: The large amount of input parameters and their possible combinations leads to a high number of test cases. Hence, the systematic and efficient coverage of the whole input domain is expensive.

In existing work (Bringmann and Krämer, 2006; Huang and Peleska, 2016), automated test approaches for various domains, for example, for automotive and railway applications, are presented. The authors investigate huge input domains and complex functional behavior. However, their focus is on event-driven reactive real-time systems, and none of these approaches is tailored to the domain of satellite on-board image processing applications, where the complexity is in the huge amount of possible input images.

In this paper, we present a novel test approach for the domain of satellite on-board image processing applications. It is based on the (unpublished) master thesis of the first author (Witteck, 2018). The objective of our approach is to achieve a high coverage of the input domain while at the same time using reasonably small test suites. For our proposed approach, we adopt the equivalence class partition testing method. In general, this method partitions a given input or output domain into disjoint sub-domains called equivalence classes (Varshney and Mehrotra, 2014). The use of some test values as representatives of each class reduces the number of required test cases (Bhat and Quadri, 2015), while still systematically covering the respective domains. In our test approach, we specify a dedicated partitioning for each input parameter of satellite on-board image processing applications. Moreover, we define multidimensional coverage criteria for our application domain, which combines the individual coverage criteria for each input parameter and enables us to assess a given test suite for its coverage on the complete input domain. Our test generation approach uses our multidimensional criteria to automatically assess given test suites with respect to their coverage of input combinations and automatically generates missing combinations. Thus, our test approach enables efficient test case generation.

In order to investigate the efficiency of our proposed test approach, we use the Fine Guidance Sys-

Witteck, U., Grießbach, D. and Herber, P.
Test Input Partitioning for Automated Testing of Satellite On-board Image Processing Algorithms.
DOI: 10.5220/0007807400150026
In *Proceedings of the 14th International Conference on Software Technologies (ICSOFT 2019)*, pages 15-26
ISBN: 978-989-758-379-7

tem (FGS) algorithm of the ESA (European Space Agency) mission PLATO (PLAnetary Transits and Oscillation of stars) as a case study. It is a satellite on-board image processing algorithm to calculate the high-precision attitude of the spacecraft by comparing tracked star positions with known star positions from a star catalog. The experimental results demonstrate the effectiveness of our partitioning approach in terms of an increased error detection capability.

This paper is structured as follows: In Section 2, we give a brief overview of equivalence class partition testing and introduce the PLATO mission as well as the FGS algorithm. In Section 3, we outline related work. In Section 4, we present our equivalence class partitioning approach for on-board satellite image processing applications. In Section 5, we present experimental results, and we conclude in Section 6.

2 PRELIMINARIES

In this section, we introduce the necessary preliminaries to understand the remainder of this paper.

2.1 Equivalence Class Partition Testing

To make testing more efficient and less time consuming, it is preferable to examine few test cases that cover a large part of the system under test. Equivalence class partition testing offers a possible solution to this problem. It is a commonly used approach in practice. The technique partitions a given input domain or output domain into disjoint sub-domains, the equivalence classes. The method partitions the domain in such a way, that all elements in an equivalence class are expected to provoke the same system behavior according to a specification. The partitioning is based on one or multiple input parameters specified in the requirements. Equivalence classes represent subsets of parameter values that completely cover the input or output domain. For the purpose of software testing, it is therefore sufficient to test some representative values of each equivalence class.

The selection of test cases from equivalence classes can be made according to various criteria: using border values, testing special values or randomly selecting test cases (Bhat and Quadri, 2015; Huang and Peleska, 2016; Peter Liggesmeyer, 2009). The approach removes redundant test cases but retains the completeness of the tests. Hence, the approach reduces the test effort compared to exhaustive testing (Bhat and Quadri, 2015).

2.2 Context: PLATO Mission

PLATO is an ESA mission in the long-term space scientific program “Cosmic Vision” (ESA, 2012). The German Aerospace Center (DLR) manages the international consortium for developing the payload and scientific operation of the project (DLR, 2017).

The main goal of the PLATO mission is the detection and characterization of Earth-like exoplanets orbiting in the habitable zone of solar-type stars. It achieves its scientific objectives by long uninterrupted ultra-high precision photometric monitoring of large samples of bright stars. This requires a very large Field of View (FOV) as well as a low noise level. To achieve a high pupil size and the required FOV the instrument contains 26 telescopes for star observation. 24 normal cameras monitor stars fainter than magnitude 8 at a cycle of 25 s. Two fast cameras observe stars brighter than magnitude 8 at a cycle of 2.5 s. The size of the FOV of a fast camera is $38.7° \times 38.7°$. The cameras are equipped with four Charge Coupled Devices (CCD) in the focal plane, each with 4510×4510 pixels.

Each fast camera comes with a data processing unit running the FGS algorithm. It calculates attitude data with an accuracy of milliarcseconds from the image data. This data is supplied to the attitude and orbit control system. The FGS is regarded as being a mission-critical component which implies a extensive test procedure.

Fine Guidance System Algorithm. Many spacecraft missions use a FGS to obtain accurate measurements of the spacecraft orientation. We use the PLATO FGS algorithm as a case study to investigate the efficiency of our test approach.

The attitude calculation of a telescope is based on measured star positions on the CCD compared to their reference directions in a star catalog. Figure 1 gives an overview of the FGS algorithm (Grießbach, 2018).

The autonomous attitude tracking is initialized with an initial attitude given by the space craft. For each pre-selected guide star an initial sub-window position is calculated by means of the camera model, which transforms from sky coordinates to pixel coordinates and vice versa (Grießbach, 2018). Guide stars are predefined stars in a star catalog that satisfy given criteria. For example, the star magnitude is within a certain range, the star has very low contamination, etc. (Grießbach, 2018). The FGS algorithm calculates centroids after reading 7×7 pixel sub-window every 2.5 s from the full CCD image.

A linear center of mass calculation estimates the initial centroid position. To get a more precise solu-

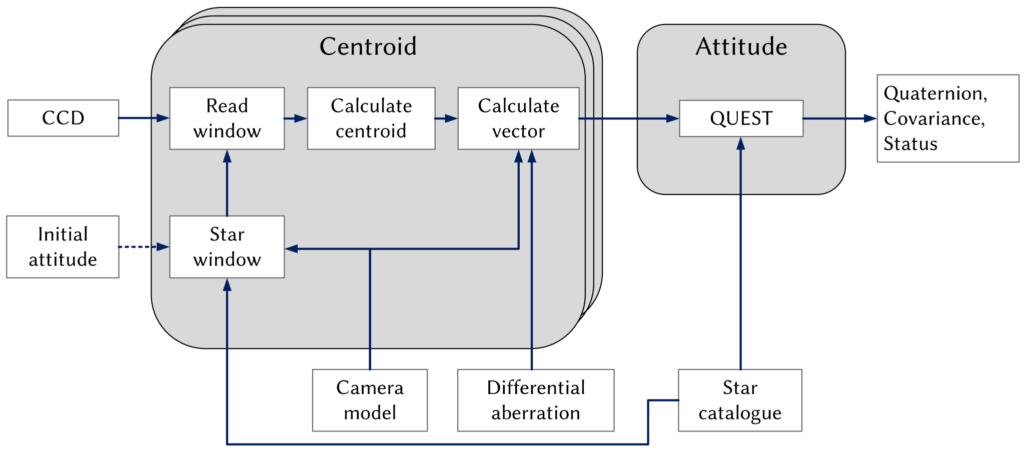

Figure 1: Overview of the FGS algorithm Grießbach (2018).

tion, the algorithm separately estimates each centroid using a Gaussian point spread function (PSF) observation model. The PSF describes the distribution of the star light over the CCD pixels. Equation (1) shows the Gaussian PSF observation model $h(i, j)$ of a single pixel (Grießbach, 2018).

$$h = \frac{I_m}{2\pi\sigma^2} \int_{i}^{i+1} e^{-\frac{(u-u_c)^2}{2\sigma^2}} \mathrm{d}u \int_{j}^{j+1} e^{-\frac{(v-v_c)^2}{2\sigma^2}} \mathrm{d}v + D + \xi \quad (1)$$

The FGS algorithm uses the measured pixel intensities to determine the centroid position $(u_c, v_c)^T$, intensity I_m, image background D and PSF width σ. A non-linear least square fitting method iteratively refines the parameters of the PSF model. The FGS algorithm calculates the correction by means of the QR-decomposition (Grießbach, 2018). In the next step, the pixel coordinates of the calculated centroid position are transformed into star direction vectors in the camera boresight reference frame. The x- and y-axis of the detector and the optical axis of the camera describe the boresight reference frame.

In the last step, the FGS algorithm calculates an attitude from at least two star directions in the boresight reference frame and the corresponding reference vectors from a star catalog. In addition, it delivers an attitude covariance matrix.

3 RELATED WORK

Equivalence class partition testing “is probably the most widely described, and one of the most widely practiced, software testing techniques” (Kaner, 2004). In this section, we present a selection of published work on equivalence class partition testing.

3.1 Equivalence Class Partition Testing

Various studies investigated equivalence class partition testing strategies for different domains: railway, automotive, avionics, etc. (Huang and Peleska, 2016).

In the automotive domain, DaimlerChrysler Research developed a test approach, called Time Partition Testing (TPT), to test the continuous behavior of control systems. Bringmann and Krämer (2006) explained the principle of the TPT approach using an exterior headlight controller as an example. In most cases, automotive embedded control systems are based on complex functional behavior and a large input domain. To increase the test efficiency the TPT approach systematically selects test cases revealing redundant or missing test scenarios. Using a graphical state machine notation, the TPT approach partitions a test scenario into stream-processing components. Each component defines the behavior of output variables depending on the behavior of input variables up to a certain point in time, specified by a temporal predicate. Test cases define variations in the state machine to test various functional aspects of the system under test.

The study shows that state machines are suitable to partition the temporal behavior of input and output variables in order to model, compare and select test cases. The modeled test cases test the complex functional requirements of control systems. A huge input domain and complex functional behavior are also characteristics of the system class we investigate in this paper. However, the behavior of systems from this class is not dependent on the arrival time of the input values. Hence, the TPT approach is not applicable to the system class that we consider.

In (Huang and Peleska, 2016), the authors presented a model-based black-box equivalence class partition testing strategy used in the railway domain. The approach automatically generates finite and complete test suites for safety-critical reactive systems in relation to fault models. Huang and Peleska investigated the approach using the Ceiling Speed Monitor of the European Train Control System as an example for systems with potentially infinite input domain but finite output domain and internal variables. Their approach models the reactive behavior of such systems by means of deterministic state transition systems. Moreover, it partitions the state space into a finite number of equivalence classes such that all states in a class provide the same output traces for the same non-empty input trace. Based on these classes, they generates a complete test suite in the following sense: First, at least one test in this suite fails when testing an application that violates a given specification. Second, each test in the suite passes for all applications that satisfy the specification. Huang and Peleska investigated models whose behavior can be represented by state transition systems. However, we have no state transition system description of our considered satellite application. Hence, we present an approach that does not need such a description.

4 MULTIDIMENSIONAL COVERAGE CRITERIA FOR AUTOMATED TESTING

Many image processing applications require various input parameters such as position of an object in the image, magnitude of an object, position of an object in a pixel, a pattern to distinguish different objects, etc. This leads to a huge input domain, which makes testing expensive. Testing such applications manually is error-prone and time-consuming. Therefore, automated test systems are needed. However, automated test generation for satellite on-board image processing applications poses a challenge: The number of possible input parameter combinations in the test image are very large. That means an enormous amount of test cases is possible. This makes it hard to efficiently capture the huge input domain.

To overcome that problem, we define a partitioning approach that systematically selects test cases from the huge input domain of satellite on-board image processing applications. Our approach can be used to assess and to enhance a given test suite. To evaluate the efficiency of our test approach, we investigate a case study, namely the PLATO FGS algorithm as described in Section 2.2. Satellite on-board image processing algorithms especially require extensive testing because such algorithms are subject to strict requirements with respect to reliability and mathematical accuracy.

Figure 2 depicts an overview of our proposed partitioning approach. The key idea is to define equivalence classes on input parameters that are typically used by satellite on-board image processing applications, namely position, magnitude, sub-pixel position, and distribution model. Our main contributions are threefold: First, we present novel concepts to partition the individual input parameters of satellite image processing algorithms into equivalence classes. Second, we define multidimensional coverage criteria based on a combination of the individual criteria for each input parameter to efficiently cover the whole input domain. Third, we define a test generation algorithm that automatically selects test cases that completely cover the whole input domain according to our multidimensional coverage criteria.

The goal of the test is to automatically detect errors in the on-board image processing application code. To achieve this, our test generation algorithm selects for each equivalence class combination a test case from a given test suite as representatives. Thus, our approach reduces the number of redundant test cases. Furthermore, our algorithm generates new test cases for missing combinations to reach complete coverage of the input domain. The result is a reasonably small test suite that covers the whole input domain of the image processing application. The selected test cases serve as input for our automated testing framework. Moreover, we insert requirements for the automated evaluation of the image processing application results. If they do not meet the requirements, the test detects an error.

The following sections describe the mentioned steps of the partitioning approach in more detail using the PLATO FGS algorithm as a case study.

4.1 Assumptions and Limitations

In the following, we consider systems with objects in an image as inputs. In the case study, the observed objects are stars with magnitudes between 5.5 to 7.0, uniformly distributed in the image (Grießbach, 2018).

We consider four parameters that affect the mathematical accuracy of the FGS algorithm: the guide star position, magnitude, sub-pixel position, and PSF shape. The test evaluation is based on the centroid position calculated by the centroid algorithm of the FGS. The input of the centroid calculation is a single star. We define a test star as a test case for the automated test generation.

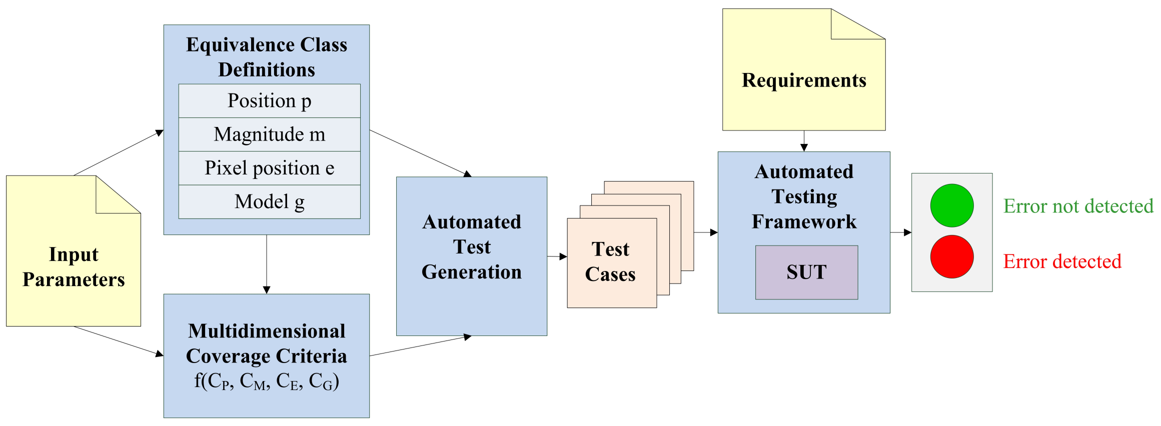

Figure 2: Overview of the partitioning approach.

4.2 Running Example: Input Parameters

One of the main contributions of this paper is our definition of multidimensional coverage criteria on the input domain of satellite on-board image processing applications. These criteria are based on the combination of equivalence classes of the FGS input star parameters given in Section 4.1. The mathematical accuracy of the centroid estimation depends on the combination of these input parameters. In this section, we describe how the parameters affect the quality of the centroid calculation.

The star signal is spread over all pixels in the sub-image. Hence, each pixel includes information about the star. However, 90 % of the energy is within 2×2 pixel around the centroid. Each pixel also contains noise. We call the information usable if the star signal is little interfered by noise and the Signal-to-Noise Ratio (SNR) is high. If the SNR in the pixel is sufficient, a linear independent equation exists for this pixel. The centroid calculation needs at least 5 linear independent equations to estimate the 5 unknown parameters of the pixel observation (cf. Equation (1)).

The distribution of the star signal depends on the star position on the Focal Plane Assembly (FPA) and the sub-pixel position. Due to optical aberrations of the telescope, the PSF shape of the star is wider in the FPA corner than close to the center. Assume the other input parameters contain reasonably good, constant values. Then a small PSF leads to a low number of pixels with a high SNR and a low number of linear independent equations. In case of a wide PSF the SNR is low but many linear independent equations exist. Both cases can be sufficient for an accurate parameter estimation (Grießbach, 2018).

The SNR in a pixel also depends on the centroid sub-pixel position. Suppose the other parameters have adequate, constant values. If the centroid is positioned in the pixel center, most star flux is accumulated in a few pixels. Then these pixels have a high SNR compared to the neighboring pixel. In contrast, more pixels have a sufficient SNR if the centroid is on the pixel border or corner. In this case, the star information is distributed more evenly over several pixels. The other pixels have a low SNR. Due to movement, the centroid may move to neighbor pixels. This leads to variations in the pixel illumination and the apparent centroid position (Grießbach, 2018).

The star magnitude affects the measured flux (photoelectrons per second) of the star. The accumulated number of photoelectrons per pixel denotes the illumination of a pixel. Equation (2) shows the relation between the magnitude m and the corresponding flux F_m in e^-/s.

$$F_m = F_0 T Q A * 10^{-0.4*m} \tag{2}$$

with magnitude m, reference flux F_0 of a star with $m = 0$, transmission efficiency T of the optical system, quantum efficiency Q of the detector, and effective light-collecting area A. As the equation shows, the star flux is non-linear to the magnitude of the star. A low magnitude corresponds to a high number of photoelectrons, corresponding to a higher SNR per pixel. More information is usable than in case of a high magnitude.

In addition, the accuracy of the centroid calculation depends on the PSF shape. In the best case scenario, the shape is a symmetric Gaussian-PSF. Then, the observation model perfectly fits the star. Therefore, the accuracy of the centroid calculation is high. But in reality, the PSF shape is non-Gaussian. In that case, the observation model is less accurate. Besides, movements lead to stronger variations in the expected centroid positions (Grießbach, 2018).

While individual parameter values might provide a good centroid estimation, a combination of param-

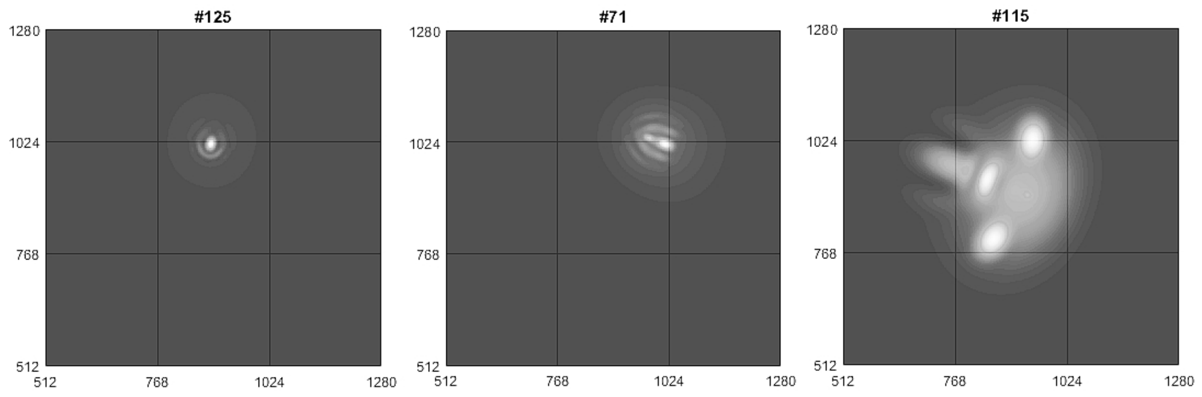

Figure 3: Examples of different low quality stars (Grießbach, 2018).

eters may change the quality of the results. For illustration, Figure 3 shows some example stars with non-Gaussian-PSF shape that are less suitable as guide stars. They all lead to inaccurate estimation results.

What the three stars have in common is that their intensity is concentrated on a pixel edge. For all stars, the magnitude and FPA position are sufficiently good. However, a small variation due to movement leads to big changes of the illumination. Since the Gaussian-PSF observation model does not fit the star PSF shape perfectly, the centroid estimation is less accurate and the FGS assumes a big movement of the star.

4.3 Equivalence Class Definitions

The quality of the centroid calculation of the FGS algorithm depends on various parameters. Thus, we define the input domain as a set of input parameters I. The set includes the position on the FPA $\mathcal{P}$, the magnitude $\mathcal{M}$, the sub-pixel position $\mathcal{E}$ and the PSF shape $\mathcal{G}$. To keep the approach more flexible the tester sets the borders of equivalence classes. Thus parameters can also be excluded from the analysis. In this section, we present our concepts for partitioning the input parameters $\mathcal{P}$, $\mathcal{M}$, $\mathcal{E}$ and $\mathcal{G}$ into equivalence classes.

Star Position on the FPA. Section 4.2 clarifies that size and shape of the PSF depend on the star position on the FPA. Since the PSF changes with the distance to the FPA center, our idea is to partition the FPA into equally sized, circular areas as seen in Figure 4. The tester specifies the initial radius r_0. The rectangles represent the image area of the fast camera CCDs.

We partition parameter $\mathcal{P}$ into equivalence classes P_{r_n}. Each class P_{r_n} corresponds to a circular FPA area with inner radius r_{n-1} and outer radius r_n.

$$\mathcal{P} = P_{r_1} \cup P_{r_2} \cup \ldots \cup P_{r_n} \cup \ldots \cup P_{r_{I_\mathcal{P}}} \text{ with } 1 \leq n \leq I_\mathcal{P} \quad (3)$$

Let S denote the set of available stars. A star $s \in S$ lies in an equivalence class P_{r_n} if following condition

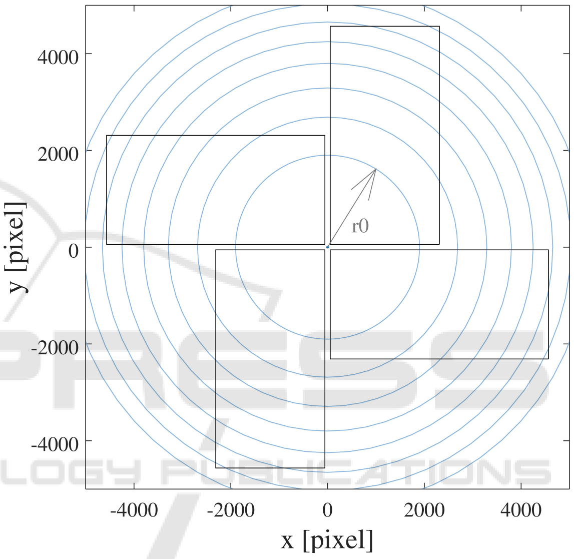

Figure 4: Example of cyclic borders of FPA equivalence classes.

holds:

$$r_{n-1} \leq p(s) < r_n, \text{ with } p(s) = \sqrt{{x_s}^2 + {y_s}^2} \quad (4)$$

where (x_s, y_s) is the position of star s on the FPA and $p(s)$ is the distance of the star s to the FPA center.

Star Magnitude. A useful partitioning of magnitude values into equivalence classes is not obvious. Our idea is to partition the star flux, which is non-linear to the magnitude, range into $I_\mathcal{M} \in \mathbb{N}$ equidistant parts that represent the equivalence classes. We define Equation (5) to obtain the upper limit of a sub-range.

$$F_{m_j} = F_{7.0} + j\frac{F_{5.5} - F_{7.0}}{I_\mathcal{M}} \quad (5)$$

F_{m_j} is the flux of magnitude m_j and $j = 1 \ldots I_\mathcal{M}$ represents the j-th equivalence class of parameter $\mathcal{M}$. $F_{5.5}$ and $F_{7.0}$ are the numbers of photons for magnitude 5.5

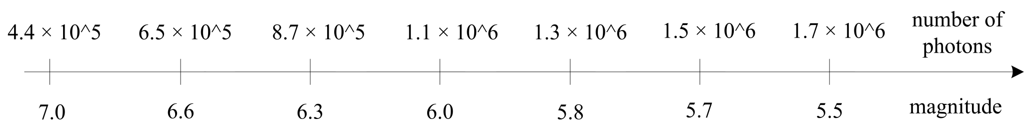

Figure 5: Example partitioning of magnitude range.

and 7.0. We calculate the flux values $F_{5.5}$ and $F_{7.0}$ by using Equation (2). Furthermore, we use Equation (6) to recalculate the magnitude m_j from the calculated flux limit F_{m_j} of the flux sub-range.

$$m = -2.5 * log\left(\frac{F_m}{F_0 TQA}\right) \tag{6}$$

From a formal point of view, we partition the parameter $\mathcal{M}$ into equivalence classes M_l.

$$\mathcal{M} = M_{7.0} \cup \ldots \cup M_{l_j} \cup \ldots \cup M_{5.5} \tag{7}$$

with $l_j \in \mathbb{R}$ and $5.5 \leq l_j \leq 7.0$. Each equivalence class M_{l_j} is a magnitude sub-range with upper limit l_j. Each available star s lies in equivalence M_{l_j} if it satisfies the condition in Equation (8).

$$l_{j-1} \leq m(s) < l_j \tag{8}$$

where $m(s)$ denotes the observed magnitude of star s and l_j with $j = 1 \ldots I_{\mathcal{M}}$ is the upper limit of the j-th magnitude sub-range. The tester specifies the number of equivalence classes $I_{\mathcal{M}} \in \mathbb{N}$ of the parameter $\mathcal{M}$. Figure 5 illustrates an example partitioning of the magnitude range.

Star Sub-pixel Position. The quality of the centroid estimation of stars close to a pixel border is as sensitive to movements as the estimation of stars with centroids on a pixel corner and vice versa. For this reason, we divide the pixel area into different sub-areas as shown in Figure 6.

The tester specifies the ratio r of the central area of the pixel to the pixel area, for example, 1/2, 3/5, etc. If a is the pixel size, then the length of the edge of the central area results from Equation (9).

$$b = a\sqrt{r} \tag{9}$$

With that, we obtain the lower left corner l and the upper right corner u of the central pixel area, with

$$l = \left(\frac{a}{2} - \frac{b}{2}, \frac{a}{2} - \frac{b}{2}\right) \text{ and } u = \left(\frac{a}{2} + \frac{b}{2}, \frac{a}{2} + \frac{b}{2}\right) \tag{10}$$

Based on these corners, we partition parameter $\mathcal{E}$ into equivalence classes E_i with $i = 0 \ldots 8$. The equivalence class E_i is the i-th pixel sub-area. A star s lies in an equivalence class if it satisfies the corresponding condition.

$$\mathcal{E} = E_0 \cup E_1 \cup \ldots \cup E_8 \tag{11}$$

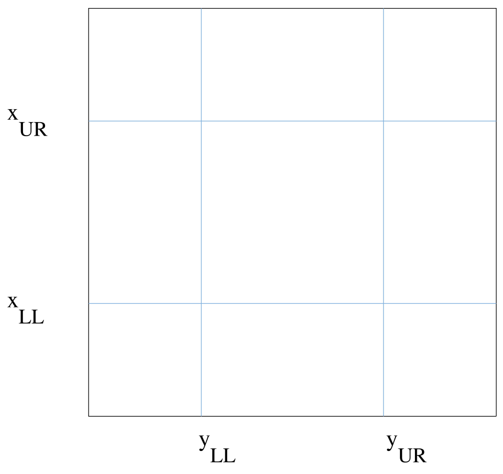

Figure 6: Example borders of pixel equivalence classes.

$$\begin{array}{ll}
E_0: & 0 \leq e_x(s) < x_l \wedge 0 \leq e_y(s) < y_l \\
E_1: & 0 \leq e_x(s) < x_l \wedge y_l \leq e_y(s) < y_u \\
E_2: & 0 \leq e_x(s) < x_l \wedge y_u \leq e_y(s) < a \\
E_3: & x_l \leq e_x(s) < x_u \wedge 0 \leq e_y(s) < y_l \\
E_4: & x_l \leq e_x(s) < x_u \wedge y_l \leq e_y(s) < y_u \\
E_5: & x_l \leq e_x(s) < x_u \wedge y_u \leq e_y(s) < a \\
E_6: & x_u \leq e_x(s) < a \wedge 0 \leq e_y(s) < y_l \\
E_7: & x_u \leq e_x(s) < a \wedge y_l \leq e_y(s) < y_u \\
E_8: & x_u \leq e_x(s) < a \wedge y_u \leq e_y(s) < a
\end{array} \tag{12}$$

$e_x(s)$ and $e_y(s)$ return the x-coordinate and y-coordinate of s in the pixel respectively.

Star PSF Shape. We partition the parameters $\mathcal{G}$ in two equivalence classes G_G and G_{NG} since two PSF shapes are distinctive (see Section 4.2). If a star has a Gaussian-PSF shape it is in class G_G otherwise it is in class G_{NG}.

4.4 Multidimensional Coverage Criteria

We define multidimensional coverage criteria on the input domain of satellite on-board image processing applications to measure the coverage of a test suite with respect to input parameter combinations. If the measured coverage of a test suite is not complete, our automated test generation algorithm automatically inserts test cases for missing combinations. This section presents our definitions of multidimensional coverage criteria on the input domain $I = \{\mathcal{P}, \mathcal{M}, \mathcal{E}, \mathcal{G}\}$.

The individual coverage of an input parameter denotes the ratio of equivalence classes that are covered by at least one test case from a given test suite to the number of equivalence classes of this input parameter. Equations (13) to (16) show this definition for the input parameters $\mathcal{P}, \mathcal{M}$, $\mathcal{E}$ and $\mathcal{G}$.

$$C_{\mathcal{P}} = \frac{\#\,covered\ elements\ of\ \mathcal{P}}{|\mathcal{P}|} \tag{13}$$

$$C_{\mathcal{M}} = \frac{\#\,covered\ elements\ of\ \mathcal{M}}{|\mathcal{M}|} \tag{14}$$

$$C_{\mathcal{E}} = \frac{\#\,covered\ elements\ of\ \mathcal{E}}{|\mathcal{E}|} \tag{15}$$

$$C_{\mathcal{G}} = \frac{\#\,covered\ elements\ of\ \mathcal{G}}{|\mathcal{G}|} \tag{16}$$

The coverage domain for our multidimensional coverage criteria is the Cartesian product of equivalence classes of the input parameters $\mathcal{P}$, $\mathcal{M}$, $\mathcal{E}$ and $\mathcal{G}$. Therefore, an input combination is a tuple of equivalence classes (P_i, M_j, E_k, G_l), where $P_i \in \mathcal{P}$, $M_j \in \mathcal{M}$, $E_k \in \mathcal{E}$ and $G_l \in \mathcal{G}$. Furthermore, a test case is a star represented by a tuple of parameter values $(p, m, e, g) \in (P_i, M_j, E_k, G_l)$. The following example test cases clarify these definitions.

Example 1:

$(930.4, 6.5, (0.4, 0.1), G) \in (P_{2000} \times M_{6.6} \times E_3 \times G_G)$

The test star position is in the FPA area with radius 2000. The star belongs to equivalence class $M_{6.6}$ because its magnitude value is between 6.3 and 6.6. The star center is located in the middle-left pixel subarea. That corresponds to equivalence class E_3. The star is part of equivalence class G_G, because it has a Gaussian-PSF shape.

Example 2:

$(579.1, 6.5, (0.9, 0.2), G) \in (P_{2000} \times M_{6.6} \times E_6 \times G_G)$

The test star is similar to the star in the first example, but it belongs to equivalence class E_6, which means that the center of the star is positioned nearby the upper left pixel border.

Our multidimensional coverage criterion is fully satisfied if the test cases in a test suite cover all possible input combinations at least once. The number of required covered input combinations for a complete coverage is $|\mathcal{P} \times \mathcal{M} \times \mathcal{E} \times \mathcal{G}|$. In the remaining sections, we denote a test suite that completely covers the input domain with respect to our multidimensional coverage criteria as a complete test suite. The multidimensional coverage C results from the ratio of the amount of input combinations covered by at least one test case to the total number of input combinations.

$$C = \frac{\#\,covered\ input\ combinations}{|\mathcal{P} \times \mathcal{M} \times \mathcal{E} \times \mathcal{G}|} \tag{17}$$

Our test approach calculates the individual and multidimensional coverage of a given test suite using Algorithm 1. The input parameters $\mathcal{P}$, $\mathcal{M}$, $\mathcal{E}$, and $\mathcal{G}$ contain $I_{\mathcal{P}}, I_{\mathcal{M}}, I_{\mathcal{E}}, I_{\mathcal{G}}$ equivalence classes respectively.

```
Input: Test suite $TS$
Output: Multidimensional coverage $Cov$ of $TS$
1  $C_{\mathcal{P}} = C_{\mathcal{M}} = C_{\mathcal{E}} = C_{\mathcal{G}} = C = \emptyset$;
2  foreach tc with (p, m, e, g) $\in$ TS do
3      $i_P$ = getPosECId(p);
4      $C_{\mathcal{P}} \leftarrow C_{\mathcal{P}} \cup i_P$;
5      $i_M$ = getMagECId(m);
6      $C_{\mathcal{M}} \leftarrow C_{\mathcal{M}} \cup i_M$;
7      $i_E$ = getPixECId(e);
8      $C_{\mathcal{E}} \leftarrow C_{\mathcal{E}} \cup i_E$;
9      $i_G$ = getModECId(g);
10     $C_{\mathcal{G}} \leftarrow C_{\mathcal{G}} \cup i_G$;
11     $C \leftarrow C \cup (i_P, i_M, i_E, i_G)$;
12 end
13 $Cov_{\mathcal{G}} = |C_{\mathcal{P}}|/I_{\mathcal{P}}$;
14 $Cov_{\mathcal{M}} = |C_{\mathcal{M}}|/I_{\mathcal{M}}$;
15 $Cov_{\mathcal{E}} = |C_{\mathcal{E}}|/I_{\mathcal{E}}$;
16 $Cov_{\mathcal{G}} = |C_{\mathcal{G}}|/I_{\mathcal{G}}$;
17 $Cov = |C|/(I_{\mathcal{P}} \cdot I_{\mathcal{M}} \cdot I_{\mathcal{E}} \cdot I_{\mathcal{G}})$
```

Algorithm 1: Coverage calculation.

For each test case in the given test suite, the algorithm computes for each input parameter the index i_P, i_M, i_E, i_G of the corresponding equivalence class from $\mathcal{P}$, $\mathcal{M}$, $\mathcal{E}$ and $\mathcal{G}$. The algorithm adds the indices to the sets $C_{\mathcal{P}}$, $C_{\mathcal{M}}$, $C_{\mathcal{E}}$ and $C_{\mathcal{G}}$ respectively. Moreover, it inserts the tuple (i_P, i_M, i_E, i_G) into the set C that contains all covered input combinations. As the algorithm uses the union operator to add the tuples to the set, each tuple is included in the set only once. The algorithm applies Equations (13) to (17) to compute the individual and multidimensional coverage.

Our partitioning approach can be used to assess the quality of test suites with respect to their coverage on the input space of a satellite on-board image processing application with respect to individual and multidimensional coverage criteria.

4.5 Automated Test Generation

In order to systematically generate a test suite that completely covers the input domain according to our multidimensional coverage criteria, we propose Algorithm 2. The complete test generation algorithm uses 1 to assess a given test suite and then systematically generates missing test cases.

```
Input: Input combination universe $U$, covered
       input combination set $C$, test suite $TS$
Output: Complete test suite $TS$
Cov = computeMultidimCoverage(TS);
if $Cov < 1$ then
    $W \leftarrow U \setminus C$;
    foreach $w \in W$ do
        tc = generateTC(w);
        $TS \leftarrow TS \cup tc$;
    end
end
```

Algorithm 2: Generate complete test suite.

The algorithm generates set W that contains all input combinations not covered by the given test suite. For each input combination in W, the algorithm uses procedure `generateTC` that generates a test case by randomly selecting values from the equivalence classes of the missing combinations. The algorithm adds the newly generated test case to the test suite. In this way, it efficiently inserts missing but relevant test cases into the test suite. This increases the multidimensional coverage and therefore the error detection capability of the given test suite. The result of our automated test generation is a complete test suite.

If the set of covered input combinations C is empty, then the set of uncovered input combinations W is equal to the universe of possible input combinations U. Hence, Algorithm 2 can also be used to generate a new test suite that completely satisfies the multidimensional coverage criteria. Our automated testing framework only selects one test case per input combination. This efficiently reduces the number of redundant test cases for the test execution.

5 EVALUATION

To evaluate our approach, we have implemented the proposed partitioning and test generation algorithm. We have evaluated its applicability and error detection capability for satellite on-board image processing applications using the FGS algorithm of the PLATO mission with various test suites.

5.1 Implementation

Figure 7 shows the block diagram of our test setup. As the figure depicts, our test environment runs on a Windows system. We have implemented the environment in C++ based on the model-view-controller pattern. We allow the tester to specify input parameters with or without graphical user interface (GUI).

Our automated test generation algorithm returns a star catalog to simulate star data for missing input combinations. The catalog is a text file that includes right ascension, declination, and magnitude of stars that should be simulated. We manually insert the catalog into the PLATO simulator PlatoSim (Marcos-Arenal et al., 2014). PlatoSim writes the simulated star data to a HDF5 file (The HDF Group, 2018). Each file contains an image sequence of several time steps of a star in a hierarchical file format. Since PlatoSim is not developed for Windows systems, the simulator runs in a Linux virtual machine.

As shown in Figure 7, we connect the Windows system via a SpaceWire USB brick to a GR-XC6S FPGA development board (PENDER ELECTRONIC DESIGN GmbH, 2011) running at 50 MHz. SpaceWire is a data-handling network for spacecraft defined in (ECSS Executive Secretariat, 2008). For that, our test environment uses the C SpaceWire USB API Library for the SpaceWire USB brick. A prototype of the FGS algorithm, written in C, runs on the evaluation board. We load the software with the Leon debug monitor GRMON onto this board. Via a UART interface we receive debug information in a terminal. For example, stack size, hardware information, etc.

Our objective is to evaluate our approach for the development and test of the FGS algorithm implementation. Moreover, our goal is to test execution time and mathematical accuracy of the FGS algorithm under realistic conditions. For example, a calculation on the development board is slower than the same calculation on a Windows system. Therefore, we run the application under test on the target hardware and keep the test system in the software development cycle.

5.2 Experimental Results

In this section, we present the experimental results for generating a test suite using our partitioning approach for testing the PLATO FGS algorithm.

Since no investigations of the class boundaries have yet been carried out, we have started by estimating the following parameters for the experiments:

- Initial radius r_0 of FPA partitioning: 1900 pixel
- Number of magnitude sub-ranges: 6
- Ratio r of central sub-area to pixel area: 0.2

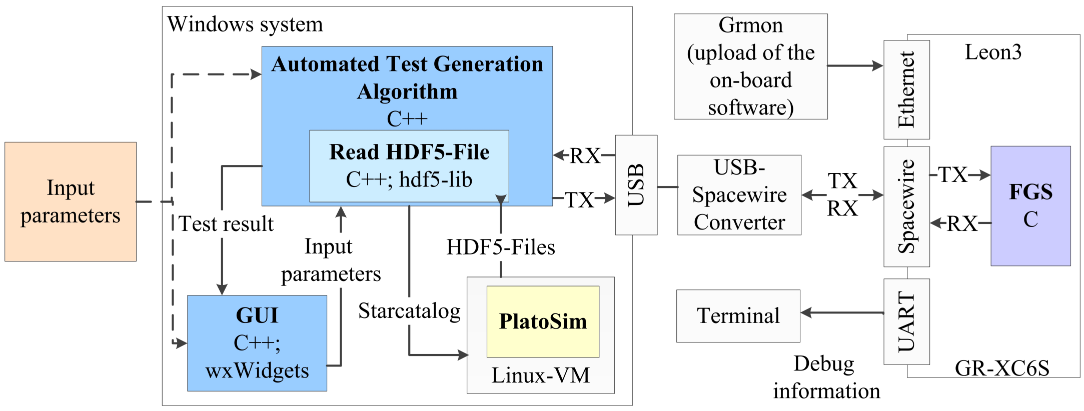

Figure 7: Block diagram of test setup.

Using the parameters leads to 8 equivalence classes of input parameter $\mathcal{P}$, 6 equivalence classes of parameter $\mathcal{M}$ and 9 equivalence classes of $\mathcal{E}$. Input parameter $\mathcal{G}$ consists of two equivalence classes (G_G and G_{NG}). Thus, our automated testing framework needs 864 test cases to completely cover the whole input domain of the FGS algorithm.

To evaluate the approach, we have investigated the quality of two different test suites with respect to their multidimensional coverage on the input domain. One test suite contains 82 randomly generated stars. Our automated test generation algorithm identifies 10 stars in the suite as redundant. The test suite achieves 8.3 % coverage of the input domain with respect to our multidimensional coverage criteria. Algorithm 2 from Section 4.5 enhances the suite to achieve complete coverage on the input domain. In the following, we call it complete test suite. The other test suite contains 902 randomly generated but evenly distributed stars. We call it random test suite and did not improve it. Table 1 shows the coverage of the test suites for each input parameter as well as the achieved multidimensional coverage.

Table 1: Coverage values of the test suites.

	random	complete
Test stars	902	874
Covered input combinations	112	864
$C_{\mathcal{P}}$ [%]	87.5	100.0
$C_{\mathcal{M}}$ [%]	16.7	100.0
$C_{\mathcal{E}}$ [%]	100.0	100.0
$C_{\mathcal{G}}$ [%]	100.0	100.0
Multidim. coverage [%]	13.0	100.0

Table 1 shows that the utilization of the equivalence class partitioning method reduces the random test suite by hundreds of redundant test cases. Since there are no unnecessary executions of redundant test cases, this saves test time. Thus, the method increases the efficiency of the test process. The random test suite achieves a high individual coverage of three input parameters. However, due to the low individual coverage of input parameter $\mathcal{M}$, the multidimensional coverage of the test suite is low. Furthermore, Table 1 exhibits that the complete test suite covers the whole input domain of the FGS algorithm.

To assess the partitioning approach, we have automatically inserted some faults into the code of the centroid calculation of the PLATO FGS algorithm. These injected faults belong to three classes: missing assignment, wrong assignment and wrong condition. For each test execution, we have injected a single error at a different position in the code. Our objective is to check if the complete test suite achieves a higher error detection capability than the random test suite.

In each experiment, our test application sent 1000 packets per selected test star, with one exposure each, to the evaluation board running the FGS algorithm. Our test application averages the resulting centroid positions over all exposures and compares it with the position in the star catalog. If the deviation of the positions is greater than a predefined value, the test detects the error. Table 2 shows the output for one test case that detects a missing assignment error. The high deviation of the calculated position from the given position reveals an error in the centroid calculation.

During the experiments, we have injected three missing assignment errors, three wrong assignment errors, and three wrong condition errors. Table 3 summarizes the experimental results for both test suites.

The table shows long execution times for the tests. The reason is that the evaluation board only receives 20 packets per second. Moreover, Table 3 shows that both test suites do not reveal all injected errors with respect to the given test criterion. The random test suite, as well as the complete test suite, detects one missing assignment error. In addition, 6 test cases from the complete test suite reveal one wrong assign-

Table 2: Output for a sample test case.

i_G	i_P	i_M	i_E	Star-Id	x [pixel]	y [pixel]	deviation [pixel]	result
1	1	0	8	28	2.96×10^{19}	2.38×10^{19}	3.80×10^{19}	error detected

Table 3: Test suites evaluation results.

	random	complete
Test cases	112	864
Execution time [h]	1.5	11.8
Detected errors	1	3
Undetected errors	9	7
Error detection ability [%]	10.0	30.0

ment error and 1 test case detects one wrong condition error. During the test development, we detected an error in the FGS algorithm code that we did not inject. But using the specified test criterion, both test suites do not detect the error. Moreover, the tests do not detect three of the injected assignment errors because the deviation only changes slightly in case of these errors. In addition, for some test cases, the deviation increases and in others decreases in such a way that the specified deviation value is not reached. The tests also do not detect the other injected errors and the unintended error because these errors affect other parameters estimated by the centroid calculation. Therefore, the deviation for all test cases is smaller than the predefined value in the test criterion. However, the error detection capability of the complete test suite is with respect to the specified test criterion three times higher than the error detection capability of the random test suite. Nevertheless, for both test suites, the error detection capability is low because not all injected errors affect the results in such a way that the deviation increases. Therefore, our specified test criterion is not suitable to detect all injected errors. This shows, that the specified test criterion plays an important role in the success of the tests.

We have specified another test criterion: the test passes if the distance between the centroid position calculated by an erroneous calculation and the centroid position resulting from an assumed error-free calculation exceeds a predefined value. In this case, not all test cases of the test suites detect the errors. This means special input combinations are more capable to detect errors than others. In case of the wrong condition errors and the unintended error, the percentage of error detecting test cases from the random test suite is up to one third compared to the percentage of error detecting test cases of the complete test suite. Therefore, using the complete test suite increases the confidence to find an error with the test. In the other cases, the percentage of error detecting test cases is approximately 99 % for both test suites. However, for this test criterion, the complete test suite has a higher error detection capability than the random test suite.

Our partitioning approach reduces the number of relevant test cases. Therefore, applying the approach increases the test efficiency. The results show that the error detection capability of the test suite that completely satisfies our multidimensional coverage criteria is significantly higher than the capability of the random test suite. The success of our approach depends on the specified test criterion as well as on the definition of the equivalence classes.

6 CONCLUSION

Due to the large input domain of on-board image processing applications, an enormous amount of test cases is possible. This makes it infeasible to capture the whole input domain and execute the test cases exhaustively. In this paper, we have presented a test partitioning approach that systematically generates a reasonably small test suite with complete coverage on the input domain of satellite on-board image processing applications. To achieve this, we have defined a dedicated partitioning for each input parameter of a satellite on-board image processing application, and we have defined coverage criteria with respect to the proposed equivalence classes. Based on these individual coverage criteria, we have furthermore defined multidimensional coverage criteria, which can be used to assess a given test suite with respect to its coverage on the complete input domain. Finally, we have presented an automated test generation algorithm that systematically generates missing test cases according to our multidimensional coverage criteria. As a result, our approach is able to fully automatically generate test suites that are complete with respect to our multidimensional coverage criteria. The tester specifies the size of our equivalence classes. This makes our approach adjustable to available test times and also to other image processing applications.

We have investigated the effectiveness of our proposed test approach on the FGS algorithm as an application with high criticality for the PLATO mission. In the experiments, our automated test generation algorithm generates a test suite that is complete with respect to our multidimensional coverage criteria. To demonstrate the effectiveness of our test approach, we have compared the error detection capability of a randomly generated test suite and the generated complete test suite. The use of our equivalence classes of the input parameters reduces the number of redundant test cases in the randomly generated test suite by 87.6 %.

During the experiments, we have successively injected 9 errors in the FGS algorithm code to investigate the error detection capability of both test suites. We have used two different test criteria: First, a test case detects an error if the distance between the calculated centroid position and a given position is larger than a predefined value. Second, a test case detects an error if the distance of the erroneous calculated position and an assumed error-free calculated position exceeds a specified value. We have observed that different test criteria lead to different test results. For the first test criterion, the complete test suite detects 3 injected errors while the randomly generated test suite detects 1 injected error. The error detection capability of the complete test suite is about 3 times higher than the capability of the randomly generated test suite. But both test suites do not detect all errors. In another experiment, we have used the second test criterion. In this case, not all test cases in the test suites detect all errors. In the case of 3 injected errors and an unintended error, the percentage of error detecting test cases in the complete test suite is again about 3 times higher than for the randomly generated test suite. For the other 6 injected errors, the percentage of error detecting test cases is for both test suites about 99 %.

The experiments showed that a systematic test using our proposed partitioning approach increases the error detection capability of a given test suite. This makes the partitioning approach efficient and effective. In addition, it facilitates the automated generation, the execution, and the evaluation of test cases.

So far, we have injected errors in the application code. But in space, many missions suffer from cosmic radiation that flips bits in binary code or cause hot pixels in input images. We plan to investigate the efficiency of our approach by injecting errors in input data or in the binary code of the application in future work. Finally, we have evaluated our approach with a single application. Later on, we plan to investigate the flexibility of our approach for other applications, for example, blob feature extraction in the robotics domain (Bruce et al., 2000; Merino et al., 2006).

REFERENCES

Bhat, A. and Quadri, S. (2015). Equivalence class partitioning and boundary value analysis-a review. In *Intl. Conf. on Computing for Sustainable Global Development (INDIACom)*, pages 1557–1562. IEEE.

Bringmann, E. and Krämer, A. (2006). Systematic testing of the continuous behavior of automotive systems. In *International Workshop on Software Engineering for Automotive Systems*, pages 13–20. ACM.

Bruce, J., Balch, T., and Veloso, M. (2000). Fast and inexpensive color image segmentation for interactive robots. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, volume 3, pages 2061–2066. IEEE.

DLR (2017). Grünes Licht für europäisches Weltraumteleskop PLATO. http://www.dlr.de/dlr/desktopdefault.aspx/tabid-10081/151_read-22858/#/gallery/27241.

ECSS Executive Secretariat (2008). Space engineering. SpaceWire – Links, nodes, routers and networks.

ESA (2012). ESA's 'Cosmic Vision'. http://www.esa.int/Our_Activities/Space_Science/ESA_s_Cosmic_Vision.

Grießbach, D. (2018). Fine Guidance System Performance Report. DLR, Berlin.

Huang, W.-l. and Peleska, J. (2016). Complete model-based equivalence class testing. *Intl. Journal on Software Tools for Technology Transfer*, 18(3):265–283.

Kaner, C. (2004). Teaching domain testing: A status report. In *Conference on Software Engineering Education and Training*, pages 112–117. IEEE.

Marcos-Arenal, P., Zima, W., De Ridder, J., Aerts, C., Huygen, R., Samadi, R., Green, J., Piotto, G., Salmon, S., Catala, C., et al. (2014). The PLATO Simulator: modelling of high-precision high-cadence space-based imaging. *Astronomy & Astrophysics*, 566:A92.

Merino, L., Wiklund, J., Caballero, F., Moe, A., De Dios, J. R. M., Forssen, P.-E., Nordberg, K., and Ollero, A. (2006). Vision-based multi-UAV position estimation. *IEEE robotics & automation magazine*, 13(3):53–62.

PENDER ELECTRONIC DESIGN GmbH (2011). Gr-xc6s-product_sheet.

Peter Liggesmeyer (2009). *Software-Qualität: Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag, 2 edition.

The HDF Group (April 05, 2018). Hdf5. https://portal.hdfgroup.org/display/HDF5/HDF5.

Varshney, S. and Mehrotra, M. (2014). Automated software test data generation for data flow dependencies using genetic algorithm. *International Journal*, 4(2).

Witteck, U. (2018). Automated test generation for satellite on-board image processing. Master thesis, TU Berlin.