Pamukkale Univ Muh Bilim Derg, 32(2), 276-294, 2026

Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi

Pamukkale University Journal of Engineering Sciences

A novel verification tool suite for industrial robotic systems evaluation results

Endüstriyel robotik sistemler için bir doğrulama aracının değerlendirme sonuçları

Alim Kerem Erdoğmuş[1*], *Uğur Yayan*[2]

[1]Inorobotics, Eskişehir, Türkiye.
akerdogmus@gmail.com
[2]Department of Software Engineering, Faculty of Engineering and Architecture, Eskisehir Osmangazi University, Eskişehir, Türkiye.
ugur.yayan@ogu.edu.tr

Received/Geliş Tarihi: 21.02.2024 Revision/Düzeltme Tarihi: 03.07.2025 doi: 10.5505/pajes.2025.37200
Accepted/Kabul Tarihi: 10.07.2025 Research Article/Araştırma Makalesi

Abstract

This study introduces a new tool suite for robotic system safety and reliability in manufacturing. Developed for a vehicle chassis inspection project, it includes a simulation-based verification tool and a camera fault injection tool. Evaluated across three scenarios with robust testing, the tools demonstrated significant improvements in system efficiency and robustness. Unique for being open-source and ROS-compatible, they fill a gap in robotic system verification and validation. Results showed a 95.39% robustness in fault detection, a 27% efficiency increase in planning, and an 80% resistance to camera faults. This advancement marks a significant leap in reducing testing costs and time for industrial robotics.

Keywords: verification tool, Robotics, Industrial quality control, Safety trajectory optimization, Manipulation of sensor data, Fault injection.

Öz

Bu çalışmada, üretimde robotik sistem güvenliği ve güvenilirliği için yeni bir araç paketi tanıtılmaktadır. Bir araç şasisi denetim projesi için geliştirilen bu araç, simülasyon tabanlı bir doğrulama aracı ve bir kamera hata enjeksiyon aracı içermektedir. Sağlam testlerle üç senaryoda değerlendirilen araçlar, sistem verimliliği ve sağlamlığında önemli gelişmeler olduğunu göstermiştir. Açık kaynaklı ve ROS uyumlu olmasıyla benzersiz olan bu araçlar, robotik sistem doğrulama ve onaylama alanındaki bir boşluğu doldurmaktadır. Sonuçlar, hata tespitinde %95,39 sağlamlık, planlamada %27 verimlilik artışı ve kamera hatalarına karşı %80 direnç göstermektedir. Bu ilerleme, endüstriyel robotik için test maliyetlerini ve süresini azaltmada önemli bir sıçramaya işaret etmektedir.

Anahtar kelimeler: Doğrulama aracı, Robotik, Endüstriyel kalite kontrol, Güvenli yörünge optimizasyonu, Sensör verilerinin manipülasyonu, Hata enjeksiyonu.

1 Introduction

With the 4th industrial revolution and technological enhancements, robotic systems have become important parts of industrial processes. IoT technologies brought cyber-physical systems and autonomous operations. In today's cyber-physical systems, complexity of the robotic systems become complex. The complexity of the cyber-physical systems is continuing to increase especially with multi-level integration of subsystems with different domains. Today, with increasing complexity in cyber-physical systems, Verification & Validation (V&V) of the of systems has never been important ever before. verification is known as a process to verify correctness of the systems with respect to its requirements. Testing can be defined as a technique with an aim of showing intended and actual behaviors of a system are satisfying the requirements or not [1]-[5].

Robotic inspection systems are known as sending robots to hazardous distant or dangerous environments and robotic inspection systems have gained importance with the aim of reducing human risk. Today, robot inspection systems are being used in situations for hazards listed: chemical and radiation, explosion and fire, wind and water, pressure and heat lack of atmosphere and Inspecting structures for leaks, faults, corrosion, or for general wear etc. According to workplace, these robots may have different system properties and options in construction and programming. Variety of inspection sensors, variety of communication mechanisms can be configured. With creation of new robotic inspection systems, it is expected them to carry out more complex tasks in future. The demand on robots to do more complex tasks is increasing complexity of the robotic systems. Thus, software quality becomes crucial. In this case verification and validation of robot inspection systems will be more critical in following years [6].

2 Literature review

In robotic systems, there are many different techniques for testing and verification. Physical testing is an important practice. However, for many of the environments, there are two main issues that make test scenarios impossible to replicate. Firstly, changes on robot itself like energy, battery degradation, equipment wear etc. make test cases unique. Also, environmental changes like temperature, air, radiation etc. cause distinctive test cases. In these cases, test cases are often carried out multiple times before deployment [6]. Physical tests usually require high costs for test execution and have higher safety risks for many other test methods. For example, imitation of faults in physical systems can cause hazards. Also, in some cases where targeted environment can't be created, physical tests may not provide good results as example of space rovers. These space rovers' working environment can only be physically modelled for similar physical conditions. The

*Corresponding author/Yazışılan Yazar

physical models may not provide same testing conditions or injection of faults may not create same effect for test scenarios and also physical test suite creates additional cost and time for developers.

In verification and validation (V&V) of the cyber-physical systems for example, industrial inspection robots, V&V activities can be done with various methods. These V&V methods are differentiating each other with general method implementation and different V&V concerns. There are methods applied for verifying multiple concepts. For example, formal and model-based methods can be applied to verify and validate safety, security and privacy concerns. Besides, penetration testing method is tailored for cybersecurity. In a recent study contributed by this paper authors, have proposed classification of V&V methods in scope of safety, cybersecurity, and privacy of automated systems. In classification of these methods, systems having different levels of autonomy and different purposes can be Validated with given general method types [7], [8]. General V&V methods are considered as;

1) Fault/Attack Injection,
2) Simulation,
3) Testing,
4) Runtime Verification,
5) Formal Analysis,
6) Semi-formal Analysis,
7) Informal Analysis.

Injection methods are done by introducing phenomenon to a system and analyzing system's response. Simulation based methods observe system behavior on modelled systems. Next, testing method considers system execution under certain conditions. Similarly, runtime verification methods evaluate system during operation. Formal analysis methods execute V&V methods with mathematical basis. In contrast, informal analysis V&V methods are executed without having any mathematical basis [8]. In robotics, there are different levels of autonomy and V&V activities are done according to these levels.

In supervised or semi-autonomous systems, direct and remote control of robots can be inaccurate and fault-prone [7]. Verification and validation of these systems are traditionally done with stability analytics, safety avoidance proofs with control theory algorithms [8]. Formal methods can also be applicable. However, these methods can become intractable in case lacking of significant abstractions or approximations of environment [9],[10]. In some studies, beyond the model-based methods, Hardware-in-the-loop (HIL) and real-worlds tests are being done [11]. However, these tests can be costly.

Today, detailed computer simulations which is also called as "digital twin" in some cases, are created for virtual prototypes of examined systems [12]. These simulations generally have detailed physic models of the system and its environment. The term of simulation-based testing is based on these simulations, and it is useful for examination of critical and/or autonomous systems testing. The benefit of simulation-based testing compared to traditional V&V techniques is cost and safety [13]. Using simulators for V&V processes can eliminate hazardous situations and only leave edge cases for physical tests. Also, simulation models and environments can provide exact same conditions for different test cases. In example application for Simulation-based testing is flight simulation software (FLIGHTLAB, X-Plane etc.) include physical models and control systems in virtual 3D environment [14]. This way, development and testing can be done relatively cheap and quick. However, it is important to understand the gap between simulation and reality [6].

For the autonomous systems where software makes key decisions, safety is in a critical state and V&V of these systems is done by several techniques. In recent studies, Formal methods are used in autonomous vehicle testing with AI-based tools with simulations [15],[16]. In another study, it is indicated that traditional methods or Monte-Carlo approaches are usually not viable for hazard identification of robotic system. In given study, CoppeliaSim is used for evaluating safety-validation scenarios with simulation-based testing [17]. Furthermore, the study made by Son et al. presented simulation-based testing and validation framework for Autonomous driving systems [18]. In paper, Vehicle dynamics are created in Anesim and Co-simulated with Prescan to simulate environment and sensors and creating trajectory safety testing for autonomous driving systems. Similar study in this topic, Chance et al. presented agent-based test generation technique, novel to model-based test generation for simulation-based testing. However, in presented study, simulation and environment are lacking for real life V&V scenarios. Following, a study proposed simulation-based testing for Ship Autonomous navigation system (ANS) and situational awareness. Comprehensive mathematical models of the ship and its equipment, including all sensors and actuators are represented in digital-twin [4], [19].

In the field of software engineering, the assessment of software reliability and system safety is of utmost importance. Among the various methods employed for this purpose, fault injection has gained significant attention. Practitioners in the software development, research, and engineering domains utilize fault injection methods to test the system's robustness against injected faults. These methods involve inducing faults in software-based systems and devising new techniques that can be applied to both hardware and software. A notable fault injection method involves the injection of faults into hardware components, such as chip pins, internal circuits, and registers. Such faults cannot be rectified through software modifications. In contrast, injecting faults into software can cause a direct alteration in the overall state of the software. Consequently, hardware methods are suitable for evaluating low-level fault detection and masking mechanisms, while software methods are appropriate for testing higher-level mechanisms [20].

Several studies have explored various software and interfaces that facilitate fault injection. Notable examples in the literature include GemFI by Parasyris et al., GOOFI by Aidemark et al., SASSIFI by Hari et al. and MODIFI by Svenningsson et al. [21]-[24]. These studies have developed fault injection tools that enable testing for fault tolerance and system weaknesses in diverse software, simulation, or hardware systems.

Robotic inspection systems are intricate machines that need sophisticated tool sets to check each component. This work uses a novel suite of verification tools for industrial inspection robots to go through this kind of verification. The Simulation-based Robot Verification Tool (SRVT) and the Camera Fault Injection Tool (CamFITool) are the two instruments that make up this tool suite [25]-[27]. The proposed tool suite is demonstrated on the verification of the "Safety Trajectory Optimization", "Manipulation of Sensor Data" and "Anomaly Detection at Component and System Level" evaluation

scenarios from the VALU3S ECSEL H2020 project. In the VALU3S project, OTOKAR describes the use case scenario "Autonomous Robot Inspection Cell for Vehicle Chassis Quality Control" with difficulties in the robotic inspection system that have been found. In order to validate the safety trajectory for robotic systems, SRVT is utilized to identify the fastest and safest trajectory planning technique. CamFITool is a tool for injecting faults into camera sensor data to test the stability of camera-based perception systems and detect anomalies at images. One of the uniqueness of this V&V Tool suite is open-source and Robot Operating System (ROS) compatible are [28]. Such tools are not common in the ROS environment. In this study, a verification tool suite is applied to a specified use case scenario, and evaluation outcomes are provided in relation to evaluation scenarios. The suggested verification tool suite successfully ensures the required key performance indicators (KPIs) for each evaluation scenario. In the following section, use case scenario 'Autonomous Robot Inspection Cell for Vehicle Chassis Quality Control' which is used in this study and evaluation scenarios are explained. In Chapter 4, proposed verification tool suite (SRVT and CamFITool) is explained in detail. In Chapter 5, contains tool qualification reviews of the SRVT and CamFITool tools. In Chapter 6, Industrial Robotic body-in-white Inspection Cell Verification tool suite evaluation scenario results are given. Conclusion and future works are in Chapter 7.

3 Use case overview

The targeted use case involves using new visual inspection techniques to improve automotive body-in-white inspection. The goal is to provide a more fault-tolerant production system and improve quality control. A cartesian robot and camera sensor system will automatically check the existence of 2500-3000 body parts using digital twin software and CAD data. The system will use a programmable logic controller to position the cartesian robot, and sensors will capture 2D images to compare with synthetic 2D images stored in a server. Quality reports and system status data will be stored for the quality control team to review and give final confirmation [25].

Given use case is defined as Use Case 11 in the VALU3S project is aiming to provide a better fault-tolerant production line to achieve better quality control for bus body- in-white [26]. The components of the robotic inspection system are given in Figure 1.

Quality control has been carried out by means of the camera system positioned on the cartesian robot located on both sides of the vehicle body (i.e bus). The data obtained from the CAD data of the large-bodied vehicle is compared with the actual data obtained from the camera system by means of the synthetic data obtained from the developed data, and the item presence-absence check and critical measurement controls acquired from sensor and actuator as shown in Figure 2.

Robotic inspection system (ROKOS) camera-based quality control system is a software that analyzes the deficiencies of the parts of the bus body-in-white inspected by ROKOS. The software, which sends the body-in-white pictures taken by ROKOS, gives information about how many parts are missing in the body-in-white. Software interface could be seen at Figure 3.

To ensure existence of vehicle parts and quality, verification tool suite is applicable to the robot inspection cell for quality control. Use case application will cover an automated fault injection application, specifically for controlling the entire industrial automated line.

The existing Quality check processes still very long and ineffective without advanced safety concepts. Additionally, Quality check in existing manufacturing environment is not very responsive and adaptive to online sensing. It works in Stop&Go mode to provide the safety.

Despite the advantages of the robotic inspection system (ROKOS) system, there are also some disadvantages. The ROKOS system works with a task-based working model, where robot arms move to the points where it will record photos and the commands for taking pictures are defined as a task. However, not all joints of the robot arms work simultaneously in the currently used system. This creates a restriction on the robot arm movement trajectory. Due to this restriction, various reset points have been added to the task lists so that the robot arms can move without collision. Robot arms can go to these reset points between some tasks. This leads to a longer term of operation. Thus, in use case, evaluation scenarios were determined for the correction and improvement of these situations and these scenarios were started to be implemented with SRVT and CamFITool.

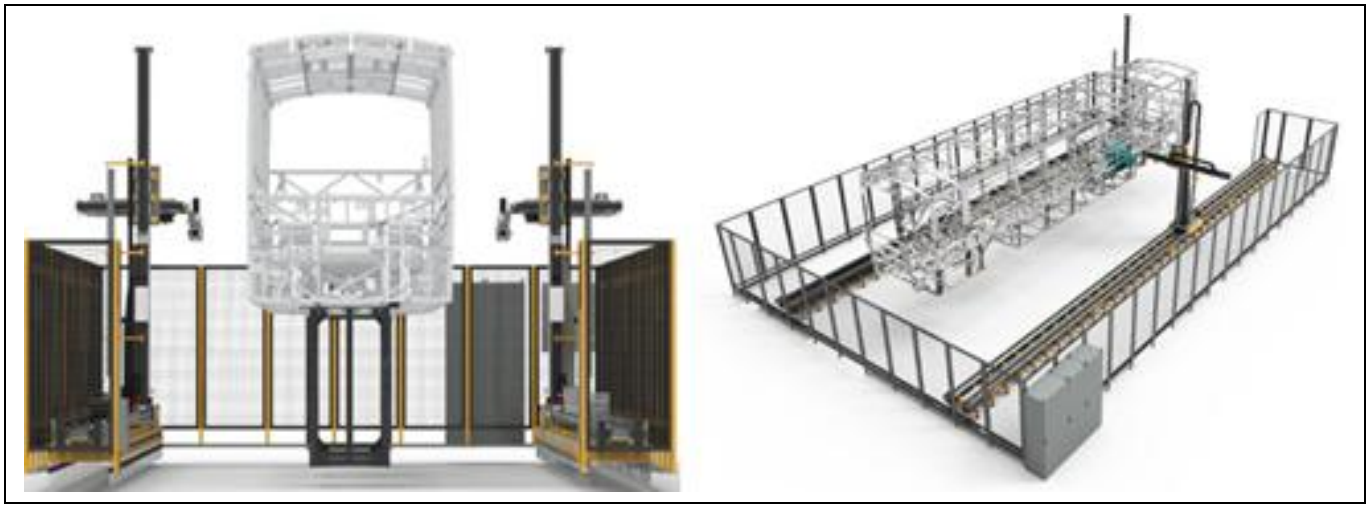

Figure 1. Robotic inspection cell for quality control.

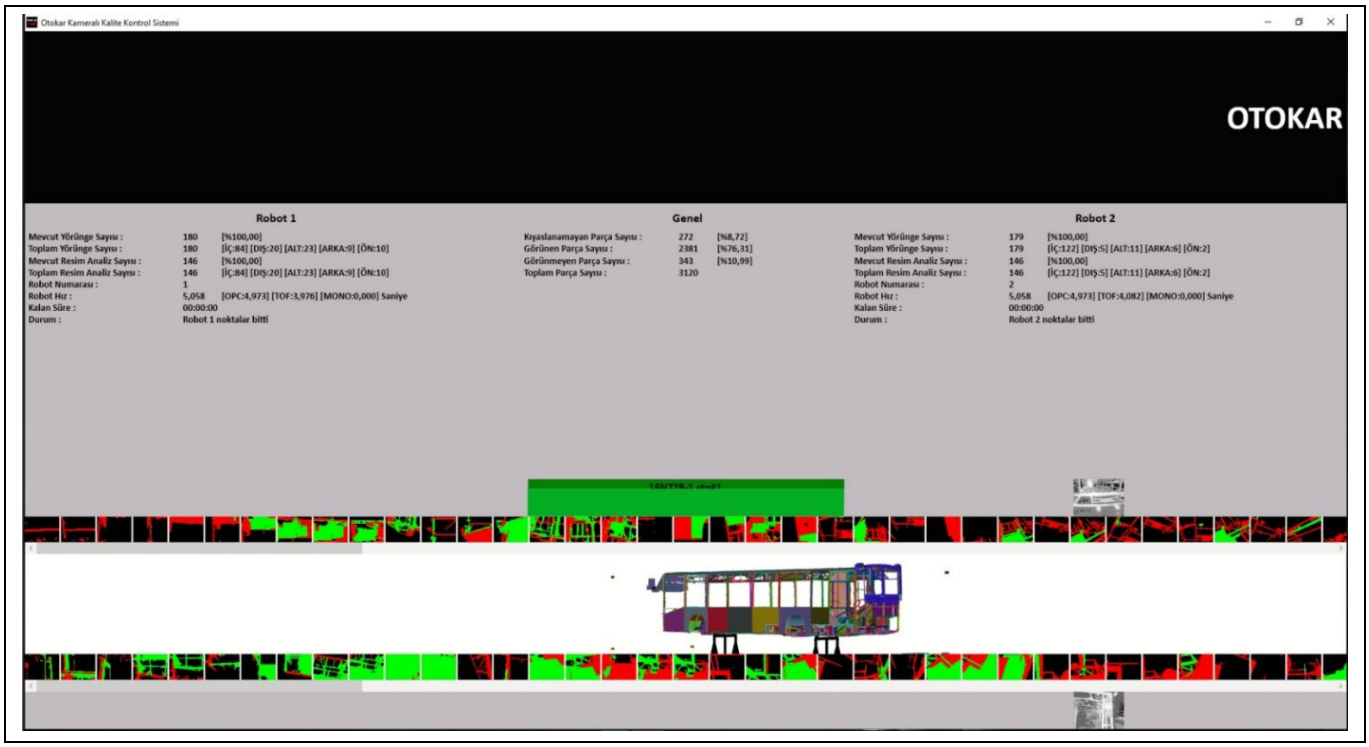

Figure 2. OTOKAR Camera Quality Control System interface.

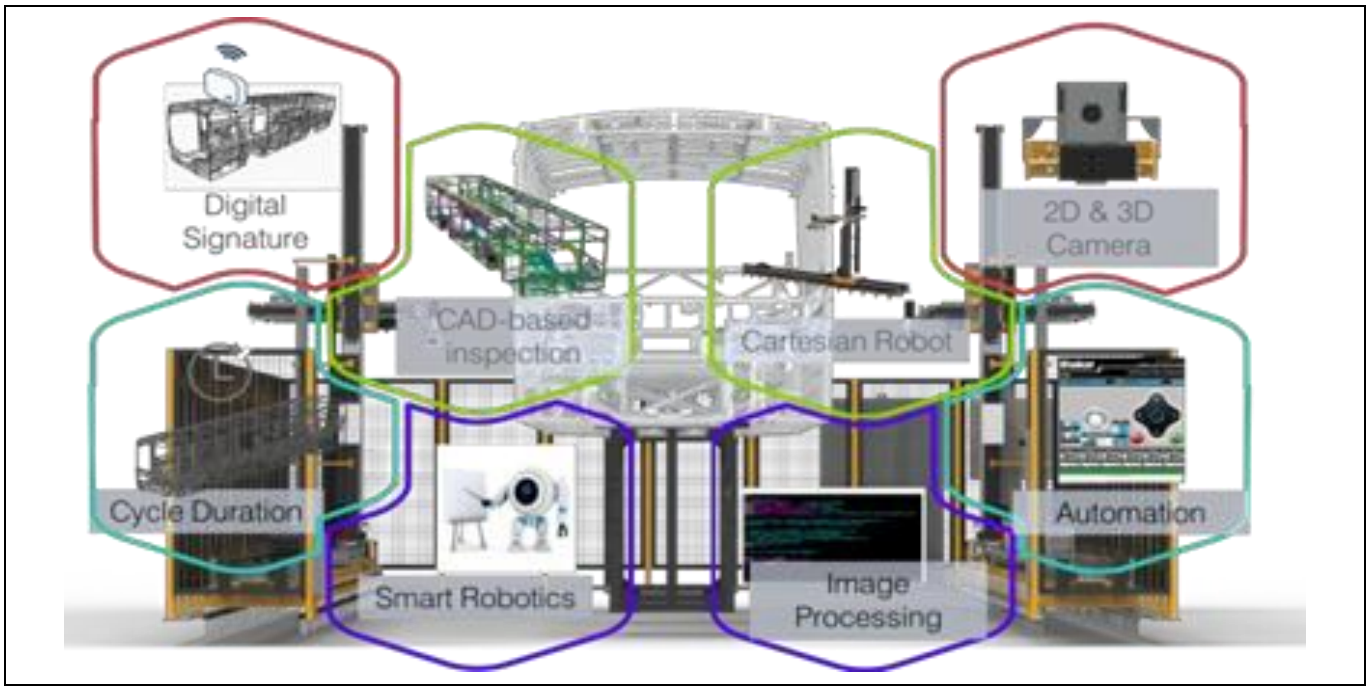

Figure 3. The components of body-in-white inspection system for world-selling OTOKAR buses.

The evaluation scenarios and criteria, which are the requirements of the robotic system to be determined and developed in this direction, are in Table 1 and Table 2.

4 Proposed verification tool suite

The targeted use case involves using new visual inspection techniques to improve automotive body-in-white inspection. The goal is to provide a more fault-tolerant production system and improve quality control.

4.1 Simulation-based robot verification (SRVT)

SRVT can be thought of as a tool that allows a robotic system to be transferred to the simulation environment and applied to verification tests [25]. The base of the system is the coordinated use of some critical software for the ROS ecosystem (see Figure 4).

Table 1. VALU3S Evaluation Scenarios.

Code	Eval. Scenario Name	Description
ES1	Manipulation of Sensor Data	Manipulation of sensor data stream at camera and safety sensors.
ES2	Safety Trajectory Optimization	Creating robot trajectory points automatically covering the safety of the robot and its apparatus as well as static objects in the workspace.
ES3	Anomaly Detection at Component and System Level	Observing machine learning and/or pre-training data in the process of reviewing and predicting normal detections in data checks

Table 2. VALU3S Evaluation Criterias.

Code	Eval. Criteria Name	Description	Measured Artifacts
EC1	Number of malicious attacks and faults detected	The evaluation criterion measures the effectiveness of detecting malicious attacks and faults in the System Under Test (SUT), defined by a risk assessment specific to the system's context and application.	Number of detected faults
EC2	Simulation-level system robustness	This criterion evaluates the robustness of the system by assessing the quality of the tested software for poor architectural and coding practices that could result in operational risks or costs, taking into account non-software components as well.	System Robustness
EC3	Effort for test creation	This criterion deals with the estimation of effort for deriving and/or maintaining test suites, e.g., for fault injection and runtime verification campaigns (manual design vs. model- based generation)	Time (lower the better)
EC4	Effort needed for test	This evaluation criterion measures the effort required to perform a test on a system, including dataset generation, execution of the test cases, and result validation. It is used to compare the effort spent on manual versus automated work.	Total person-hours cost

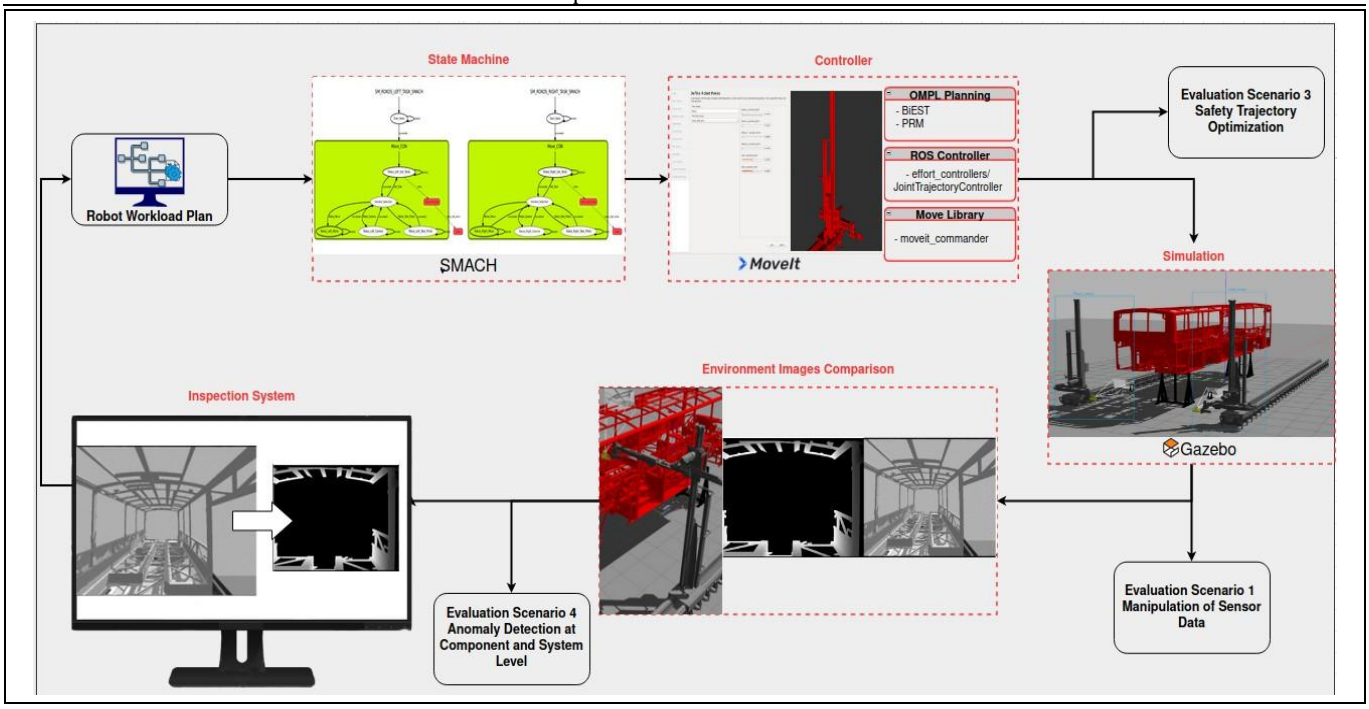

Figure 4. SRVT Architecture.

In SRVT, Gazebo Simulator is used for Simulation, MoveIt is used for trajectory planning [29], ROS SMACH packages are used for finite state machine monitoring, and a dynamic verification system was created in a single ROS package [30]. Besides, there are two other nodes which are Image and Task Server.

The SRVT approach involves remodeling the ROKOS robot arms and bus body-in-white within the Gazebo simulation environment (refer to Figure 5). The Gazebo simulator is a widely used simulation engine in the domain of robotic simulation, owing to its realistic physics engine and seamless integration with the ROS system. The ROKOS robotic system, with its actual physical measurements and values, was transferred to Gazebo and employed in the simulation [31].

MoveIt is used for controlling the robot arms in the simulation. MoveIt inculdes open-source trajectory planning algorithms in the library [32],[33]. In Figure 6, Moveit interface with bus body-in-white could be seen. In the study, it has been studied to improve the safe trajectory planning performed by SRVT Moveit and the completion times of ROKOS on this route, and to determine the ideal planning algorithm [26].

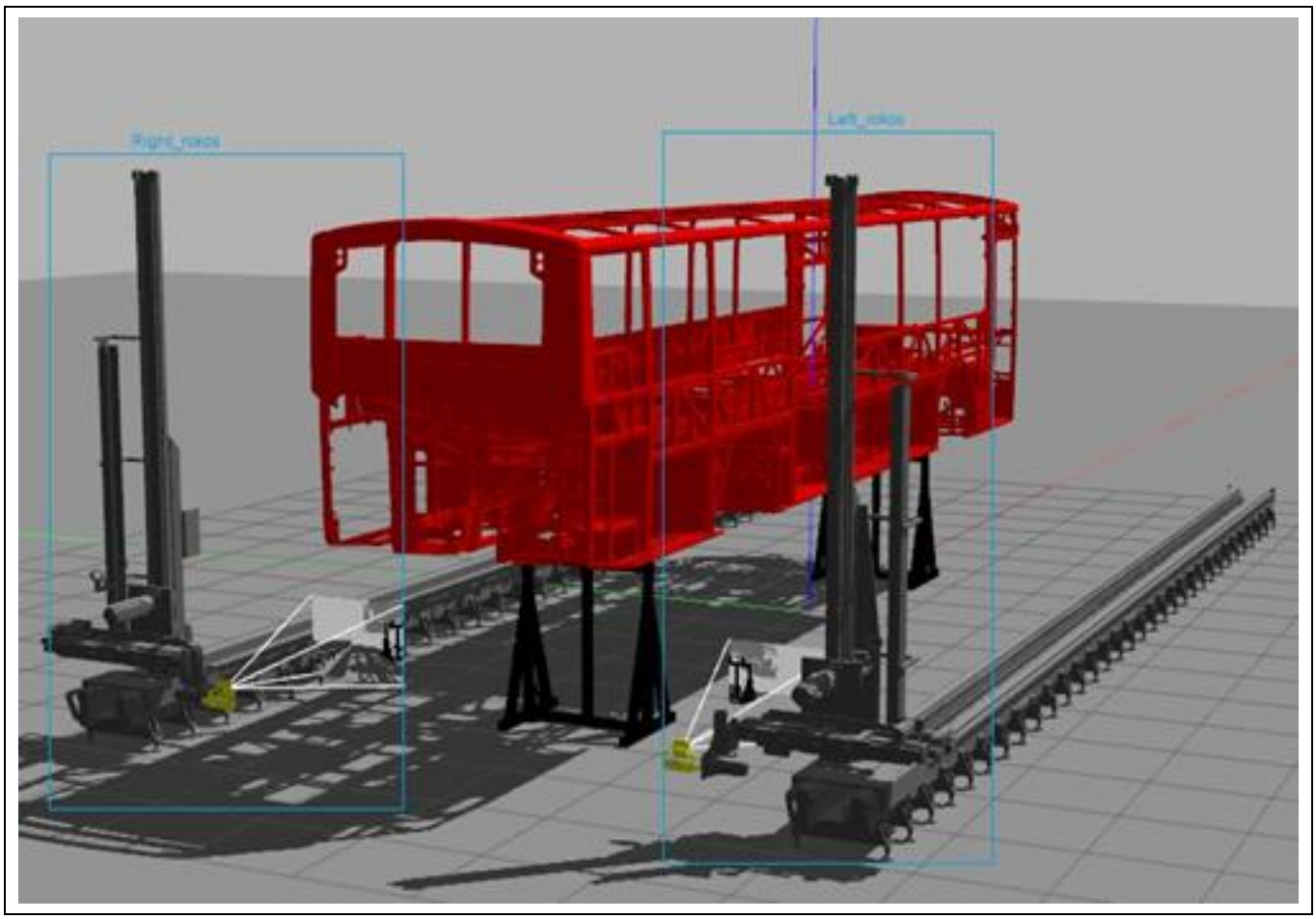

Figure 5. ROKOS modeled on the Gazebo simulation environment.

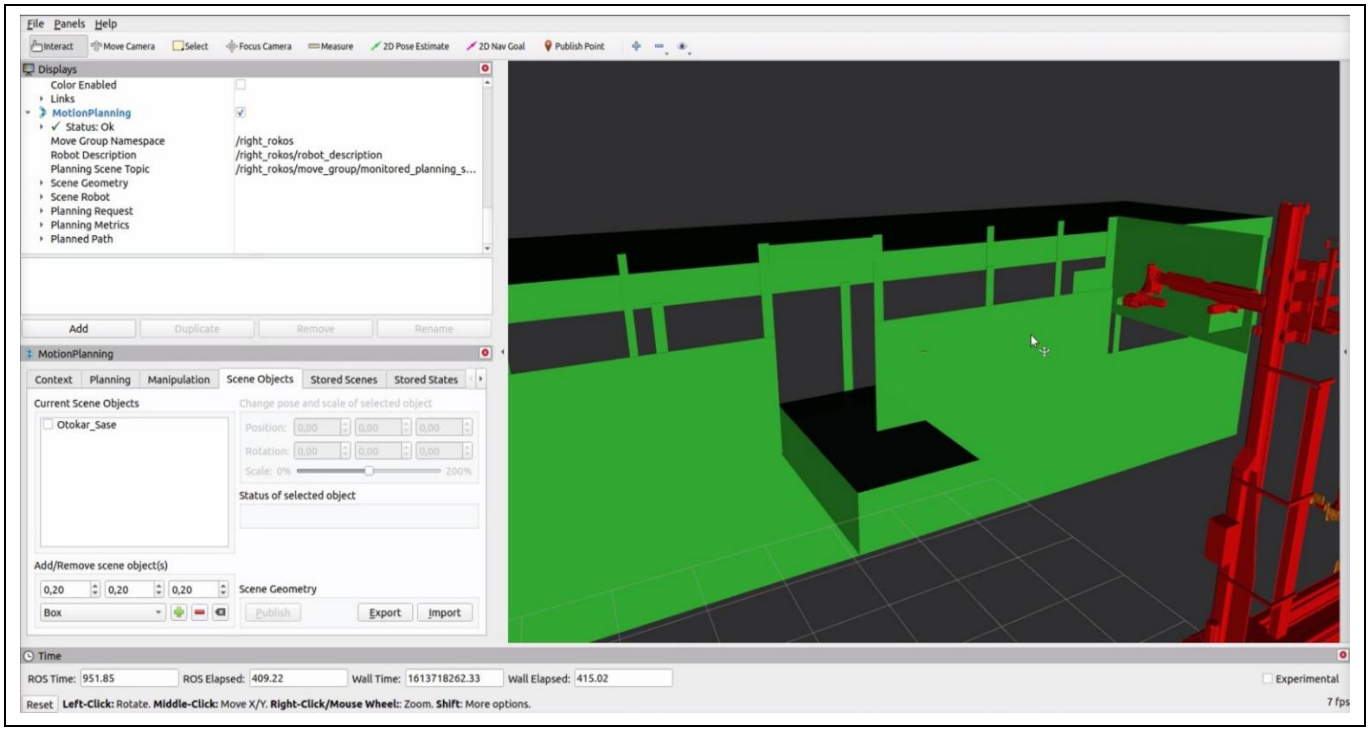

Figure 6. Bus body-in-white added to the planning environment and use of the Moveit planner.

SRVT SMACH node serves as finite state machine for controlling SRVT behaviors. This node connects with each of other nodes and sends commands to them. In addition, the SRVT SMACH node records the executed task information and the time it was performed and the motion planning data. Looking closer at Figure 7, the states that the SRVT SMACH node includes are: Left Move, Left Camera, Left Take Photo, General Selection and Left Get Tasks.

SRVT Image Server is formed by the operation of the ROS node, which enables the SRVT to take camera images from the simulation environment. SRVT Task Server node reads the tasks of the left and right ROKOS robot arms from the task files.

These tasks can also be filtered according to the Task ID value if desired. If the value of ROKOS is None, all tasks are sent to the client. The ROKOS SMACH node makes a request to the Task Server node if a task is not available. In response, Task Server receives the coordinate and orientation information that the robot arm needs to apply, as well as the image information it will take. This information includes Task ID, vehicle number and tag information.

While performing tasks, every transaction made by the robots is recorded. These records, which contain information such as robots' time to perform tasks, applied motion values, and task codes, provide users with the opportunity to examine and solve the problem in case of a potential problem.

4.2 Camera fault injection tool (CamFITool)

The Camera Fault Injection Tool (CamFITool) is an open-source tool that leverages cutting-edge vision-based fault injection techniques to inject faults into RGB and TOF cameras. The tool is specifically designed for performing verification and validation activities on robotic systems [27]. CamFITool is implemented using Python and features a Qt5 interface. Moreover, the tool is compatible with ROS Noetic, as depicted in Figure 8.

With Realtime fault injection, CamFITool can perform fault injection to any cur- rently running ROS camera stream. This injection can be made into RGB camera broadcasts with six different fault types (TOF camera support will be added in the next updates). During the injection, the relevant broadcast can be viewed on the screen to be opened by pressing the "Robot Cam" button in CamFITool (Figure 9).

Another feature of CamFITool is anomaly detection. This feature is related with Evaluation Scenario 3. In Figure 10, CamFITool Anomaly Detection screen could be seen.

CamFITool uses CNN algorithm for Anomaly detection in the images taken from ROKOS or SRVT system. An interface plugin has been developed for CamFITool (Fault Anomaly Detector Plugin - FIAD) that allows detecting anomalies in images and detecting faults in faulty images. This interface enables the use of models trained with CNN algorithms. The Binary Classification model, which determines whether the picture is faulty or not, and the Multiclass Classification models are trained to determine which faults the pictures have. The binary one of these models has been trained with libraries with different fault rates of 6 fault types (Dilation, Erosion, Gaussian, Gradient, Poisson, Saltpepper), 100 different images from each other, and 66 images determined as test images. It consists of a total of 1200 images (600 normal, 600 faulty) training set and 397 images (199 normal, 198 faulty) images test set. The multiclass model is based on 6 different fault types. 500 trainings different from each other, with different fault rates for each fault type, were trained with 100 test pictures (a total of 3000 trainings, 600 test pictures) [27].

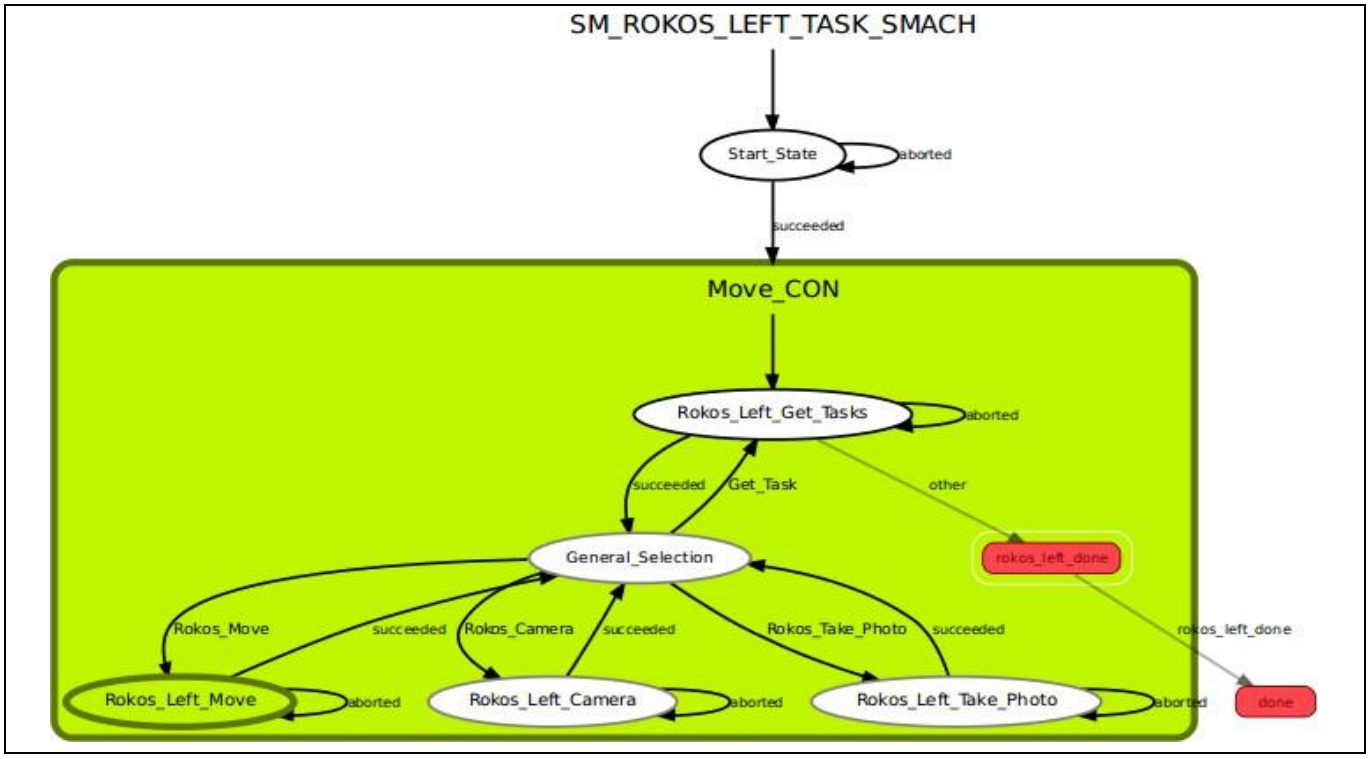

Figure 7. SMACH interface of left ROKOS arm.

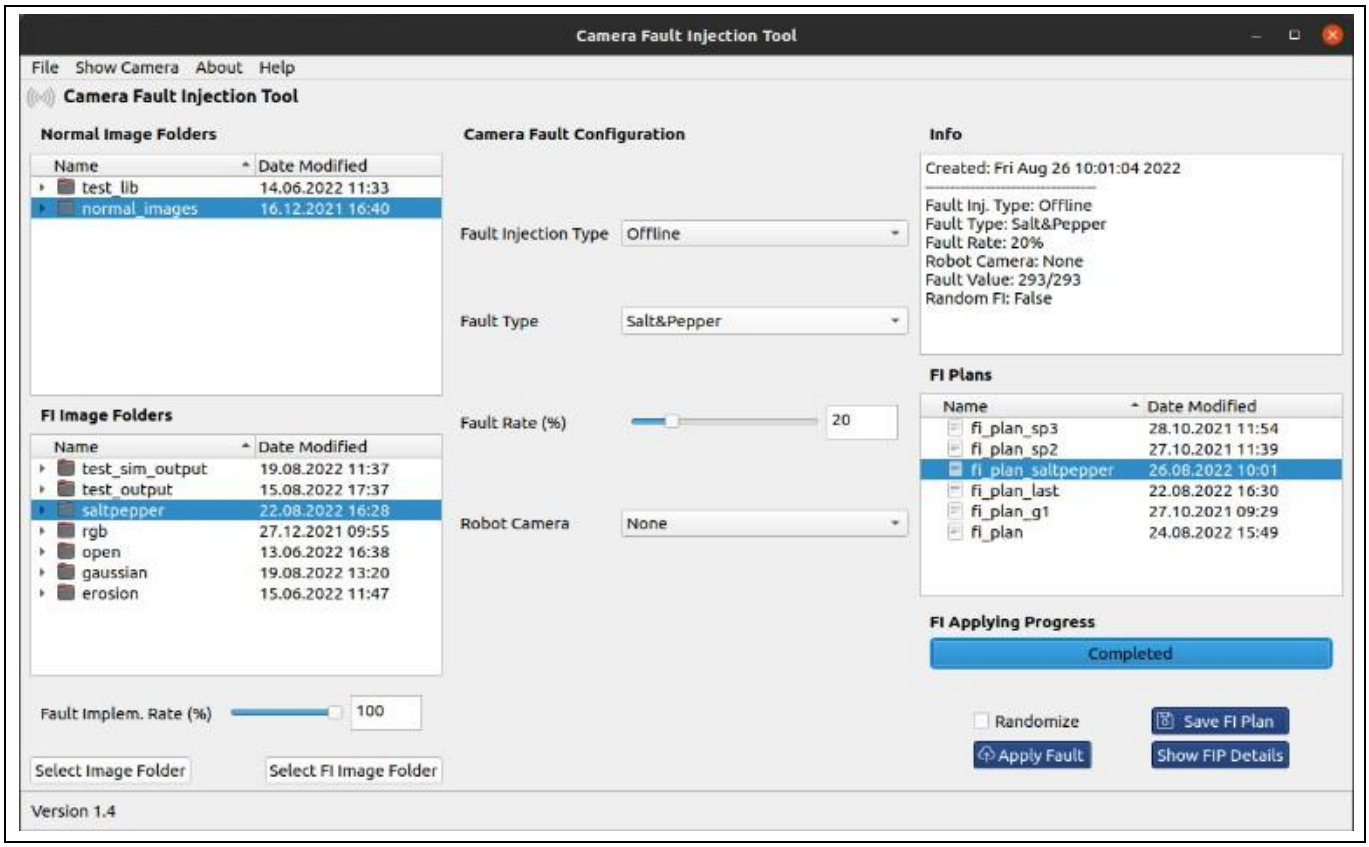

Figure 8. CamFITool offline FI screen.

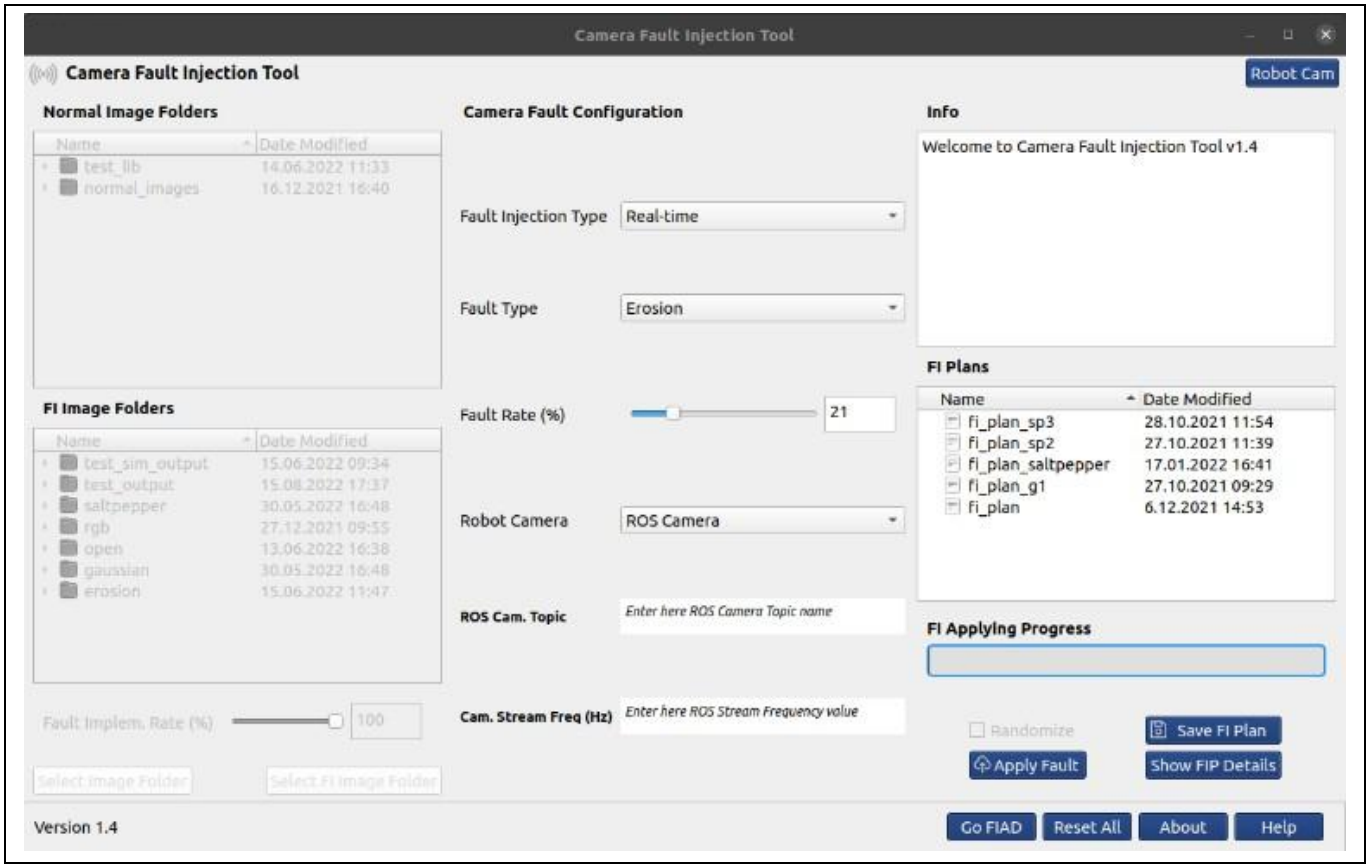

Figure 9. CamFITool realtime FI screen.

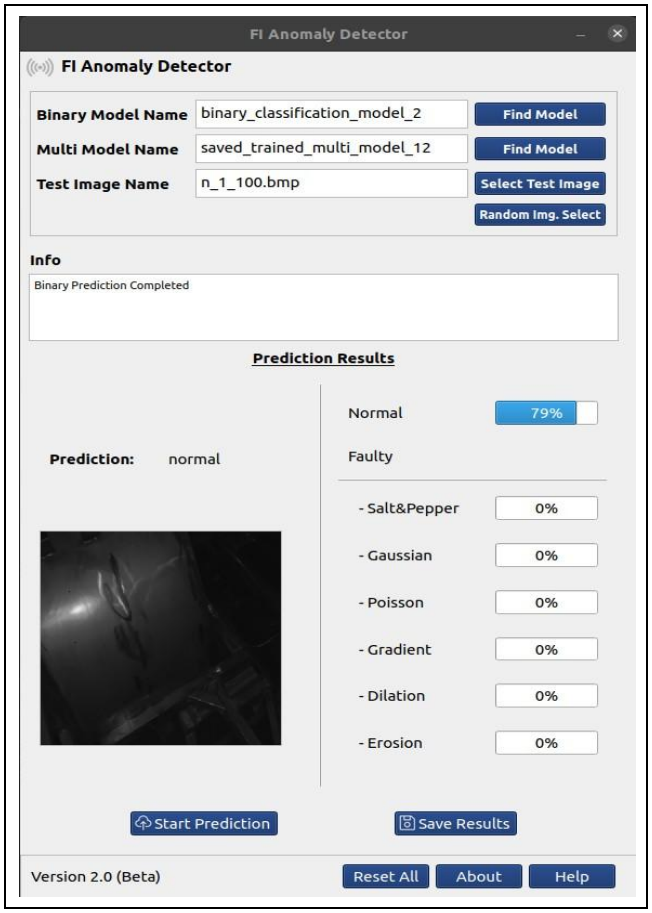

Figure 10. CamFITool Anomaly Detection screen.

5 V&V tool suite qualification assessment

A procedure called tool qualification enables us to show that a tool may be utilized to realize a software application with a predetermined safety target.

5.1 Qualification approach

1. Tool impact: If a tool can introduce bugs into the code or is inefficient at dis- covering errors, it is assumed that it will have an effect on the final product. Tool Impact 2 refers to this situation. If the tool has no effect on the output of the software, it comes under TI 1.

2. Tool error detection: The tool's use-case scenarios help decide whether it has enough safeguards to both detect and stop malfunctioning and inaccurate output. The degree of confidence in mistake detection is determined by the tool's capacity.

3. Tool confidence level: Tool Confidence Level extraction is simple to complete once Tool Impact and Tool Error Detection levels have been established. The inferences drawn lead to the following tool confidence levels:

a) TCL 1 has the lowest level of confidence. This tool's negligible influence on the final product's quality. As a result, tool qualification is not required,

b) TCLs 2 and 3 have medium and high degrees of confidence, respectively. A qualification needs to be offered in order to demonstrate how trustworthy a tool is.

Based on the ISO 26262 standards of SRVT and CamFITool, the components that make up the tool suite we have developed, the measurement of software quality and the compatibility of standards with these software tools are also provided.

The ISO 26262 standard is a standard that covers the development process of soft ware used in critical security applications. ISO 26262 also covers tools used for the development and testing of tools and also stipulates their qualification. The qualification requirement is determined by whether the tools can cause a defect in the end product and the detectability of that defect [34].

ISO 26262 defines three different levels of trust for tools:

- TCL1 (low impact)
- TCL2 (moderate)
- TCL3 (high impact)

While qualification is required for TCL2 and TCL3, tool qualification is not required for TCL1. Tool qualification is done by following two steps that determine the qualification's requirement. In the first step, it is determined whether the tools can cause a defect in the final product. In the second step, when the vehicle generates a fault, its detectability is evaluated. ISO 26262 recommends four different methods for tool qualification. These:

a) Confidence increased by previous use,

b) Development process evaluation,

c) Verifying the software tool,

d) Development in accordance with the security standard.

5.2 Modules and tool qualifications

The tool qualifications of the SRVT modules are listed in Table 11 (in the Appendix A). According to the reports generated during the SRVT development process, the levels of faults obtained with the safety gain were primarily focused in groups A, B, and C. The inferences drawn from the continuation of the development process and parallel testing of safety issues concentrated the fault levels of the development process under A and B. The error levels were primarily under C and D due to the safety implications of the issues discovered during the validation of the software tool. The safety levels achieved through improvements according to ISO 26262 standards were generally concentrated under D, indicating that the safety levels of the developments in accordance with ISO 26262 standards are high.

The tool qualifications of the CamFITool modules are listed in Table 12 (in the Appendix A). During the development of CamFITool, the levels of errors obtained with the security gain were thoroughly analyzed and reported in groups A, B, and C. The development process and concurrent testing of security issues led to the conclusion that the fault levels were mainly concentrated under A and B. However, the error levels were mainly under C and D due to the security implications of the issues identified during the software tool's validation. The security levels achieved through improvements according to ISO 26262 standards were generally concentrated under D, indicating that the security levels of the developments in accordance with these standards are high.

6 Industrial robotic body-in-white inspection cell application verification results

To demonstrate applicability and effectiveness of the V&V tool suite, proposed tool suite is applied to Industrial robotic inspection cell use case. In scope of application, previously explained evaluation scenarios are studied in this chapter. In this manner, manipulation of sensor data and anomaly detection at component and system level are realized through CamFITool and trajectory safety optimization scenario is realized with SRVT. In this chapter, description and execution of test cases are explained for each use case scenario. Results obtained from these three evaluation scenarios are shown in detail. With test results, effects and improvements done by V&V tool suite are explained in this section.

6.1 Manipulation of sensor data

The fault injection software on Robotic Camera systems is turned into a fault injection interface called CamFITool. This interface runs and injects faults into a database of images on robot video recording cameras during the mission, and various test scenarios were performed with these images. Evaluation Scenario I has been evaluated with these tests and the expansion of the fault injection system. Test Cases (TC) is defined for VALU3S project Evaluation Scenario-1 in Table 3.

Table 3. VALU3S evaluation scenario-I test cases.

Test Case	Test Case Desc.	Expected Results
TC 1	Fault Injection to Robotic System	Robot system will handle faults and continue operating in normal mode
TC 2	Fault Injection to Robotic System	Robot system will detect fault and report situation.
TC 3	Fault Injection to Robotic System	The camera quality control system will detect the fault and report the situation.
TC 4	Fault Injection to Robotic Camera System	The camera quality control system will detect which faults have been injected according to varying fault types and rates.

The developed fault injection software has been developed to prevent the cameras from injecting faults into the cameras of the ROKOS robot arms while the system is running, and to prevent the cameras from performing the inspection by saving the faulty images. The purpose of this software is to create a product under the VALU3S project to create a V&V system in robotics [35].

Virtual environment faults are artificial image faults that manifest exclusively in simulation environments and can be detected only through software, such as OpenCV Python libraries [36]. Examples of virtual environment faults include Dilation/Erosion, Open/Close, Gradient, and InjectionPayload faults. Unlike real environment/hardware faults, virtual environment faults cannot be detected with non-software methods.

Conversely, real environment faults (also known as hardware faults) are image faults that may arise not only due to software issues but also due to problems with the camera hardware in real-world environments. Examples of real environment faults include Motion-blur, Partialloss, Gaussian, Poisson, and Salt&Pepper faults. For instance, in electronic circuits, Poisson noise may emerge from the random fluctuations of electric current in a Direct Current, caused by the discrete flow of charges (electrons), potentially resulting in related corruption in camera hardware.

Fault injection was applied to 293 normal images whose are taken from ROKOS cameras, with nine different fault types, different rates and amounts. Afterwards, image libraries consisting of fault-injected images were given as input to the camera quality control software, and comparisons were made with the results obtained by examining the normal image library of the software.

As can be seen in Table 4, nine different camera faults supported by two different camera types were injected into a normal image library taken by ROKOS cameras. Although the pictures taken from the ROKOS camera are TOF camera type, the effect of fault types applied to RGB cameras on the software has also been examined. Three different application tests were determined for each fault with 5%, 20% and 40% fault rates. These fault rates are varied to apply to 10% and 30% of the image library. Adding 20% fault injection test to the entire image library, seven fault plans from each fault method were tested. With this calculation, a total of 49 different fault injection plans were tested in the study.

In Table 5, Multiple Fault Injection Test configurations are shown. In this configuration, tests were carried out using the specified rates for both fault types and fault injection at the rate of the image applied fault.

In Table 6, the outputs of the normal image database obtained from the ROKOS system in the quality control detection software are given.

The faulty image libraries obtained because of fault injections applied using CamFITool were passed through the quality control system software one by one. The results obtained from each fault method are analyzed in the following subsections.

It was observed that the quality control system gave “more visible pieces” output as the fault rate increased in the image libraries injected with Salt&Pepper, Gaussian and Poisson faults. The increase in fault rate in image libraries injected with Open fault affected the output very little (effect 0.3%). Likewise, the number of faulty pictures did not change this situation much. The increase in fault rate in image libraries injected with the Close, Dilation and Motionblur faults affected the output at a very low rate. Likewise, the number of faulty pictures did not change this situation much. It has been observed that the quality control system outputs more” invisible pieces” as the fault rate increases in image libraries injected with Erosion and Gradient faults. By using CamFITool, faults were injected into the 293 body-in-white image library obtained from ROKOS robot cameras, and the tests and results of the OTOKAR camera quality control system software’s responses to these images were compiled. With 49 different fault injection tests with 9 different fault types, different fault rates and wrong image amounts, which fault injection affected the system and how it was investigated. As a result of these tests, the effects of defect types on the “number of visible pieces” are shown in Table 7.

Table 4. Fault methods and fault rates implemented with CamFITool (single fault injection tests).

Cam Type	Fault Method	Test Code	App. Fault Rate (%)	Fault Img. Rate (%)	Fault Img. Amount
TOF	Salt&Pepper	SP	5-20-40	10-30-100	29-88-293
TOF	Gaussian	G	5-20-40	10-30-100	29-88-293
TOF	Poisson	P	5-20-40	10-30-100	29-88-293
RGB	Open	O	5-20-40	10-30-100	29-88-293
RGB	Close	C	5-20-40	10-30-100	29-88-293
RGB	Dilation	D	5-20-40	10-30-100	29-88-293
RGB	Erosion	E	5-20-40	10-30-100	29-88-293
RGB	Gradient	GR	5-20-40	10-30-100	29-88-293
RGB	Motionblur	MB	5-20-40	10-30-100	29-88-293

Table 5. Fault methods and fault rates implemented with CamFITool (multiple fault injection tests).

Cam Type	Fault Method	Test Code	App. Fault Rate (%)	Fault Img. Rate (%)	Fault Img. Amount
TOF	Salt&Pepper + Gaussian	SPG	5-20-40	10-30-100	29-53-146
TOF	Gaussian + Poisson	GP	5-20-40	10-30-100	29-53-146
TOF	Poisson + Salt&Pepper	PSP	5-20-40	10-30-100	29-53-146

Table 6. Quality control system software output of ROKOS normal image database.

Database Name	Visibility Rate (%)	Visible Pieces	Invisibility Rate (%)	Invisible Pieces	Incomplete Part Rate (%)	Incomplete Part Pieces	Total
Normal Image Data	76,31	2381	10,99	343	8,72	272	2996

Table 7. Effects of defect types on the number of visible parts on the basis of applied tests.

Fault Type	Effect	Average Change Rate
Salt&Pepper	Increased	4.38%
Gaussian	Increased	4.75%
Poisson	Increased	4.71%
Open	Ineffected	-0.30%
Close	Ineffected	-0.01%
Dilation	Low Effected	1.49%
Erosion	Decreased	-3.16%
Gradient	Decreased	-4.24%
Motionblur	Ineffected	-0.01%

In the fault injection tests, the image fault injections are performed with Salt&Pepper, Gaussian and Poisson faults, which are defined as TOF camera faults, manipulated the inspection software by an average of 4.61%. This manipulation led to the conclusion that ROKOS detects more parts in the bus chassis analysis, thus reducing the number of missing parts and deceiving the system. Considering that the analysis software re-examines the relevant chassis when it finds more than a certain number of missing parts, the probability of continuing the production of a bus chassis with missing parts increases as a result of a fault in the camera. While Open, Close and Motionblur faults in the tests did not affect the analysis software, the Dilation fault, which is called one of the RGB faults, caused the number of missing parts to be high (1.49%), albeit at a low rate. Erosion and Gradient faults, on the other hand, caused the number of missing pieces to be higher at an average rate of 3.7%.

6.2 Safety trajectory optimization

The SRVT platform is utilized to verify and validate the Safety Trajectory Optimization evaluation scenario. To this end, ROS MoveIt tool's OMPL (Open Motion Planning Library) and EST (Expansive Space Tree) trajectory planning algorithms were employed, and their performance evaluations were determined. To test the effectiveness of these algorithms in a simulated environment, comprehensive tests were conducted on a use-case scenario, and the results were analyzed. Three different scenarios, namely Quick test, Full Test with Reset, and Reset-Free Full Test were evaluated, and the validation activities were improved in terms of time and cost for the ROKOS system transferred to SRVT [25]. Test cases for the Evaluation Scenario-2 of the VALU3S project are defined in Table 8.

In SRVT, RRT, RRT*, RRTConnect, PRM, PRM*, EST (Expansive Space Trees) and BiEST (Bidirectional Expansive Space Trees) algorithms were used [26]. The results of these algorithms in three tests were compared, and the effects of changing test conditions on the task completion time were examined. Before the application of the test methods determined for the trajectory planning algorithms, the time to find the planning and plan implementation solutions of these algorithms were also tested Planning Setup (SimpleSetup) and Solution Finding (SolutionFound) values. The mean and standard deviation values of these time values of the algorithms are shown in Table 9. These calculations are taken from the current ROKOS system.

The SimpleSetup times in the table above give the times when the algorithms perform the planning. SolutionFound times are the time they have these plans applied to the robot arms. These times were calculated by analyzing the times obtained from all the motion plans made by the robot arms during a task. The time graphs obtained with these times are as given in Figure 11, Figure 12.

Table 8. VALU3S evaluation scenario-II test cases.

Test Case	Test Case Description	Expected Results
TC_5	Robot Trajectory Planning	The joint values of the robot tracking the trajectories keeps within the robot's joint limits.
TC_6	Robot Trajectory Planning	The velocities and accelerations of joints of the robot tracking the trajectories keeps within the robot's joint limits.
TC_7	Robot Trajectory Planning	The jerks of joints of the robot tracking the trajectories keeps within the robot's limits.
TC_8	Robot Trajectory Planning	The robot does not collide with itself and automotive body.
TC_9	Robot Movement According to the Trajectory Plan	The robotic system should avoid from obstacles within the limits of safety.
TC_10	Robot Movement According to the Trajectory Plan	The robot obeys safety-rated stop/speed and separation monitoring requirements of the standard ISO15066:2016 and ISO10218- 2:2011.
TC_11	Robot Trajectory Test	Existance control of minimum 95% parts of vehicle in less than 25 minutes and in each trajectory point minimum 15% of each part must be visible.
TC_12	Task Safety in Faulty Situation	Mutating operations with faulty situations. System does not cause any unsafe movement or behaviour.
TC_13	Task Safety in Faulty Situation	Manipulating data transfer nodes. System does not cause any unsafe movement or behaviour.

Table 9. SimpleSetup and solutionfound average and standart deviation times table.

Plan Alg.	Simple Setup Avg. (sec)	Simple Setup St. Dev (sec)	Solution Found Avg. (sec)	Solution Found St. Dev. (sec)
RRT-Left	0.0279	0.0309	0.829	0.788
RRT-Right	0.0700	0.0837	0.6766	0.8448
RRT*-Left	0.0278	0.0308	0.8298	0.7885
RRT*-Right	0.0952	0.098	3.3718	1.5641
RRTCon.-Left	0.0058	0.0075	0.3990	0.2824
RRTCon- Right	0.0084	0.0088	0.5043	0.6409
EST-Left	0.0250	0.0247	0.8178	0.7611
EST-Right	0.0804	0.1019	0.9056	1.3322
BiEST-Left	0.0088	0.0076	0.5275	0.6486
BiEST-Right	0.0413	0.0566	0.5314	0.5798
PRM-Left	0.0317	0.0405	1.0742	1.0307
PRM-Right	0.1008	0.1049	0.9218	0.7563
PRM*-Left	0.0375	0.0425	4.2290	0.5992
PRM*-Right	0.1157	0.1215	4.0598	1.2331

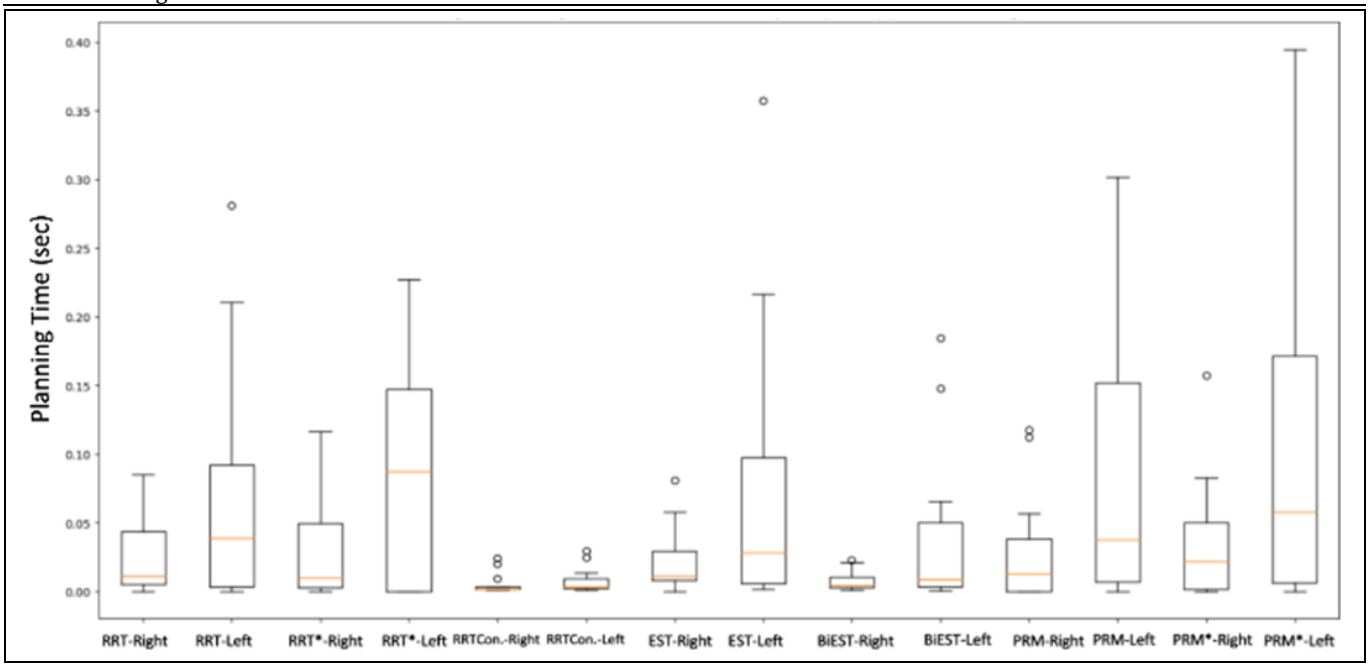

Figure 11. SimpleSetup times graph for each robot arm of trajectory planning algorithms

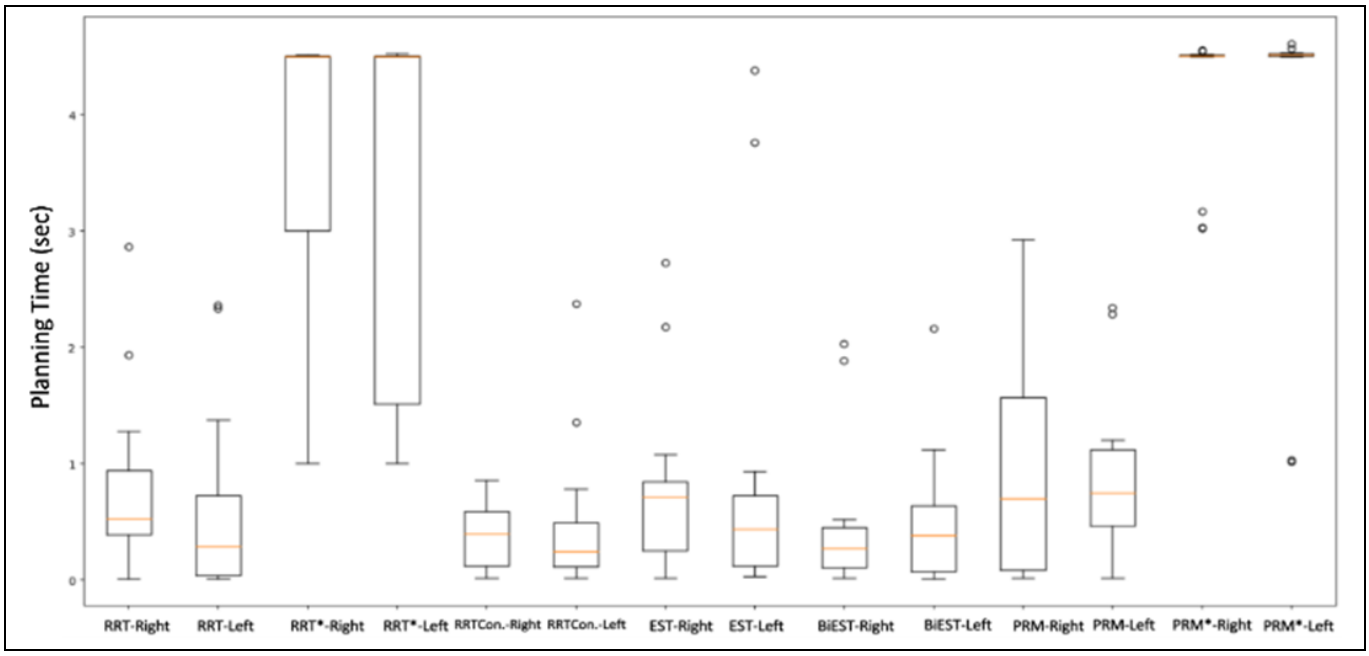

Figure 12. SolutionSetup times graph for each robot arm of trajectory planning algorithms.

The Reset Full Test is the standard task list that ROKOS applies in the real-world environment. Improvements have been made in the SRVT environment on this task list, and since SRVT MoveIt is capable of dynamic trajectory planning, reset points have been removed and a Reset-Free Full Test has been created [11]. This test can be considered as the new and improved task list of the ROKOS system to be integrated into the SRVT environment. Therefore, the determination of the algorithm to be applied to the ROKOS system in the real environment from the test data obtained should be the algorithms with the best planning and implementation time obtained from the Reset-Free Full Test results. In line with the improvements made in this context, the outputs of the best algorithm that can be used for ROKOS can be seen in the graphics in Table 10 and Figures 13 and 14.

The graph in Figure 13 shows the best and worst task completion times for each robot arm of each algorithm as a result of the Reset-Free Full Test. In Figure 14, the average task completion times for each robot arm of each algorithm in the full test result are given. According to the data in the given tables and graphs, BiEST and RRT algorithms showed a successful performance in the current task list, considering the key performance indicators. It was observed that the percentage of task completion increased by 27.9% when the points that might be unnecessary for the planner were removed from the task lists. It has also been observed that task completion times can be reduced to less than 20 minutes. Based on this, it has been concluded that the dynamic planning system applied to the ROKOS system works more effectively at less location points. In this research, more than 900 hours of testing were conducted on seven motion planning algorithms of the OMPL planner to identify the optimal planning algorithm. The BiEST algorithm was found to be the most efficient algorithm for the task. Task completion times using the BiEST algorithm were observed to reduce the task completion time of the ROKOS system by less than 25 minutes. In the study, it was determined that the average times obtained for the right ROKOS robot arm were approximately 19 minutes, and 20 minutes for the left ROKOS robot arm. By eliminating reset points from the ROKOS task list and using the optimal planning algorithm, an average task completion time gain of 20% was achieved for the ROKOS system.

This increase in efficiency resulted in significant time savings in the bus production line, leading to increased productivity.

6.3 Anomaly detection at component and system level

CamFITool Anomaly Detection feature is used for Evaluation Scenario-3 which is anomaly detection at component level. Test cases defined for VALU3S project Evaluation Scenario – 3 in Table 11.

This evaluation is conducted in two layers with CamFITool Anomaly Detection Feature. First, image is classified as normal or faulty and then faulty image is classified according to fault types. This classification gives information about possible failure at component (cable, HDD, camera etc.) in the ROKOS system. One of CNN models is given at Figure 15.

After that, 48-image prediction test was performed on the designed Binary classification model. As a result of testing with this image test library, which consists of faulty and normal images, the trained model achieved a correct prediction rate of 87.5% (42 correct out of 48). When the trained multi-class classification model was put to the prediction test with a test library of 40 images, an accurate prediction rate of 80.0% was achieved (32 correct predictions out of 40). An example prediction test can be seen in Figure 16. In Figure 17, two steps anomaly detection of Poisson faulty image example result on CamFITool FIAD plugin interface. The created anomaly detection add-on makes it feasible to avoid collecting photos that can interfere with the robot control system's part detection, as shown in Figure 3. To make sure that the part counting is done correctly, the images provided to the part control system are first run via the anomaly detection plugin. Only the images without anomalies are then sent to the system. As seen in Table 7, improper part counts brought on by inaccurate photos are therefore avoided.

Table 10. Simple setup and solution found times with improvements table.

Plan Alg.	T1	T2	T3	Avg. t	Imp. Rate
RRT-Left	0.0279	0.0309	0.829	0.788	13.29%
RRT-Right	0.0700	0.0837	0.6766	0.8448	21.7%
RRT*-Left	0.0278	0.0308	0.8298	0.7885	22.67%
RRT*-Right	0.0952	0.098	3.3718	1.5641	26.12%
RRTCon.-Left	0.0058	0.0075	0.3990	0.2824	4.63%
RRTCon- Right	0.0084	0.0088	0.5043	0.6409	24.87%
EST-Left	0.0250	0.0247	0.8178	0.7611	0.02%
EST-Right	0.0804	0.1019	0.9056	1.3322	18.13%
BiEST-Left	0.0088	0.0076	0.5275	0.6486	20.21%
BiEST-Right	0.0413	0.0566	0.5314	0.5798	24.08%
PRM-Left	0.0317	0.0405	1.0742	1.0307	12.54%
PRM-Right	0.1008	0.1049	0.9218	0.7563	27.9%
PRM*-Left	0.0375	0.0425	4.2290	0.5992	24.13%
PRM*-Right	0.1157	0.1215	4.0598	1.2331	25.9%

Table 11. VALU3S evaluation scenario-III test cases.

Test Case	Test Case Description	Expected Results
TC_14	Fault injection to robot camera system	The camera quality control system will detect which faults have been injected according to varying fault types.
TC_15	Fault injection to robot camera system	The detailed evaluation of the results that occur when the tested files are mutated, ensures that the user has information about the reliability of the tested file.

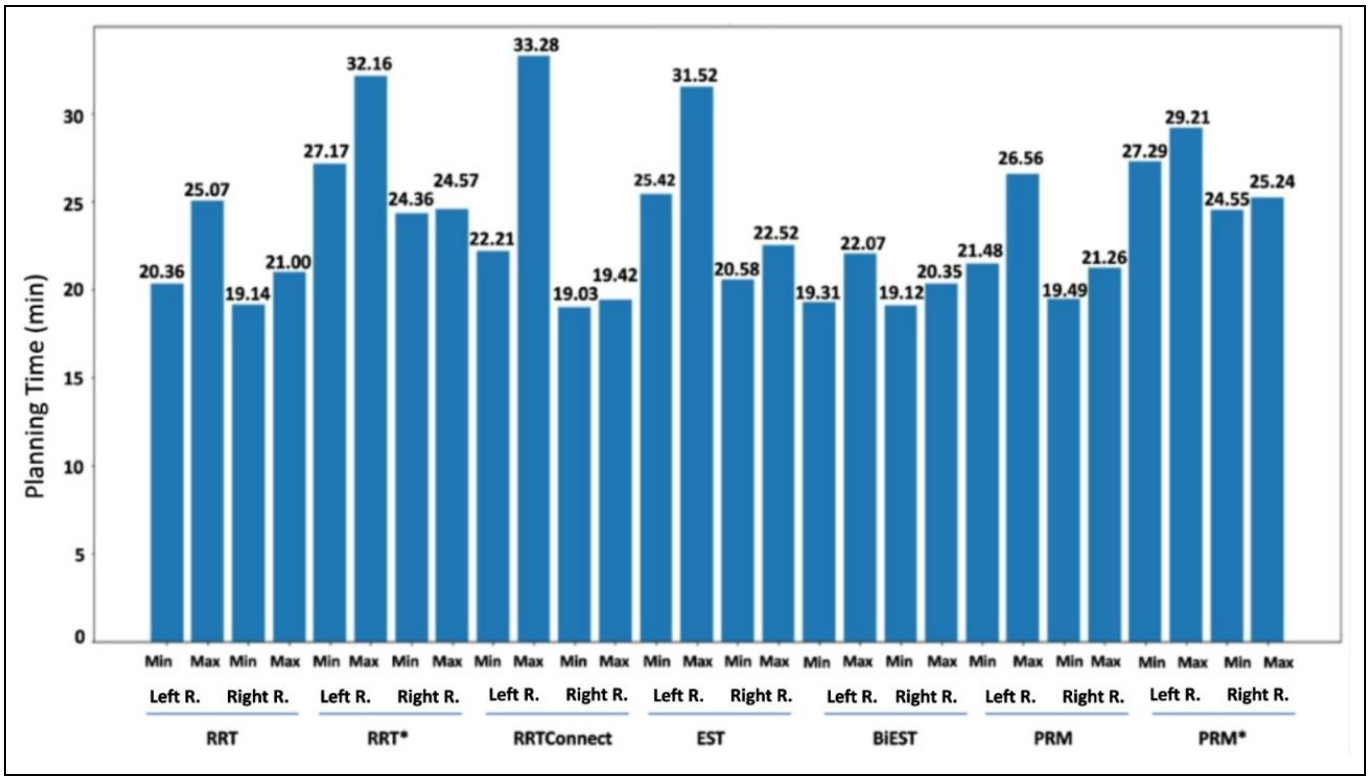

Figure 13. Full Test Results applied to current task list (without Reset) (Best and worst times of algorithms based on robot arms).

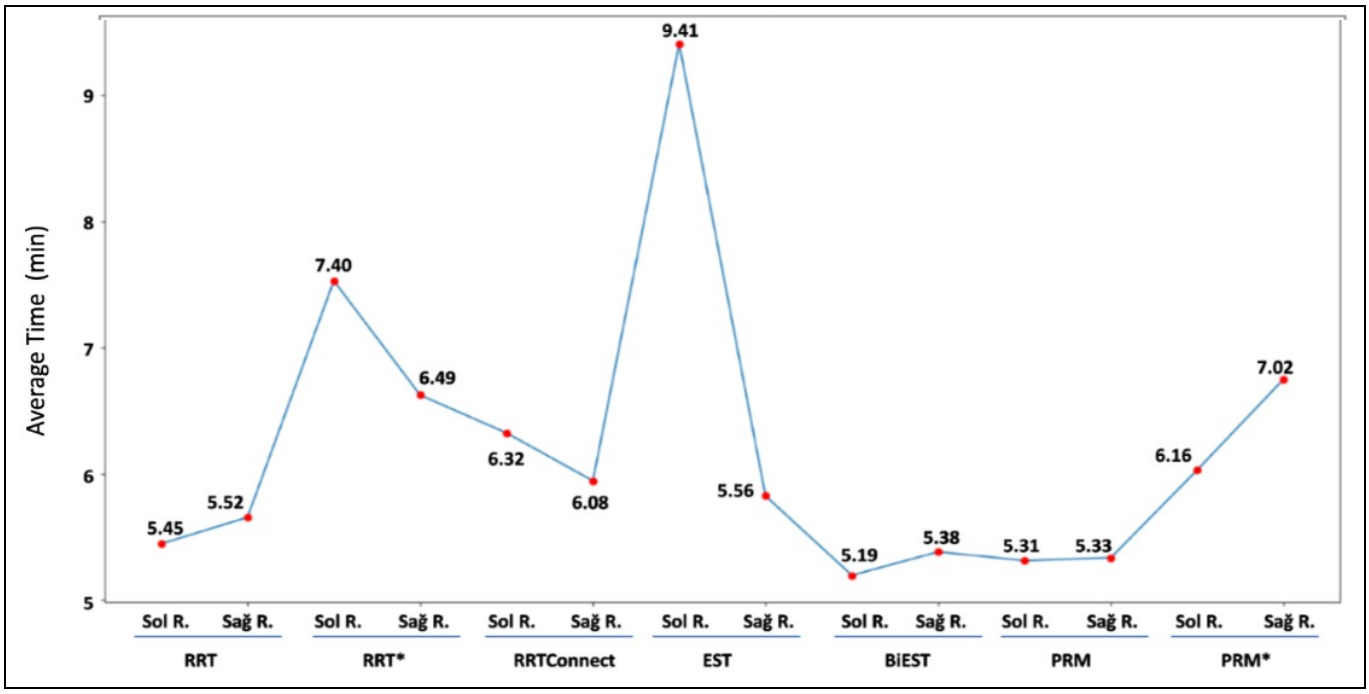

Figure 14. Full Test Results applied to current task list (without Reset) (Average times of algorithms based on robot arms).

```
model_1.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896

 conv2d_1 (Conv2D)           (None, 124, 124, 32)      9248

 max_pooling2d (MaxPooling2D  (None, 62, 62, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 60, 60, 64)        18496

 conv2d_3 (Conv2D)           (None, 58, 58, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 29, 29, 64)       0
 2D)

 flatten (Flatten)           (None, 53824)             0

 dense (Dense)               (None, 128)               6889600

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 6,955,297
Trainable params: 6,955,297
Non-trainable params: 0
_________________________________________________________________
```

Figure 15. CamFITool anomaly detection CNN model-1 summary.

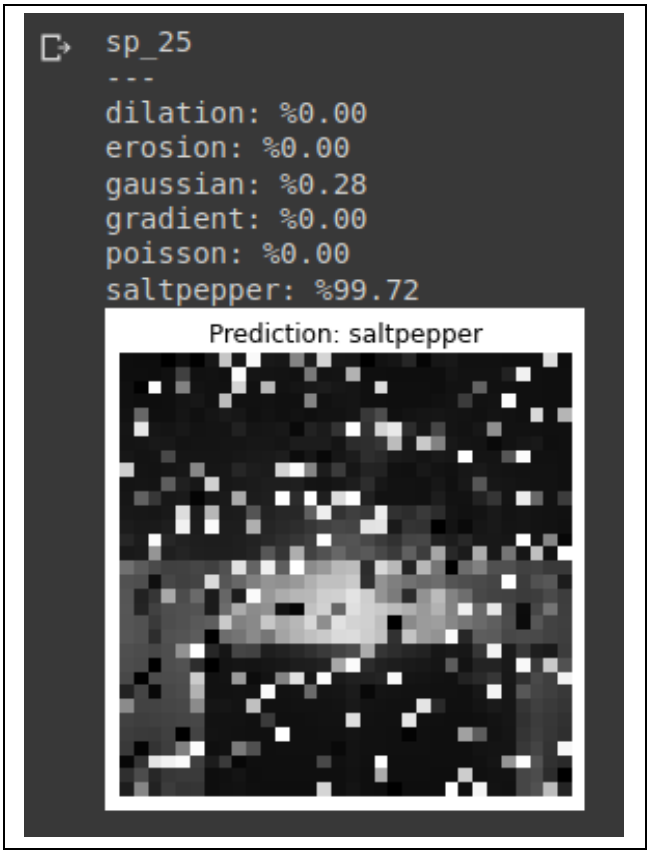

Figure 16. CamFITool Anomaly Detection Multiclass Classification model prediction test example.

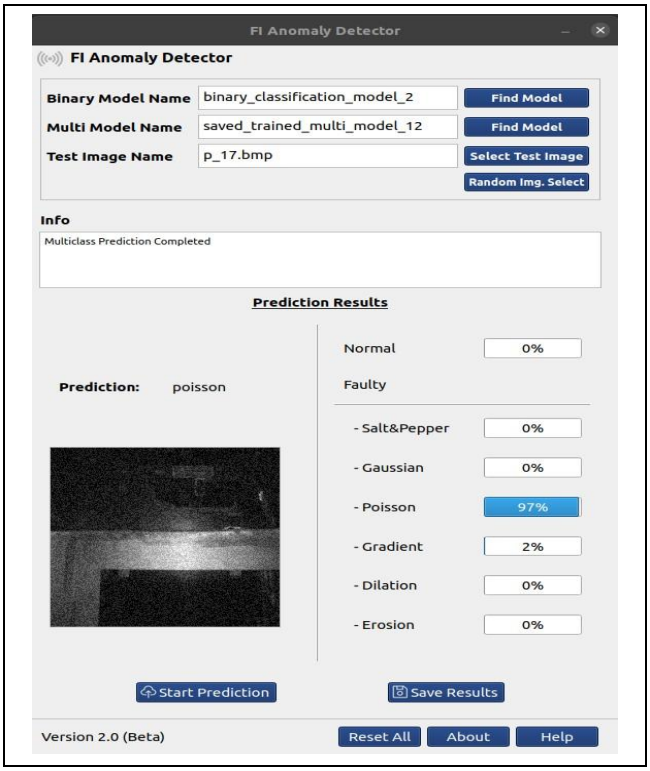

Figure 17. CamFITool Anomaly Detection Plugin usage example

7 Conclusions

In conclusion, this study has presented a novel verification tool suite for industrial robotic systems. The proposed tool suite, which includes Simulation-based Robot Verification Tool (SRVT) and Camera Fault Injection Tool (CamFITool), was specifically designed to address the challenges of testing and validating robotic systems in a manufacturing context. The tool suite was tested and evaluated within the framework of the VALU3S ECSEL H2020 project, "Autonomous Robot Inspection Cell for Vehicle Chassis Quality Control" proposed by OTOKAR.

The toolkit was evaluated by performing tests in three main evaluation scenarios (ES1, ES2, ES3), four evaluation criteria and fifteen test cases. ES1 focused on manipulation of camera sensors and testing the system's robustness against various types of faults using CamFITool. According to the EC1 (Number of malicious attacks and faults detected) criteria determined in this context, the results showed that the system's robustness value was 95.39%. ES2 focused on safety trajectory planning and testing of robot arms using SRVT, resulting in 27.9%-time savings in line with EC3 (Effort for test creation) and EC4 (Effort needed for test) criteria. In addition, robust functionality of the system was tested in accordance with EC2 (Simulation-level system robustness) criteria. Finally, ES3 focused on anomaly detection using the toolkit and achieved a fault detection rate of 80% (Table A3).

The results of our study demonstrate the effectiveness of the proposed tool suite in providing accurate and reliable validation results. The tool suite addresses the crucial aspects of testing and validating robotic systems, such as trajectory planning, sensor manipulation and anomaly detection. Additionally, it has the major advantage of being open-source and compatible with the Robot Operating System (ROS), making it easily accessible and adaptable to other use cases. This tool suite is one of the first of its kind in the ROS ecosystem and represents a significant step forward in the field of robotic system verification and validation. It allows for the performance of tests on a simulated robotic control system in a faster and more cost-effective way, while also ensuring the safety and reliability of the final product.

The open-source application related to this study can be accessed at https://github.com/Akerdogmus/camfitool and detailed tests can be performed. The application here makes it possible for users to test all the test cases included in the article.

8 Author contribution statements

Alim Kerem Erdoğmuş: conceptualization, validation, review, and final editing/proofreading and writing original draft.

Uğur Yayan: validation, review, supervision and final editing/proofreading.

9 Declarations committee approval and conflict of interest statement

"Ethics committee permission is not required for the prepared article".

"There is no conflict of interest with any person/institution in the prepared article".

10 References

[1] Aiello F, Garro A, Lemmens Y, Dutre S. "Simulation-based verification of system requirements: An integrated solution". *IEEE 14th International Conference on Networking, Sensing and Control,* Falerna, Italy, 16-18 May 2017.

[2] Bauer T, Agirre J, Furcho D, Herzner W, Hruska B, Karaca M, Pereira D, Proenca J, Schlick R, Sicher R, Smrcka A, Yayan U, Sangchoolie B. "Cross-domain modelling of verification and validation workflows in the large scale european research project valu3s". *International Conference on Embedded Computer Systems,* Taipei, Taiwan, 23-25 August 2022.

[3] Kanak A, Ergun S, Ozkan M, Cokunlu G, Yayan U, Karaca M, Arslan AT. "Verification and validation of an automated robot inspection cell for automotive body-in-white: a use case for the valu3s ecsel project". *Open Research Europe,* 1(115), 115, 2021.

[4] Chance G, Ghobrial A, Lemaignan S, Pipe T, Eder K. "An agency-directed approach to test generation for simulation-based autonomous vehicle verification". *2020 IEEE International Conference on Artificial Intelligence Testing (AITest),* Oxford, United Kingdom, 3-6 August 2020.

[5] Utting M, Pretschner A, Legeard B. "A taxonomy of model-based testing approaches". *Software testing, verification and reliability,* 22(5), 297–312, 2012.

[6] Huck TP, Ledermann C, Kroger T. "Simulation-based testing for early safety- validation of robot systems". *2020 IEEE Symposium on Product Compliance Engineering*, Portland, Oregon, USA, 16-20 November 2020.

[7] Yang Y, McLaughlin K, Littler T, Sezer S, Im EG, Yao Z, Pranggono B, Wang H. "Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in smart grid scada systems". *International conference on sustainable power generation and supply (SUPERGEN 2012),* Hangzhou, China, 8-9 September 2012.

[8] Vara JL, Bauer T, Fischer B, Karaca M, Madeira H, Matschnig M, Mazzini S, Nandi GS, Patrone F, Pereira D. "A proposal for the classification of methods for verification and validation of safety, cybersecurity, and privacy of automated systems". *International Conference on the Quality of Information and Communications Technology*, Algarve, Portugal, 8-11 September 2021.

[9] Hobbs A, Lyall B. "Human factors guidelines for remotely piloted aircraft system (rpas) remote pilot stations (rps). Technical report". San José, California, USA, 34128, 2016.

[10] Simrock S. *Control theory*. 1st ed. Bern, Switzerland, CERN, 2008.

[11] Fisher M. *An Introduction to Practical Formal Methods Using Temporal Logic.* 1st ed. John Wiley & Sons, Portland, USA, 2011.

[12] Xiao A, Bryden KM. "Virtual engineering: A vision of the next-generation product realization using virtual reality technologies". *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Salt Lake City, Utah, USA, 28-30 September 2004.

[13] Robert C, Guiochet J, Waeselynck H. "Testing a non-deterministic robot in simulation-how many repeated runs?". *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, Taichung, Taiwan, 9-11 November 2020.

[14] Cavalcanti A, Sampaio A, Miyazawa A, Ribeiro P, Filho MC, Didier A, Li W, Timmis J. "Verified simulation for robotics". *Science of Computer Programming*, 174(1), 1–37, 2019.

[15] Garoche PL. *Formal Verification of Control System Software*. 67th ed. Princeton, New Jersey, USA, Princeton University Press, 2019.

[16] Webster M, Western D, Araiza-Illan D, Dixon C, Eder K, Fisher M, Pipe AG. "A corroborative approach to verification and validation of human–robot teams". *The International Journal of Robotics Research*, 39(1), 73–99, 2020.

[17] Bogaerts B, Sels S, Vanlanduit S, Penne R. "Connecting the coppeliasim robotics simulator to virtual reality". *SoftwareX,* 11(1), 100426, 2020.

[18] Son TD, Bhave A, Auweraer HV. "Simulation-based testing framework for autonomous driving development". *2019 IEEE International Conference on Mechatronics (ICM),* Ilmenau, Germany, 18-19 March 2019.

[19] Pedersen TA, Glomsrud JA, Ruud EL, Simonsen A, Sandrib J, Eriksen BOH. "Towards simulation-based verification of autonomous navigation systems". *Safety Science*, 129(1), 104799, 2020.

[20] Hsueh MC, Tsai TK, Iyer RK. "Fault injection techniques and tools". *Computer,* 30(4), 75–82, 1997.

[21] Parasyris K, Tziantzoulis G, Antonopoulos CD, Bellas N. "Gemfi: A fault injection tool for studying the behavior of applications on unreliable substrates". *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks,* Atlanta, Georgia, USA, 23-26 June 2014.

[22] Aidemark J, Vinter J, Folkesson P, Karlsson J. "Goofi: Generic object-oriented fault injection tool". *2001 International Conference on Dependable Systems and Networks*, Gothenburg, Sweden, 1-4 July 2001.

[23] Hari SKS, Tsai T, Stephenson M, Keckler WS, Emer J. "Sassifi: An architecture-level fault injection tool for gpu application resilience evaluation". *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Santa Rosa, California, USA, 24-25 April 2017.

[24] Svenningsson R, Vinter J, Eriksson H, Torngren M. "Modifi: a model implemented fault injection tool". *International Conference on Computer Safety, Reliability, and Security*, Vienna, Austria, 14-17 September 2010.

[25] Erdogmus AK, Yayan U. "Development of simulation-based testing for automated robot cell for quality inspection of automotive body-in-white system". *TOK 2021 - Otomatik Kontrol Ulusal Kongresi*, Van, Türkiye, 2-4 September 2021.

[26] Yayan U, Erdogmus AK. "Endüstriyel robot hareket planlama algoritmaları performans karşılaştırması". *Journal of Science, Technology and Engineering Research,* 2(2), 31–45, 2022.

[27] Yayan U, Erdoğmuş AK. "Development of a fault injection tool & dataset for verification of camera-based perception in robotic systems". *Eskişehir Osmangazi Üniversitesi Mühendislik ve Mimarlık Fakültesi Dergisi,* 30(3), 328–339, 2022.

[28] Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng AY. "Ros: an open-source robot operating system". *ICRA Workshop on Open-Source Software,* Kobe, Japan, 12-17 March 2009.

[29] Chitta S, Sucan I, Cousins S. "Moveit! [ros topics]". *IEEE Robotics & Automation Magazine,* 19(1), 18-19, 2012.
[30] ROS Wiki. "SMACH article". http://wiki.ros.org/SMACH (23.02.2024).
[31] GAZEBO Website. "GAZEBO Description". http://GAZEBOsim.org/ (23.02.2024).
[32] Kuffner JJ, LaValle SM. "Rrt-connect: An efficient approach to single-query path planning". *Proceedings 2000 ICRA Millennium Conference, IEEE International Conference on Robotics and Automation, Symposia Proceedings,* San Francisco, California, USA, 24-28 April 2000.
[33] Sucan IA, Moll M, Kavraki LE. "The open motion planning library". *IEEE Robotics & Automation Magazine,* 19(4), 72–82, 2012.
[34] BTC Embedded. "When and How to Qualify Tools According to ISO 26262". https://www.btc-embedded.com/when-and-how-to-qualify-tools-according-to-iso-26262/ (23.02.2024).
[35] Osadcuks V, Pudzs M, Zujevs A, Pecka A, Ardavs A. "Clock-based time synchronization for an event-based camera dataset acquisition platform". *2020 IEEE International Conference on Robotics and Automation (ICRA),* Paris, France, 1-4 June 2020.
[36] Özkan K, Seke E, Işık Ş. "Derin öğrenmeye dayalı görünür yakın kızılötesi kamera kullanılarak buğday sınıflandırması". *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, 27(5), 618-626, 2021.

Appendix A.

Evaluation Scenario Tables for V & V Tool Suite Modules Tables containing some of the tool features mentioned in the study are included in this section. The first of these tables, Table A1, is a table showing Tool Qualifications of SRVT's modules. Table A2 shows the Tool Qualifications of CamFITool's modules. Finally, Table A3 is a table showing the evaluation scenarios, criteria and test cases provided by the V & V Tool Suite.

Appendix Table A1. SRVT modules tool qualifications.

Module Name	Module Purpose	Tool Impact	TI Desc.	Test Error Detec.	TD Desc.	TCL	TCL Desc.
SRVT Gazebo	It is a robotic simulation system. It provides a test environment.	TI2	Correct installation of model packages, accuracy of URDF files and accuracy of launch files are important.	TD2	System faults are reported as ROS errors with limited details, leading to variable solutions.	TCL2	Problems that may arise in this module have a moderate impact on the system's security.
SRVT Moveit	It is a motion planning and implementation module.	TI2	Defining the config files correctly and editing the scene is important.	TD3	System faults typically arise from improper scene creation, but are hard to detect as they require observational identification.	TCL2	Problems that may arise in this module have a moderate impact on the system's security.
SRVT Smach	It is a system control and communication module. Tasks such as task planning, application, routing rotate through this module.	TI2	It is important to plan the communication system correctly and write the Python code cleanly.	TD2	System faults can halt other modules' operations due to communication issues; they're easy to detect but may require checking all modules for the root cause.	TCL3	Problems that may arise in this module have a high impact on the security of the system.
SRVT Task Server	It is the module that allows the task list to be published with a ROS node.	TI1	It is important that the node publisher code, task list and launch file are written in the correct format.	TD1	The fault is easily detectable as an error in Python code, potentially leading the robot to incorrect positions.	TCL2	Problems that may arise in this module have a moderate impact on the system's security. However, the probability of a problem is low.
SRVT Image Server	It is the module that allows the image save function to be broadcast with a ROS node.	TI1	It is important that the node publisher code, task list and launch file are written in the correct format.	TD1	The fault, manifesting as an error in Python code, is easily detectable and may result in the robot capturing inaccurate images.	TCL1	Problems that may arise in this module have a low impact on the system's security.

Appendix Table A2. CamFITool modules tool qualifications.

Module Name	Module Purpose	Tool Impact	TI Desc.	Test Error Detec.	TD Desc.	TCL	TCL Desc.
CamFITool Offlin Fault Injection	It is the module that performs fault injection to prepared image libraries.	TI2	It is important that the library locations are entered correctly and the Python code is written cleanly.	TD2	Faults that may occur in the system may be caused by file location faults. It is easy to detect as there are try-except and fault detection functions in the code.	TCL2	Problems that may arise in this module have a moderate impact on the system's security.
CamFITool Realtime Fault Injection	It is the module that performs fault injection to the defined ROS camera nodes.	TI2	It is important that the node names are entered correctly and that the ROS codes are arranged correctly.	TD3	Faults that may occur in the system can usually be caused by entering the wrong node name. It is easy to detect as there are try-except and fault detection functions in the code.	TCL2	Problems that may arise in this module have a moderate impact on the system's security.
Anomaly Detection Module	It is a module used to detect whether an image uploaded by the user is incorrect/faulty.	TI2	It is important to define the models (CNN-trained) correctly.	TD3	Faults that may occur in the system may have occurred as a result of incorrect model definition. It is easy to detect as there are try-except and fault detection functions in the code.	TCL2	Problems that may arise in this module have a moderate impact on the system's security.

Table A3. Requirements fulfilled by V&V tool suite.

Evaluation Scenarios	Related ECs	Related TCs	Outputs
ES1	EC1	TC-1, TC-2, TC-3, TC-4	System Robustness = 95.39%
ES2	EC2, EC3, EC4	TC-5, TC-6, TC-7, TC-8, TC-9, TC-10, TC-11, TC-12, TC-13	Effort Gain = 27.9%
ES3	EC1	TC-14, TC-15	Anomaly Detection/Accurate Prediction Rate = 80.0%