

DEVELOPMENT OF A FAULT INJECTION TOOL & DATASET FOR VERIFICATION OF CAMERA BASED PERCEPTION IN ROBOTIC SYSTEMS

Uğur YAYAN^{1*}, Alim Kerem ERDOĞMUŞ²

¹ Software Engineering Department, Eskisehir Osmangazi University, Eskisehir, Turkey, ORCID No : <http://orcid.org/0000-0003-1394-5209>

² Research and Development Department, Inovasyon Muhendislik Ltd.Şti., Eskisehir, Turkey, ORCID No : <http://orcid.org/0000-0001-5111-5965>

Keywords	Abstract
Robotics Verification Fault Injection Image Dataset Camera-based Perception	Nowadays, camera-based perception is most popular topic in robotic systems. Verification of camera-based perception systems are crucial and difficult with current tools and methods. This study proposes Camera Fault Injection Tool (CamFITool), which enables different kind of fault injection methods to RGB and TOF cameras in order to perform verification and validation activities on robotic systems. Besides, Fault Injected Image Database which is created by CamFITool is introduced. In addition, the study guides to readers to create new datasets by injecting faults into existing image libraries or camera streams with CamFITool. CamFITool, an open source camera fault injection tool, has been proposed as a critical tool for assessing the safety and security of fault tolerant systems. Also, a fault injected image dataset created by CamFITool for verification of camera-based perception studies in robotic systems is given.

ROBOTİK SİSTEMLERDE KAMERA TABANLI ALGININ DOĞRULANMASI İÇİN HATA ENJEKSİYON ARACI VE VERİ KÜMESİNİN GELİŞTİRİLMESİ

Anahtar Kelimeler	Öz
Robotik Doğrulama Hata Enjeksiyonu Veri Kümesi Kamera Tabanlı Algi	Günümüzde robotik sistemlerde kamera tabanlı algılama en popüler konulardan biridir. Mevcut araç ve yöntemlerle kamera tabanlı algılama sistemlerinin doğrulanması da çok önemli ve zordur. Bu çalışma, robotik sistemlerde doğrulama ve doğrulama faaliyetlerini gerçekleştirmek için RGB ve TOF kameralara farklı türlerde hata enjeksiyon yöntemleri sağlayan Kamera Hatası Enjeksiyon Aracını (CamFITool) önermektedir. Ayrıca CamFITool tarafından oluşturulan hata enjekte edilmiş resim veri kümesi tanıtılmaktadır. Buna ek olarak çalışma, CamFITool ile mevcut görüntü kitaplıklarına veya kamera akışlarına hatalar enjekte ederek yeni veri kümeleri oluşturmak için okuyuculara rehberlik etmektedir. Sonuç olarak, hataya dayanıklı sistemlerin emniyet ve güvenliğini değerlendirmek için kritik bir araç olan açık kaynaklı bir hata enjeksiyon aracı olan CamFITool önerilmiştir. Ayrıca robotik sistemlerde kamera tabanlı algılama çalışmalarının doğrulanması için CamFITool tarafından oluşturulan hata enjekte edilmiş görüntü veri kümesi verilmiştir.
Araştırma Makalesi	Research Article
Başvuru Tarihi : 07.01.2022	Submission Date : 07.01.2022
Kabul Tarihi : 29.06.2022	Accepted Date : 29.06.2022

1. Introduction

The robotics industry has evolved over the years and has become a growing market share. According to the

2021 report of World Robotics (IFR, 2021), it was stated that the use of robots in factories around the world increased by 10 percent compared to the previous year and reached 3 million. It is thought that this rate will

* Corresponding Author; e-mail : ugur.yayan@ogu.edu.tr



Bu eser, Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) hükümlerine göre açık erişimli bir makaledir.

This is an open access article under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>).

reach 15 percent in the 2022 and the robotics industry will become widespread with an increasing momentum. Considering the sectoral distribution of robots sold, it can be seen that the use of robots in heavy and medium industry is at a considerable point (see Figure 1).

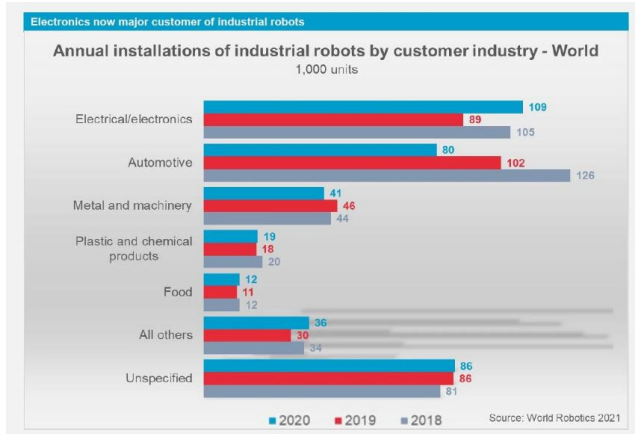


Figure 1. Annual installations of industrial robots by customer industry (World Robotics, 2021)

The widespread use of robotic systems in the industry also makes it possible for errors that may arise in robotic systems to have critical consequences. Since the importance of the software has become much more critical in the robotic systems and the anomalies that may occur in the systems are mostly possible to be realized by the software, thus it is crucial to test the system with fault injection into the system. Sensors used in robotic systems and data transfer between robots must be carried out safely. The rapid progress of robotic communication technologies has led to an increase in the data used in this field. The safely use and processing of increasing data causes situations that force designers and users to take instantaneous decisions faster. The basis of its safely use is to quickly find and intervene the anomalies that develop during the transfer. For this determination, it is critical to ensure validation and verification of systems. Implementation of an anomaly detection system based on artificial intelligence can be a solution to this verification (Kendall, Grimes and Cipolla, 2015; Osadcuks, Pudzs, Zujevs, Pecka and Ardavs, 2020; Park and Mu Lee, 2017).

Studies related to this study are given below. In Chapter 2, information is given about the literature that the robotic software (SRVT) studied. In the Chapter 3, details about SRVT, the environment and methods used by SRVT and CamFITool are given. Conclusion and future work are in Chapter 4.

2. Literature

Fault injection is an important methodology for assessing the reliability of software and associated system. Researchers, engineers, etc., who are familiar

with software, develop many new methods that can be applied both in hardware and in software to injections that may cause faults in software into the relevant systems. The situations seen among these hardware and software methods under development are as follows:

- Fault injection zones accessible to software developers and researchers,
- The cost of the injection made,
- The level of corruption caused by the fault created in the system, etc.

are factors. With fault injection into the hardware, faults can be created on chip pins and internal components such as circuits and registers that cannot be addressed by software. On the other hand, in fault injection into the software, it is possible to produce a direct change at the level of the general state of the software. Given these situations, it is possible to use hardware methods to evaluate low-level fault detection and masking mechanisms, and software methods to test higher-level mechanisms (Hsueh, Tsai and Iyer, 1997).

Studies have been carried out on many software and interfaces to create this type of fault injection. GemFI (Parasyris, Tziantzoulis, Antonopoulos and Bellas, 2014) by Parasyris and his team, GOOFI (Aidemark, Vinter, Folkesson and Karlsson, 2001) by Aidemark and his team, SASSIFI (Hari, Tsai, Stephenson, Keckler and Emer, 2017) by Hari and his team, and MODIFI (Svenningsson, Vinter, Eriksson and Törngren, 2010) by Svenningsson and his team are just a few of the important fault injection tool studies in the literature. All these studies are studies that enable fault injection for various software, simulation or hardware systems, thus enabling the testing of fault tolerance and weaknesses of the systems. Among the aforementioned sample tools, only the GemFI tool has image fault injection. Since this injection is not one of the main purposes of the study, it can be said that CamFITool's purpose of injecting fault into the image is more specific.

An anomaly in a system refers to the occurrence of the expected response, events, or other elements in a dataset that cannot usually be detected by a human expert. Such anomalies are usually caused by structural errors in the system, and these errors can turn into critical problems for the system ("Anomaly Detection, A Key Task for AI and Machine Learning, Explained", 2019). For this type of artificial intelligence systems, data sets consisting of data that are revealed as a result of the correct and incorrect operation of the mechanism that occurs anomaly can be used.

Anomalies can be seen in many different areas. For example, it is an anomaly to see plants with weak or faulty mutations during the development of plants, and research has been carried out to detect them through datasets created with samples collected from these

plants (Scharr, Minervini, Fischbach and Tsaftaris, 2014). Human actions are also an area where anomaly investigation can be done. An artificial intelligence is supported by the data sets created to detect the differences in these actions and anomaly detection can be made (Rezazadegan, Shirazi, Upcroft and Milford, 2017). Examples like this and researches show the importance of creating appropriate resources for anomaly detection using data sets.

Datasets provide accessible comparisons to improve algorithms and test new techniques. Recent developments in the fields of artificial intelligence and deep learning have accelerated with the use of data sets that are being developed. Today, data sets contain more pictures with the proliferation of researches and their use in different fields is becoming more common day by day.

Deep behavioral learning is one of the research areas where the use of data sets has an important place. In this area, the characteristics of the same people are determined and learned in different data sets (Su, Zhang, Xing, Gao and Tian, 2016), the use of object recognition features and typical movements such as describing the actions of people from video recordings (Per, Kenk, Mandeljc, Kristan and Kovacic, 2012, p. 3; Russell, Torralba, Murphy and Freeman, 2008; Wu, Oreifej and Shah, 2011) and recognizing human mobility in an environment with autonomously moving robots (Rezazadegan et al., 2017).

Datasets from images collected from the Web were also created to be used for analysis and validation in scientific research (Deng et al., 2009; Everingham, Van Gool, Williams, Winn and Zisserman, 2010; Osadcuks et al., 2020; Torralba, Fergus and Freeman, 2008). Such datasets are important studies created to collect and classify many different kinds of pictures, to be a resource that can be used for other scientific research and to be published openly for the use of researchers. As an example of this situation, Torralba et al. The large data set (Torralba et al., 2008) created by collecting about 80 million images and listing them in the dictionary dataset by tagging each image can be shown.

Datasets are used in these studies as augmented reality (Leitner, Dansereau, Shirazi and Corke, 2015; Noguchi and Harada, 2020; Orchard, Jayawant, Cohen and Thakor, 2015), robotics (Ravi, Shankar, Frankel, Elgammal and Iftode, 2007), autonomous drone navigation (Padhy, Verma, Ahmad, Choudhury and Sa, 2018), forensic techniques (Gloe and Böhme, 2010), botany (Scharr et al., 2014), data classification (Nene, Nayar and Murase, 1996; Xiao, Hays, Ehinger, Oliva and Torralba, 2010), traffic control (Fregin, Muller, Krebel and Dietmayer, 2018), it has also been used in scientific studies such as user location detection (*GAZEBO website*, 2021). In addition, it is seen that more complex

data sets are created and presented to scientists with the active use of data sets for issues such as editing and correction of distorted images (Park and Mu Lee, 2017) and the progress of studies in these areas.

Camera Fault Injection Tool (CamFITool) is an open-source tool to make necessary fault injections to robot cameras in order to take action before such robotic system faults occur. By injecting these faults into robot cameras, it is aimed to create a dataset that can be used for anomaly detection that may occur in robotic systems. For this purpose, fault injected image datasets created from images obtained by fault injections using CamFITool. With CamFITool, such datasets can be created, as well as injecting real-time faults into the camera stream and observing the system's response to it.

In this study, offline and realtime fault injections were made using CamFITool and the details of 10000-image dataset, which were created by injecting fault into the cameras in the robotic environment described in the third section, are explained. It also details how CamFITool was developed, in which environments it was tested, and a dataset generated by this software that provides verification and validation on robot cameras. Python-based CamFITool has been tested on Simulation Based Robot Verification Tool (SRVT), a robotic system running on the ROS Noetic system (Yayan and Erdoğan, 2021). These tests were performed on the robot cameras of the ROKOS system (Yayan & Erdoğan, 2021) simulated in SRVT. Additionally, it has been open sourced to make up for the lack of such a fault injection testing tool in the ROS environment.

3. Methods

In this section, the methods and application area of the study are explained. In addition, in this study, article research and publication ethics were complied with.

3.1. Experimental Environment

ROKOS (OTOKAR Robot Control System) shortens the quality control period of the product with its innovative visual inspection techniques and makes the presence-absence control of the parts that make up the bus body-in-white more sensitive. The basis of this work is to reveal a production system that performs better in terms of fault tolerance to achieve better quality control (Figure 2). With its cartesian robot and camera sensor system, ROKOS can fully automatically control the presence of 2500-3000 body parts that make up a bus body-in-white.

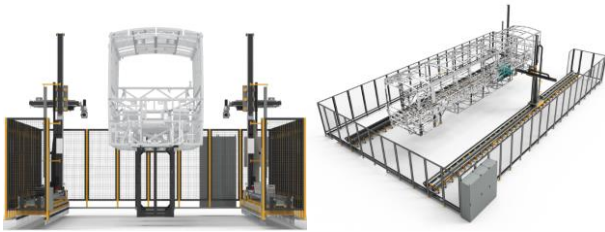


Figure 2. Robot inspection system for quality control (ROKOS)

Thanks to the Simulation-based Robot Verification Testing (SRVT) tool, the ROKOS system has been realistically transferred to the Gazebo (Figure 4) simulation environment. In the simulation environment, ROKOS can perform its tasks as in the real environment and can take images from the bus body-in-white transferred to the simulation environment in the same way (Figure 3). (Yayan and Erdoğan, 2021).

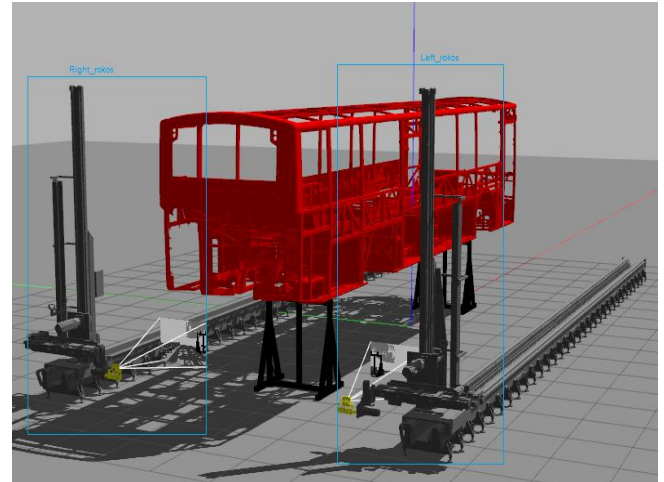


Figure 4. ROKOS system and bus body-in-white modelled on the GAZEBO simulation environment

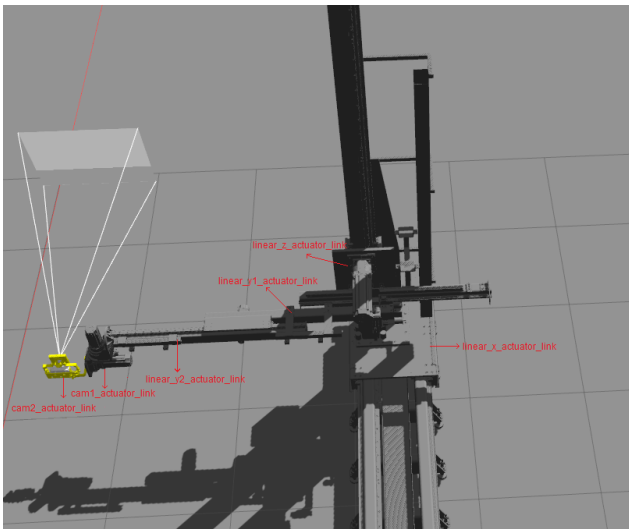


Figure 3. Image of one of the ROKOS robot arms in the SRVT simulation environment

ROKOS is converted into the SRVT environment, designed to run on versions of ROS Noetic (Quigley et al., 2009) and GAZEBO 11 (GAZEBO website., 2021) (Figure 4). Moveit (Chitta, Sucas and Cousins, 2012) software is used as the planner in SRVT.

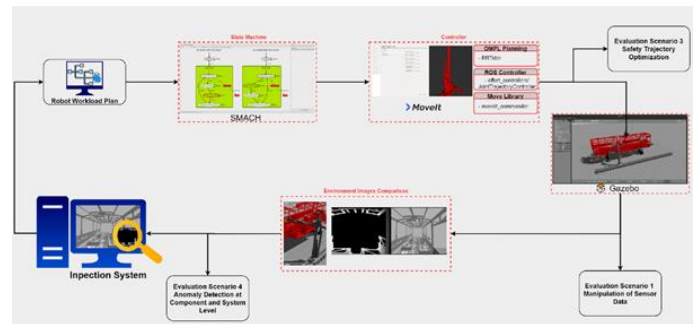


Figure 5. SRVT simulation architecture

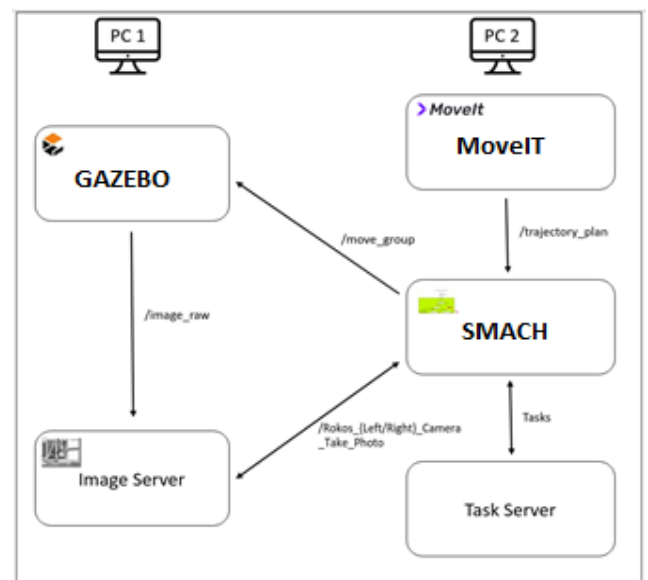


Figure 6. SRVT system architecture

The main architecture of the SRVT consists of five parts (Figure 6). These sections are;

1) Gazebo

The ROKOS robot arms and bus body-in-white are remodeled in the simulation environment of Gazebo 11. It is used as the simulation engine of SRVT.

2) Image Server

The ROS node, which enables ROKOS robot arms to take camera images from the bus body-in-white, is a system called Image Server. When this ROS node receives the "take photo" command by the ROS SMACH node, it records the photo in a folder with a special naming format defined for it. This node is manipulated at the execution of CamFITool's fault injections. In Figure 7, image examples from SRVT are given.

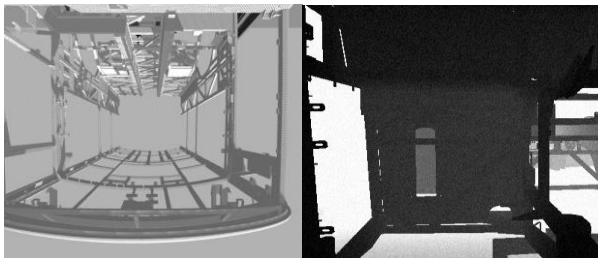


Figure 7. RGB and TOF image examples

3) Moveit

It is the planning element of the SRVT system which contains trajectory planners for robot arms.

4) Task Server

The Task Server node determines the coordinates which are visited and photo taken and sent it to the SMACH node.

5) SMACH

The SMACH node is a finite state machine for controlling the behaviors of SRVT.

3.2. Development of Camera Fault Injection Tool

This study proposes an open-source Camera Fault Injection Tool (CamFITool), which enables state-of-art fault injection methods to RGB and TOF cameras in order to perform verification and validation activities on robotic systems. This fault injection tool is written in Python and Qt5 for interface. The CamFITool is also ROS Noetic compatible (Figure 8).

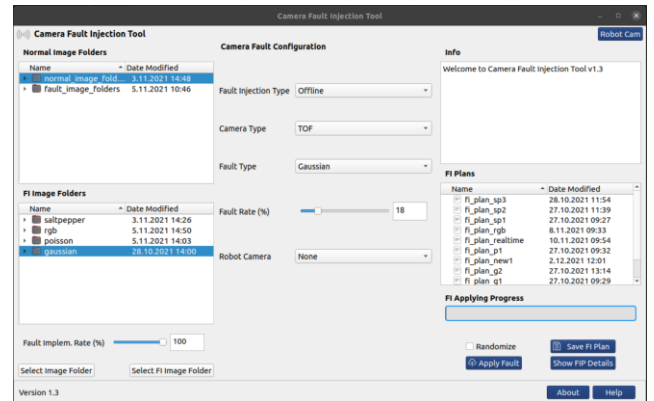


Figure 8. CamFITool main screen

CamFITool makes it possible to inject faults into camera stream or into images folder. In addition, datasets that can be used for artificial intelligence based anomaly detection studies. With CamFITool, an interface has been created that can be used in applications such as testing, verification, validation activities.

3.2.1. CamFITool Interface

CamFITool has an interface that could be apply faults in two different types (Offline and Realtime). When switching to these types, the interface changes accordingly. CamFITool, works in a single page makes it simple to use and easy to understand.

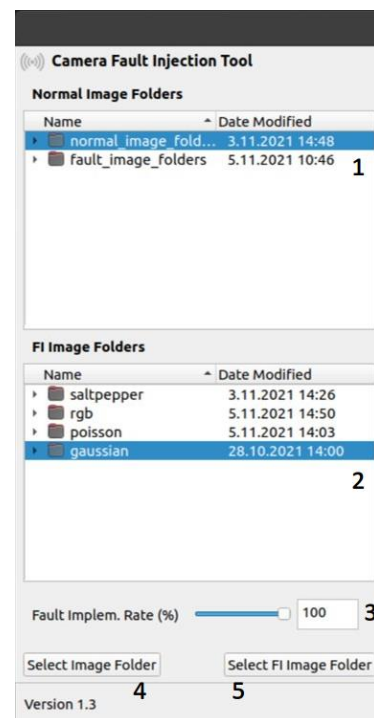


Figure 9. File selection screen of CamFITool interface

Figure 9 shows the tabs on the left side of the CamFITool interface where the folders to be processed. In Figure 9-1, The Image Folders section could be seen for image folder selection. It is possible to make this selection with the button (Figure 9-4). With the "Fault Implementation Rate" section in Figure 9-3, it is possible to select what percentage of the images in the normal image folder will be injected with faults.

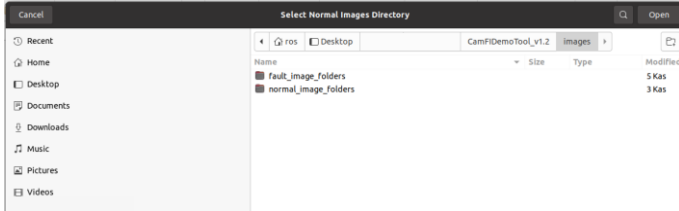


Figure 10. Normal image directory selection screen

In Figure 9-2, The FI Image Folders section is given and fault injected images could be find here. It is possible to make this selection with the button indicated by (Figure 9-5).

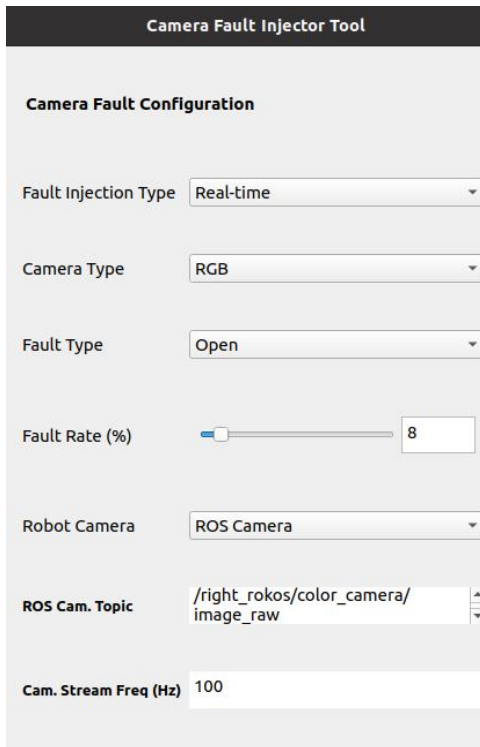


Figure 11. Camera Fault Configuration menu

Figure 11 shows the central part of the CamFITool interface, the Camera Fault Configuration menu. This menu is designed for the user to set the desired type of fault. The descriptions of the selection sections are as follows:

Fault Injection Type: In this feature, the user selects the fault application type. It has two options, Offline and Realtime. When Offline is selected, the last two options are deleted from the screen.

Camera Type: The camera type (RGB or TOF) could be selected with this feature for fault injection. For both options, different fault types are presented in the Fault Type section.

Fault Type: In this feature, the user selects the type of fault to be applied. There are Open, Close, Erosion, Dilation, Motion-blur and Gradient fault types for RGB camera, Salt&Pepper, Gaussian and Poisson fault types for TOF camera.

Fault Rate: It is the feature where the user determines the fault injection rate. The rate is directly proportional to the degradation of the image.

Robot Camera: It is the feature where the user selects the type of camera stream to be apply a fault.

ROS Camera Topic: It is the feature where the user selects the relevant topic of the ROS camera stream to be inject a fault.

Camera Stream Frequency: This feature is related with ROS camera topic and could be used in realtime fault application.

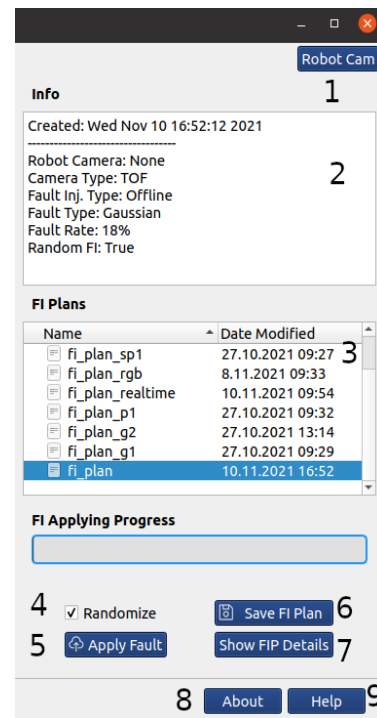


Figure 12. CamFITool interface's application and information screen

Figure 12 shows the rest of CamFITool. In this section, there is an information tab (Figure 12-2), which contains information about the processes taking place in

CamFITool, and the FI Plans tab (Figure 12-3), where the records of the fault configurations we have applied are displayed. In addition, there are buttons that provide interface control. The features of these buttons are as follows:

(Figure 12-1) Robot Cam: It is the button that allows the snapshot of the ROS camera written in the Topic section to be displayed on the screen.

(Figure 12-2) Information Tab: Information tab contains information about the processes taking place in CamFITool.

(Figure 12-3) Fault Injection Plans Tab: FI Plans tab where the records of the fault configurations we have applied are displayed.

(Figure 12-4) Randomize: This is the option that allows the offline fault application to be applied to a random number of images.

(Figure 12-5) Apply Fault: It is the button that starts the fault injection of with the fault configuration set by the user.

(Figure 12-6) Save FI Plan: This is the button that allows the user to save the fault configuration set. Saved plans can be viewed in the FI Plan tab.

(Figure 12-7) Show FIP Details: This is the button used for the user to view the details of the recorded fault plans in the Info tab. The user can select the plan who wants to see from the FI Plans tab and click this button to view the details.

(Figure 12-8) About: This is the button that displays CamFITool's tag information.

(Figure 12-9) Help: This is the button that opens the help section prepared for CamFITool.

3.3. Construction of Fault Injected Image Dataset

The faulty image dataset is an image library that contains samples of faulty images that a robot camera has recorded as a result of various faults it may encounter. Possible camera malfunctions, such as the camera recording the faulty images at that time and the system continuing to work on these faulty images, can cause serious problems in the system. CamFITool is designed to simulate such failures, collect faulty picture outputs of the systems as a result of these failures, and create faulty output source for artificial intelligence systems to be developed to detect these problems. CamFITool can inject faults into the SRVT or any other camera-based detection system, corrupting the images that the system is supposed to record with various image degradation techniques. The faulty image dataset mentioned in this study consists of normal images and degraded images recorded by the SRVT system from ROKOS cameras running on Gazebo (Figure 13).

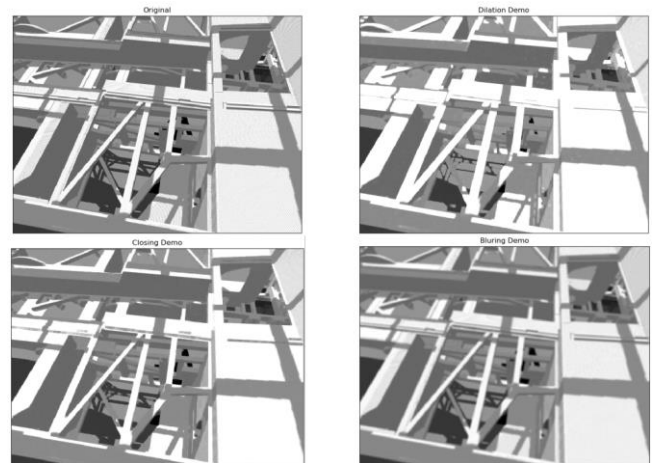


Figure 13. Camera fault injection outcome examples


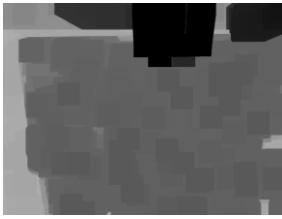
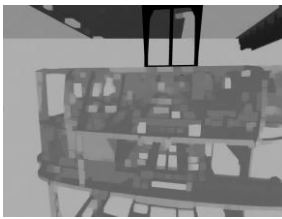
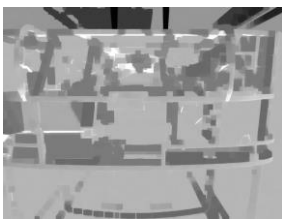
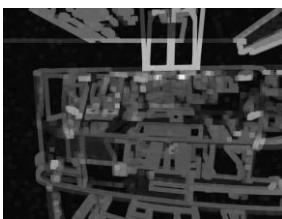

The dataset was created using two different application types to test all the fault injection features of CamFITool and to obtain different outputs. The first dataset was created from fault injected and normal images obtained by applying realtime fault injection to the cameras on the robot arms operating in the SRVT. The second dataset was created from fault injected images obtained by offline fault injection to the images taken by the SRVT. Dataset creation processes with these two fault injection application types are explained in the following sections.

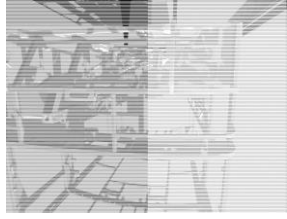
3.3.1. Realtime Fault Injected Image Dataset from SRVT Simulation Environment

In order to create this dataset, fault injection was performed by CamFITool to the RGB image nodes of two robot cameras running on the SRVT system. While injected 6 types of image faults defined in accordance with RGB cameras, the system continued to work and save pictures. The tasks were repeated until a certain number of pictures of each fault type were recorded (Erdogmus, Karaca and Yayan, 2021).

CamFITool allows us to create a dataset of 5000 normal and 5000 fault injected images (see Table 1) from the SRVT system. In addition, users can create such datasets in desired configurations and camera-based perception-based environments by using CamFITool. It has been observed that there is no such tool in ROS yet. CamFITool makes use of a software fault injection system that works directly in compatible with ROS (CamFITool ROS Wiki, 2021).

Table 1
RGB Camera Fault Types

Fault Method	Description	Example Image
Dilation	It is used to enlarge the highlighted parts of an image (Jankowski, 2006). There are 476 of these faults injected pictures in the dataset.	
Erosion	It is used to reduce the highlighted part of an image (Jankowski, 2006). There are 637 of these faults injected images in the dataset.	
Open	It is created by first etching and then spreading an image (Acton and Mukherjee, 2000). There are 640 of these faults injected images in the dataset.	
Close	It is created by first spreading and then etching an image (Acton and Mukherjee, 2000). There are 841 of these faults injected images in the dataset.	
Gradient	It is created by subtracting the etched version from the smeared version of an image (Larnier, Fehrenbach and Masmoudi, 2012). There are 632 of these faults injected images in the dataset.	
Motion-blur	It is created by providing blur in an image (Ji and Liu, 2008). There are 687 of these faults injected images in the dataset.	

Partialloss	It is created by destroying a specified part of an image. There are three different partialloss types: horizontal, vertical and odd-even. There are 1087 of these faults injected pictures in the dataset.	
--------------------	--	---

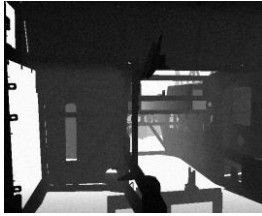
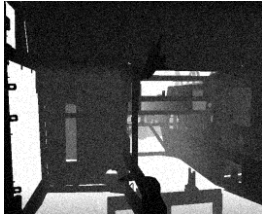
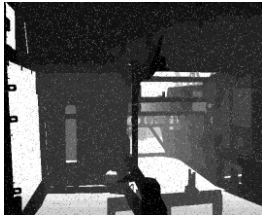
The dataset was created by extracting the fault injected and normal ones from the pictures collected while the SRVT system was running. All images have been tagged with standard SRVT tags. This is because the system also saves fault injected pictures as if they were normal pictures for inspection processes. The created dataset could be used to train the artificial intelligence that will detect the fault injected pictures of the system and to create this anomaly detection system.

3.3.2. Offline Fault Injected Image Dataset from SRVT Created Images

To use CamFITool's offline fault injection feature, an image dataset of images recorded by TOF cameras in the SRVT was used. TOF camera images captured by the SRVT were injected by CamFITool with faults of three types (see Table 2). As in the studies described above, different types of fault-injected images can be obtained, and datasets can be created from these images.

The listed image fault types can be examined in two different classes (see Table 3) as virtual environment faults and real environment faults. Virtual environment faults like Dilation/Erosion, Open/Close, Gradient, InjectionPayload are artificial image errors that can only be revealed in the simulation environment and through software (OpenCV Python libraries, etc.). These faults can be counted as virtual environment faults. Such faults are not seen in real environment/hardware with non-software methods.

Table 2
TOF Camera Fault Types

Fault Method	Description	Example Image
Gaussian	Gaussian noise is statistical noise with a probability density function equal to the normal distribution, also known as the Gaussian distribution. In other words, the values that the noise can take are Gaussian distribution (Figure 17 (Barbu, 2013).	
Poisson	Shot noise or Poisson noise is a type of noise that can be modeled with a Poisson process. In electronics, gravitational noise is due to the discrete nature of electric charge. Shot noise also occurs in photon counting in optical devices where the shot noise is related to the particle nature of light (Figure 18) (Blanter and Büttiker, 2000; Schottky, 2018).	
Salt&Pepper	Salt and pepper noise is a type of noise that can sometimes be observed. Also known as impulse noise. This noise can be caused by sharp and sudden disturbances in the image signal. It presents itself as sparsely occurring white and black pixels (Figure 19) (Rosin and Collomosse, 2012).	

Real environment faults like (Motion-blur, Partialloss, Freeze/Slow, Random, Gaussian, Poisson, Salt&Pepper) (or camera / hardware faults), on the other hand, are image faults that may occur in real environment cameras, not only through software, but also due to problems that may occur in the camera hardware. These faults can be counted as real environment faults. For example, a Poisson noise may be generated in electronic circuits by random fluctuations of electric current in a DC current, which is caused by the fact that the current is actually the flow of discrete charges (electrons). This fault may cause related corruption in camera hardware.

Table 3
General Camera Fault Types/Methods

Fault Types	Camera Types	Virtual/Real Environment Fault	Dataset
Dilation	RGB	Virtual	X
Erosion	RGB	Virtual	X
Open	RGB	Virtual	X
Close	RGB	Virtual	X
Gradient	RGB	Virtual	X
Motion-blur	RGB/TOF	Real	X**
Partialloss	RGB/TOF	Real	X**
Freeze	RGB	Real	
Slow	RGB	Real	
InjectionPayload	RGB/TOF	Virtual	
Random	RGB/TOF	Real	
Gaussian	TOF	Real	X*
Poisson	TOF	Real	X*
Salt&Pepper	TOF	Real	X*

*** In marked these ones, only RGB samples are available in the dataset.

** In marked these ones will be added to the dataset.

4. Conclusion and Future Works

In this study, CamFITool, an open-source fault injection tool, which is a critical tool for assessing of fault tolerant systems' safety and security, is proposed. In addition, CamFITool can apply faults not only to the Realtime camera stream, but also to the offline recorded image folders. As a result, a fault injected image dataset for the camera-based perception studies in robotic systems, and to help determine the fault tolerances of the systems is published. In addition, users can create such datasets in desired configurations and camera-based perception-based environments by using CamFITool. It has been observed that there is no such tool in ROS or robotics community yet.

In the future works, it is planned to develop the CamFITool interface and continue to add new features. The interface is completely open source and developments can be followed in the current version of the interface, Github repository (CamFITool Github repository, 2021). It is also planned to work on a deep learning-based anomaly detection system that will detect fault injected pictures in created datasets by CamFITool.

Acknowledgment

This study is supported by TÜBİTAK Project under grant number 120N803 which conducted by the İnovasyon Mühendislik.

Also, the research leading to this paper has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey. The views expressed in this document are the sole responsibility of the authors and do not necessarily reject the views or position of the European Commission.

Conflicts of Interest

They declared that there is no conflict of interest between the authors and their respective institutions.

Contributions of the Authors

In this work, Uğur YAYAN created the article concept, conducted the research, tested it, edited it, managed the project, and wrote the article in this work. The article's concept, research, coding, testing, development of SRVT and CamFITool, and writing have been completed by Alim Kerem ERDOĞMUŞ.

References

- Acton, S. T., & Mukherjee, D. P. (2000). Scale space classification using area morphology. *IEEE Transactions on Image Processing*, 9(4), 623-635. <https://doi.org/10.1109/83.841939>
- Aidemark, J., Vinter, J., Folkesson, P., & Karlsson, J. (2001). GOOFI: Generic object-oriented fault injection tool. *2001 International Conference on Dependable Systems and Networks*, 83-88. <https://doi.org/10.1109/DSN.2001.941394>
- Anomaly Detection, A Key Task for AI and Machine Learning, Explained. (2019). *KDnuggets*. Available 06 January 2022, <https://www.kdnuggets.com/anomaly-detection-a-key-task-for-ai-and-machine-learning-explained.html/>
- Barbu, T. (2013). Variational Image Denoising Approach with Diffusion Porous Media Flow. *Abstract and Applied Analysis*, 2013, e856876. <https://doi.org/10.1155/2013/856876>
- Blanter, Ya. M., & Büttiker, M. (2000). Shot noise in mesoscopic conductors. *Physics Reports*, 336(1), 1-166. [https://doi.org/10.1016/S0370-1573\(99\)00123-4](https://doi.org/10.1016/S0370-1573(99)00123-4)
- CamFITool Github repository. (2021). *Camera Fault Injection Tool* [Python]. Inovasyon Muhendislik. <https://github.com/inomuh/camfitool> (Original work published 2021)
- CamFITool ROS Wiki. (2021). *camfitool—ROS Wiki* [Wiki]. CamFITool ROS Wiki Page. <http://wiki.ros.org/camfitool/>
- Chitta, S., Sucan, I., & Cousins, S. (2012). MoveIt! [ROS Topics]. *IEEE Robotics & Automation Magazine*, 19(1), 18-19. <https://doi.org/10.1109/MRA.2011.2181749>
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248-255. <https://doi.org/10.1109/CVPR.2009.5206848>
- Erdogmus, A. K., Karaca, M., & Yayan, A. P. D. U. (2021). Manipulation of Camera Sensor Data via Fault Injection for Anomaly Detection Studies in Verification and Validation Activities For AI. *arXiv:2108.13803* [cs]. <http://arxiv.org/abs/2108.13803>
- Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303-338. <https://doi.org/10.1007/s11263-009-0275-4>
- Fregin, A., Muller, J., Krebel, U., & Dietmayer, K. (2018). The DriveU Traffic Light Dataset: Introduction and Comparison with Existing Datasets. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 3376-3383. <https://doi.org/10.1109/ICRA.2018.8460737>
- GAZEBO website. (2021). <http://GAZEBOsim.org/>
- Gloe, T., & Böhme, R. (2010). The "Dresden Image Database" for benchmarking digital image forensics. *Proceedings of the 2010 ACM Symposium on Applied Computing*, 1584-1590. <https://doi.org/10.1145/1774088.1774427>
- Hari, S. K. S., Tsai, T., Stephenson, M., Keckler, S. W., & Emer, J. (2017). SASSIFI: An architecture-level fault injection tool for GPU application resilience evaluation. *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 249-258. <https://doi.org/10.1109/ISPASS.2017.7975296>
- Hsueh, M.-C., Tsai, T. K., & Iyer, R. K. (1997). Fault injection techniques and tools. *Computer*, 30(4), 75-82. <https://doi.org/10.1109/2.585157>

- IFR. (2021). *IFR presents World Robotics 2021 reports*. IFR International Federation of Robotics. Available 06 January 2022, <https://ifr.org/ifr-press-releases/news/robot-sales-rise-again>
- Jankowski, M. (2006). *Erosion, dilation and related operators*. 10.
- Ji, H., & Liu, C. (2008). Motion blur identification from image gradients. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8. <https://doi.org/10.1109/CVPR.2008.4587537>
- Kendall, A., Grimes, M., & Cipolla, R. (2015). *PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization*. 2938-2946. https://openaccess.thecvf.com/content_iccv_2015/html/Kendall_PoseNet_A_Convolutional_ICCV_2015_paper.html
- Larnier, S., Fehrenbach, J., & Masmoudi, M. (2012). The Topological Gradient Method: From Optimal Design to Image Processing. *Milan Journal of Mathematics*, 80(2), 411-441. <https://doi.org/10.1007/s00032-012-0196-5>
- Leitner, J., Dansereau, D., Shirazi, S., & Corke, P. (2015). The need for dynamic and active datasets. *CVPR Workshop on The Future of Datasets in Computer Vision*, 1-1. <https://eprints.qut.edu.au/105801/>
- Nene, S. A., Nayar, S. K., & Murase, H. (1996). *Columbia Object Image Library (COIL-100)*. 6.
- Noguchi, A., & Harada, T. (2020). RGBD-GAN: Unsupervised 3D Representation Learning From Natural Image Datasets via RGBD Image Synthesis. *arXiv:1909.12573* [cs]. <http://arxiv.org/abs/1909.12573>
- Orchard, G., Jayawant, A., Cohen, G. K., & Thakor, N. (2015). Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades. *Frontiers in Neuroscience*, 9, 437. <https://doi.org/10.3389/fnins.2015.00437>
- Osadcuks, V., Pudzs, M., Zujevs, A., Pecka, A., & Ardavs, A. (2020). Clock-based time sync hronization for an event-based camera dataset acquisition platform *. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 4695-4701. <https://doi.org/10.1109/ICRA40945.2020.9197303>
- Padhy, R. P., Verma, S., Ahmad, S., Choudhury, S. K., & Sa, P. K. (2018). Deep Neural Network for Autonomous UAV Navigation in Indoor Corridor Environments. *Procedia Computer Science*, 133, 643-650. <https://doi.org/10.1016/j.procs.2018.07.099>
- Parasyris, K., Tziantzoulis, G., Antonopoulos, C. D., & Bellas, N. (2014). 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*, 622-629. <https://doi.org/10.1109/DSN.2014.96>
- Park, H., & Mu Lee, K. (2017). *Joint Estimation of Camera Pose, Depth, Deblurring, and Super-Resolution From a Blurred Image Sequence*. 4613-4621. https://openaccess.thecvf.com/content_iccv_2017/html/Park_Joint_Estimation_of_ICCV_2017_paper.html
- Per, J., Kenk, V. S., Mandeljic, R., Kristan, M., & Kovacic, S. (2012). Dana36: A Multi-camera Image Dataset for Object Identification in Surveillance Scenarios. *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance*, 64-69. <https://doi.org/10.1109/AVSS.2012.33>
- Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). *ROS: an open-source Robot Operating System*. 6.
- Ravi, N., Shankar, P., Frankel, A., Elgammal, A., & Iftode, L. (2007). Indoor Localization Using Camera Phones. *Seventh IEEE Workshop on Mobile Computing Systems Applications (WMCSA'06 Supplement), Supplement*, 1-7. <https://doi.org/10.1109/WMCSA.2006.4625206>
- Rezazadegan, F., Shirazi, S., Upcroft, B., & Milford, M. (2017). Action recognition: From static datasets to moving robots. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 3185-3191. <https://doi.org/10.1109/ICRA.2017.7989361>
- Rosin, P., & Collomosse, J. (2012). *Image and Video-Based Artistic Stylisation*. Springer Science & Business Media.
- Russell, B. C., Torralba, A., Murphy, K. P., & Freeman, W. T. (2008). LabelMe: A Database and Web-Based Tool for Image Annotation. *International Journal of Computer Vision*, 77(1), 157-173. <https://doi.org/10.1007/s11263-007-0090-8>
- Scharr, H., Minervini, M., Fischbach, A., & Tsaftaris, S. A. (2014). *Annotated Image Datasets of Rosette Plants*. 17.
- Schottky, W. (2018). On spontaneous current fluctuations in various electrical conductors. *Journal of Micro/Nanolithography, MEMS, and MOEMS*, 17(4), 041001. <https://doi.org/10.1117/1.JMM.17.4.041001>
- Su, C., Zhang, S., Xing, J., Gao, W., & Tian, Q. (2016). Deep Attributes Driven Multi-camera Person Re-identification. İçinde B. Leibe, J. Matas, N. Sebe, & M. Welling (Ed.), *Computer Vision – ECCV 2016* (ss. 475-491). Springer International Publishing. https://doi.org/10.1007/978-3-319-46475-6_30

- Svenningsson, R., Vinter, J., Eriksson, H., & Törngren, M. (2010). MODIFI: A MODel-Implemented Fault Injection Tool. İçinde E. Schoitsch (Ed.), *Computer Safety, Reliability, and Security* (ss. 210-222). Springer. https://doi.org/10.1007/978-3-642-15651-9_16
- Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11), 1958-1970. <https://doi.org/10.1109/TPAMI.2008.128>
- Wu, S., Oreifej, O., & Shah, M. (2011). Action recognition in videos acquired by a moving camera using motion decomposition of Lagrangian particle trajectories. *2011 International Conference on Computer Vision*, 1419-1426. <https://doi.org/10.1109/ICCV.2011.6126397>
- Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., & Torralba, A. (2010). SUN database: Large-scale scene recognition from abbey to zoo. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3485-3492. <https://doi.org/10.1109/CVPR.2010.5539970>
- Yayan, U., & Erdoğan, A. (2021). Endüstriyel Robot Hareket Planlama Algoritmaları Performans Karşılaştırması. *Journal of Scientific, Technology and Engineering Research*, 2(2), 31-45. <https://doi.org/10.53525/jster.979689>