Proceedings of the 35th European Safety and Reliability & the 33rd Society for Risk Analysis Europe Conference
Edited by Eirik Bjorheim Abrahamsen, Terje Aven, Frederic Bouder, Roger Flage, Marja Ylönen
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-94-3281-3_ESREL-SRA-E2025-P6239-cd

A Time-series Data Generation Tool for Risk Assessment of Robotic Applications

Yuliang Ma[1], Apurv Patel[1], Don Kurian[1], Julien Siebert[2], Silvia Vock[3], Andrey Morozov[1]

[1] *University of Stuttgart, Institute of Industrial Automation and Software Engineering (IAS), Germany. E-mail: {first.last}@ias.uni-stuttgart.de*

[2] *Fraunhofer Institute for Experimental Software Engineering, Germany. E-mail: {last.first}@iese.fraunhofer.de*

[3] *Federal Institute for Occupational Safety and Health (Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, BAuA), Germany. E-mail: {last.first}@baua.bund.de*

Robotic systems increasingly rely on artificial intelligence (AI) to enhance their capabilities in performing complex tasks across various domains. The development and evaluation of AI systems usually require high-quality datasets. In addition to normal datasets, faulty datasets are critical for enabling anomaly detection and failure prevention, which are essential for ensuring the safety and reliability of safety-critical robotic applications. However, faults are rare in real-world environments. Although fault injection techniques allow for the manual injection of configurable faults, deploying such methods directly in real-world settings is rather risky. As such, it is important to develop a data generation tool which is low-cost, safe, and efficient. To address this, we developed a time-series data generation tool for the risk assessment of robotic applications. In this paper, we used Robot Operating System (ROS) Quigley et al. (2009) as the middleware. This ROS-based simulation tool integrates three key modules: (1) a Gazebo-based scene generator that can configure different working scenarios (e.g., drilling and welding) by adjusting end-effectors, workpieces, and hand positions; (2) an online fault injector that can introduce faults into robotic systems with configurable parameters; and (3) a risk monitor that records faulty data and safety violations in real time by measuring the distance between hands and end-effectors. Proposed tool facilitates the generation of time-series fault data and helps identify faults that may pose risks in human-robot collaboration scenarios. Additionally, the proposed simulation tool enables fast and safe deployment for other robot-related research areas, e.g., deep learning-based anomaly detection, failure prediction, and risk assessment.

Keywords: Fault injection, execution monitoring, robotic manipulator, human-robot collaboration.

1. Introduction

Currently, AI is increasingly enhancing the ability of robotic systems in many fields Vrontis et al. (2023), and valid datasets are usually critical for deploying AI agents. Compared with the normal dataset collected when the robot operates normally, datasets with anomalies are equally important for anomaly detection and failure prevention in robotic applications. In addition, modern robots can perform various complex tasks by leveraging dense integration among multiple sensors (e.g., cameras, IMU sensors, LiDAR). This intensive integration between hardware and software increases the behavioral and structural complexity of robotic systems, which significantly raises the likelihood of faults and errors. These issues can lead to hazards and unexpected accidents, especially in human-robot collaboration (HRC) scenarios, where safety is critical. Sensor faults and errors in such environments can threaten human safety. Deep learning-based anomaly detection (DLAD) methods show promise in identifying errors and enhancing safety proactively Luo et al. (2021), Chirayil Nandakumar et al. (2024). However, the effects of sensor faults and errors are uncertain and it is essential to study the abnormal behaviors caused by these issues. For detecting unsafe robot behaviors, existing methods can generally be categorized into reactive approaches Inceoglu et al. (2021) and proactive approaches Ji et al. (2022). The availability of effective data and labels is critical for all these studies. However, faults and errors are rare in real-world scenarios, and obtaining risk labels can be both challenging and unsafe. As such, it is necessary to have a data generation tool that could generate faulty datasets

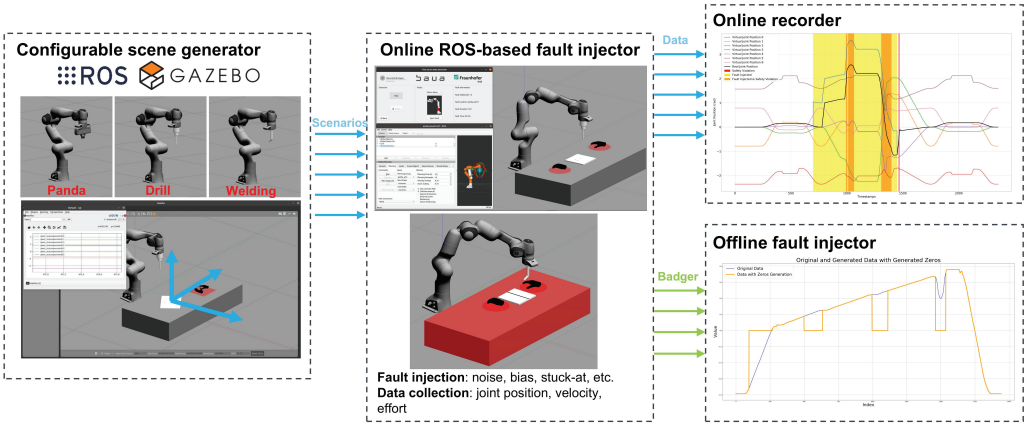


Fig. 1. Overview of the proposed tool.

in robotic applications. To address this challenge, we propose a time-series data generation tool based on the ROS-Gazebo simulator. This tool can (1) generate configurable industrial scenarios, (2) inject sensor faults into robots online, and (3) monitor and label hazards automatically. Specifically, we define configurable scenarios for practical tasks such as welding and drilling. Faults with random parameters are injected into the joint sensors of the robotic arm, and hazards are automatically labeled by monitoring the distance between the end-effector and the human hand. An overview of the proposed tool is shown in Fig. 1. The main contributions of this work are summarized as follows:

- A time-series data generation tool for manipulator joint states. This tool simulates various industrial scenarios and injects two types of signal faults into joint sensors in real time, recording unsafe behaviors caused by faults. Additionally, it includes an offline fault injection feature for post-processing. This easy-to-deploy tool supports research areas such as deep learning-based anomaly detection and risk assessment in HRC.
- The relationship between injected faults and safety violations is explored. Specifically, whether a fault leads to abnormal manipulation behaviors depends on various factors, such as fault parameters and scenarios. This insight offers new perspectives for anomaly detection studies in robotics, particularly on classifying faults that genuinely threaten human safety in HRC.
- A publicly available Fault-to-Hazard[a] dataset generated using our time-series data tool. The dataset includes faulty joint states (from sensors) and reactive joint states (from actuators). In addition, fault flag, safety violations, fault parameters, and the distance between hands and the end-effector are concluded in the dataset.

2. Related works

2.1. *Fault injection*

Fault injection is an efficient method for risk analysis, especially in many safety-critical applications. In the field of robotics, many related methods has been introduced. Our previous work introduced a ROS-based fault injection method for risk assessment. Ma et al. (2023). The case study illustrated the failure modes might happen after injecting faults in a pick-and-place task. Favier et al. (2020) introduced a framework to analyze error propagation chain for mobile robots. The proposed framework could inject faults based on ROS-Gazebo simulator. In addition, Hsiao et al. (2023) proposed a framework, MAFVI, for error analysis. This end-to-end framework could inject

[a]https://www.kaggle.com/datasets/yuliangma/fault-to-hazardts-generator

faults and provide mitigation strategy. For manipulators, Li et al. (2016) introduced a method that could automatically inject faults into the robot control module, and a feedback is provided to the operator.

2.2. *Safety in HRC*

Ensuring safety in HRC is an important and challenging task. As specifies in ISO/TS 15066[b], collaborative robots are required to at least have one or more following functions: *Safety-rated monitored stop*, *Hand guiding*, *Speed and separation monitoring (SSM)* and *Power and force limiting (PFL)*. Among these requirements, SSM-related works have be widely studied. In brief, the core of SSM method is to keep humans at a minimum distance from robots during operation. Malm et al. (2019) developed a safety system that monitors the speed and separation between humans and robots. The system could dynamically determine the minimum safety distance and avoid collisions. Kamezaki et al. (2024) introduced a new concept about dynamic collaborative workspace. This concept dynamically adjusts the workspace that the robot is restricted from entering via predicting the human's trajectory. In addition, Park et al. (2021) and Heo et al. (2019) introduced some deep learning-based collision detection methods for HRC. In general, aforementioned works rely to varying degrees on valid datasets and labels. However, directly collecting data from real-world is expensive and even dangerous when it's related to safety topic in HRC. Motivated by this, our goal is to develop a time-series data generation tool using simulators, which is convenient, efficient, and safe.

3. Method

This section provides a detailed introduction to the core functionalities of the proposed tool, including (i) **Configurable scene generator**, (ii) **Online fault injector**, (iii) **Execution monitor**, and (iv) **Offline fault injector**.

[b]Robots and Robotic Devices—Collaborative Robots, International Organization for Standardization, ISO/TS Standard 15066:2016.

3.1. *Configurable scene generator*

First, customized end-effectors for drilling and welding tasks are designed. These CAD files were incorporated into the corresponding robot description file, where they were defined as new end-effector links for the manipulator. To enable the creation of configurable working scenarios, we developed the following functions:

- *Randomize workpiece geometry/position:* The size and pose of the workpiece could be an be randomly adjusted within a predefined area on the workbench.
- *Randomize drilling points/welding line:* For different tasks, the positions of drilling points and welding lines are randomly assigned on the workpiece. These target points and lines are then automatically retrieved by the execution program.
- *Randomize hand's location:* The positions of two hands are randomly assigned to regions outside the boundaries of the workpiece. These positions are automatically recorded and used for subsequent distance calculations.

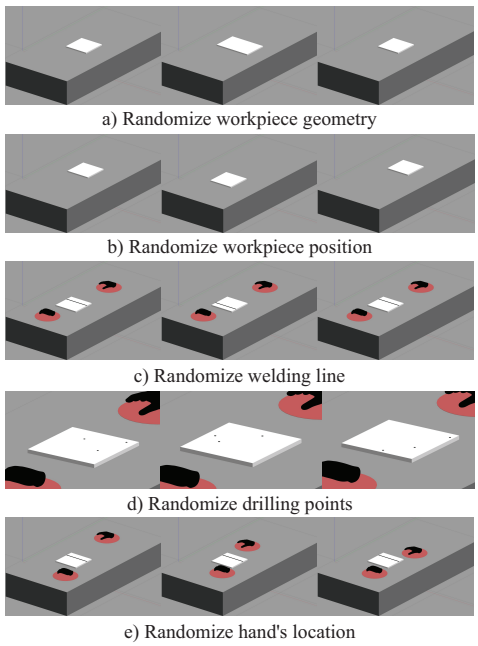


Fig. 2. Function demos.

All above functions are achieved using the ROS-Gazebo **SetModelState** service. Fig. 2 illustrates the demo of each functions.

3.2. *Online fault injector*

Fault injection is a promising approach for risk analysis in many safety-critical applications. In this study, signal faults are injected into joint position sensors to simulate potential malfunctions that may arise in real-world scenarios. When these faults are injected randomly, the onboard path planner may process erroneous position data during trajectory planning. This can lead to inaccurate trajectories, resulting in abnormal manipulator behavior and potential safety violations in HRC scenarios. Figure 3 illustrates the workflow of the online fault injector. During execution, both virtual sensor data (with injected faults) and real actuator data (reflecting the robot's response to the faults) are recorded as ROS topics.

In this study, faults with random fault parameters are generated and then injected into sensors. In our setup, only one fault is injected into a specific joint of the robotic manipulator during each round of execution. Fault parameters are shown in Table 1. A brief explanation of fault parameters is as follows:

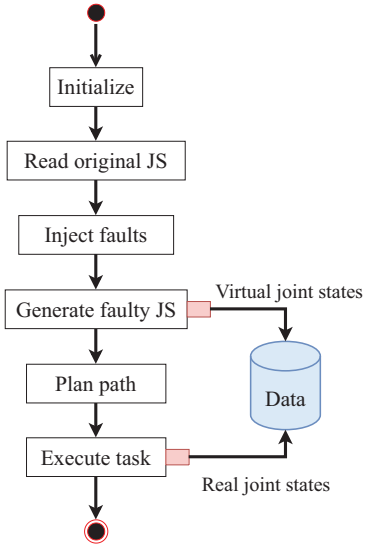


Fig. 3. The workflow of online fault injector.

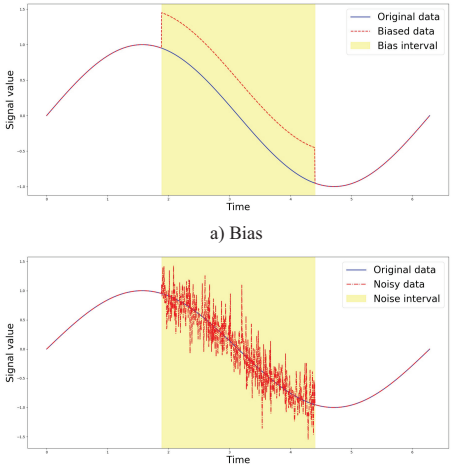


Fig. 4. Online fault demo.

- *Fault type:* Two common faults are considered, bias and noise. Biased sensory data could be caused due to unexpected loads or vibration in robotic applications, while noise might occur because of external environmental factors. The demo of faults is illustrated in Fig. 4.
- *Fault magnitude:* The value is randomly selected in a continuous interval.
- *Fault duration:* For how long the fault is injected.
- *Fault location:* Which joint position data of the manipulator would be manipulated. We select Franka Emika Panda, a 7-DoF manipulator, as our robot model.
- *Starting time:* According to the simulator clock, when the fault injector will start injecting faults.

3.3. *Execution monitor*

For collaborative robots, ISO/TS 15066 highlights four fundamental functions essential for HRC.

Table 1. Fault parameters

Parameters	Set notations
Type	T = {Bias, Noise}
Magnitude	M = (0, 1]
Duration	D = [1, 10]
Location	L = {Joint1, ..., Joint7}

One of these functions is *Speed and Separation Monitoring (SSM)*, where the robot is designed to maintain a consistent speed and a safe distance from humans. This function ensures protective stopping in response to unexpected robot behaviors. Building on this, we developed an execution monitor to detect safety violations caused by injected faults via calculating the distance between the robot's end effector and the human hand. Specifically, the monitor determines the pose of the end effector using Denavit-Hartenberg (DH) parameters, which define the robot's kinematic chain. The Euclidean distance between the end effector and the center of the human hand is then computed in real-time. In this case study, a recorded sample is classified as a safety violation if the observed distance is less than 0.2 meters. To facilitate the generation of time-series data and corresponding labels in ROS, the execution monitor integrates several ROS topics, including: virtual joint states (from sensors), real joint states (from actuators), fault labels, the distance between the hand and the end effector, safety violation labels, and fault parameters.

3.4. *Offline fault injector*

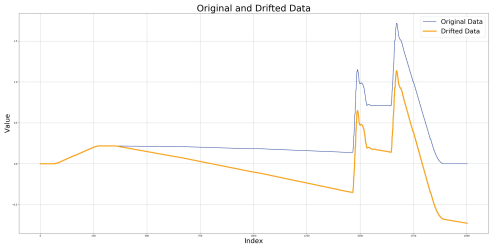
The key difference between online and offline fault injection is their purpose and application. Online fault injection allows us to know how the robot reacts in real-time when faults occur, while offline fault injection primarily addresses errors during data collection process. We integrate an offline fault injector based on Badgers Siebert et al. (2023) and it can be used to augment data with faults. In this case study, we select several practical faults, such as **Missing values** (randomly dropping some values), **Drift** (introducing a linear trend over a randomly selected time interval), **Random patterns** (injecting random patterns), and **Stuck-at zero** (randomly injecting zeros) as the fault injection options. Fig. 5 shows the corresponding fault demos.

4. Experiments

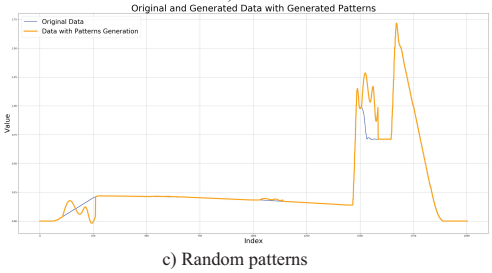
4.1. *Data collection*

In each execution round, the task scenario is configured using the scene generator's functions.

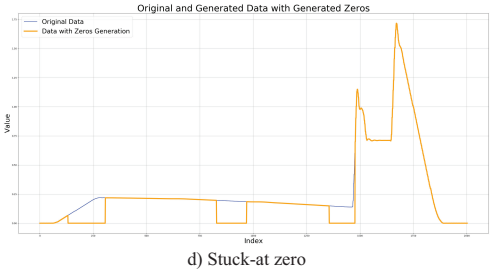
a) Missing value



b) Drift



c) Random patterns



d) Stuck-at zero

Fig. 5. Offline fault demo.

Then, fault injector starts injecting faults after the robot is added into simulator. During the subsequent task execution phase, the monitor is activated. It records all required time-series data and automatically labels injected faults and safety violations based on the robot's behavior.

4.2. *Results and analysis*

Fig. 6 shows a hazard demonstration of welding tasks when a bias fault is injected. From top to bottom, the plot illustrates the virtual sensor data

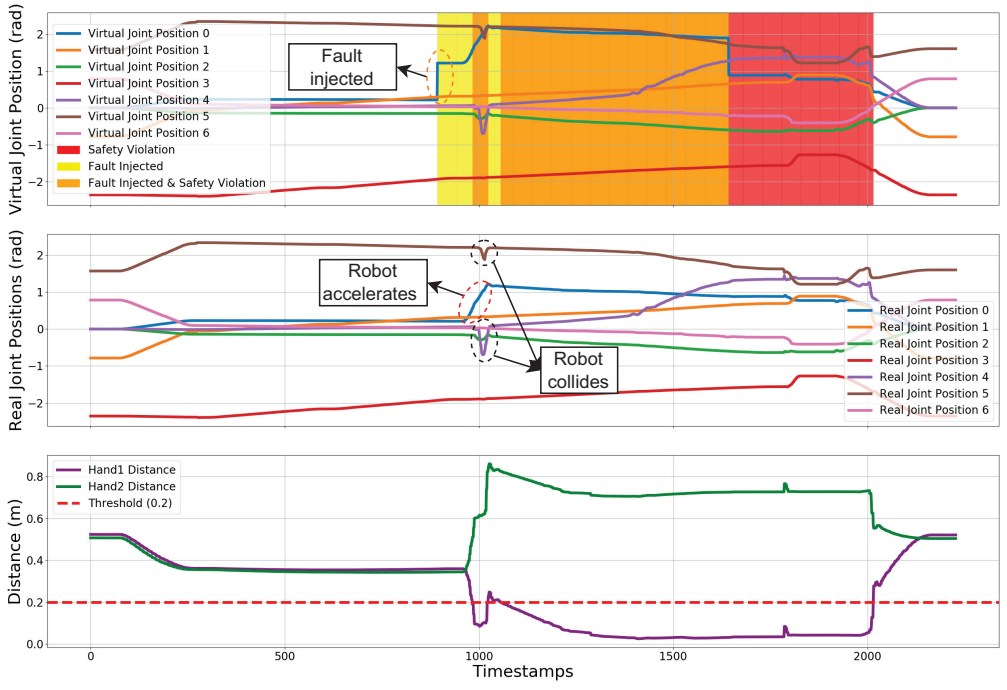


Fig. 6. Hazard demo.

Table 2. Fault injections experimental results

Fault types	Task	#Samples	#Faults	#Safety violations
Bias	Welding	107138	13301	1402
	Drilling	102682	13564	1092
Noise	Welding	86705	14187	0
	Drilling	86415	12690	0

(where faults are injected), the real sensor data (output from actuators), and the distance between the end effector and two hands, respectively. In the virtual sensor data plot, yellow, red, and orange intervals represent accidents that are **Fault Injected**, **Safety Violation**, and both. Based on these intervals, it can be concluded that a hazard does not immediately occur when a fault is injected. In fact, results from other fault injection experiments show that sensor malfunctions do not necessarily lead to danger. This is due to multiple factors, e.g., fault location, the current working scenario, and fault timing. Furthermore, by comparing virtual and real sensor signals, it can be concluded that signal channels with injected faults cause reactions in their corresponding real signal channels, while other channels remain unaffected. However, subsequent events, such as collisions, may induce abnormal patterns in other channels. Specifically, in Fig. 6, a bias fault is first injected into **Virtual Joint Position 0**, and then **Real Joint Position 0** exhibits a steep slope after several timestamps, while other channels remain unaffected. A sudden change in the position sensing signal indicates that the manipulator is accelerating, which can be observed in the simulator. Subsequently, a collision occurs due to the manipulator's unexpected acceleration, which is reflected in other data channels

(**Real Joint Position 4 and 5** in this demo).

To further explore the potential relationship between fault parameters and safety violations, we conduct extensive fault injection experiments for both industrial tasks. For each task, we execute 100 trials, with two types of faults equally injected (50 for bias and 50 for noise). The results are shown in Table 2. The results indicate that, in our setup, bias faults are more likely to cause safety violations compared to noise faults. In the simulator, bias faults often result in unexpected robot acceleration, which causes the manipulator to enter the safety zone and trigger a safety violation. In contrast, when the robotic system receives noisy data, the path planner typically can not plan the path until the noise stops. As such, noise faults mostly lead to system delays and omission of certain processes (e.g., skip a drilling step), rather than safety violations.

5. Conclusion

In this work, we introduced a time-series data generation tool designed to support risk assessment and anomaly detection in HRC applications. Built on the ROS-Gazebo simulator, the tool provides a structured approach to simulating real-world robotic tasks while allowing controlled fault injection and automatic hazard monitoring. The main contribution of this tool is its ability to systematically generate labeled time-series data for safety-critical robotic applications, addressing the challenge of scarce real-world fault data. Specifically, the tool: (1) Creates configurable industrial scenarios, enabling realistic task simulations. (2) Injects sensor faults dynamically, allowing for controlled analysis of robot behavior under failure conditions. (3) Monitors and labels hazards automatically, facilitating reliable dataset generation for anomaly detection. and (4) Supports offline fault injection, augmenting datasets for training AI models in risk assessment. Our experiments reveal that not all sensor faults lead to hazards, emphasizing the need for nuanced risk assessment. The insights gained from fault-injection experiments can enhance anomaly detection models by distinguishing between faults that genuinely impact safety and those that do not. By providing an easy-to-deploy, scalable, and reproducible framework, this tool serves as a foundational step toward improving safety analysis, fault classification, and AI-driven anomaly detection in robotics. Future work could extend the tool's capabilities by incorporating additional fault types, more complex scenarios, and AI-based safety mechanisms.

Acknowledgement

This research is supported by the Bundesanstalt für Arbeitsschutz und Arbeitsmedizin (BAuA, Germany) funds, project BAuA-555989-Me.

References

Chirayil Nandakumar, S., D. Mitchell, M. S. Erden, D. Flynn, and T. Lim (2024). Anomaly detection methods in autonomous robotic missions. *Sensors 24*(4), 1330.

Favier, A., A. Messioux, J. Guiochet, J.-C. Fabre, and C. Lesire (2020). A hierarchical fault tolerant architecture for an autonomous robot. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 122–129. IEEE.

Heo, Y. J., D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung (2019). Collision detection for industrial collaborative robots: A deep learning approach. *IEEE Robotics and Automation Letters 4*(2), 740–746.

Hsiao, Y.-S., Z. Wan, T. Jia, R. Ghosal, A. Mahmoud, A. Raychowdhury, D. Brooks, G.-Y. Wei, and V. J. Reddi (2023). Mavfi: An end-to-end fault analysis framework with anomaly detection and recovery for micro aerial vehicles. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6. IEEE.

Inceoglu, A., E. E. Aksoy, A. Cihan Ak, and S. Sariel (2021). Fino-net: A deep multimodal sensor fusion framework for manipulation failure detection. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6841–6847.

Ji, T., A. N. Sivakumar, G. Chowdhary, and K. Driggs-Campbell (2022). Proactive anomaly detection for robot navigation with multi-sensor fusion. *IEEE Robotics and Automation Letters 7*(2), 4975–4982.

Kamezaki, M., T. Wada, and S. Sugano (2024). Dynamic collaborative workspace based on human interference estimation for safe and productive human-robot collaboration. *IEEE Robotics and Automation Letters*.

Li, X., H. Alemzadeh, D. Chen, Z. Kalbarczyk, R. K. Iyer, and T. Kesavadas (2016). A hardware-in-the-loop simulator for safety training in robotic surgery. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5291–5296.

Luo, Y., Y. Xiao, L. Cheng, G. Peng, and D. Yao (2021). Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Computing Surveys (CSUR) 54*(5), 1–36.

Ma, Y., P. Grimmeisen, and A. Morozov (2023). Case study: Ros-based fault injection for risk analysis of robotic manipulator. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pp. 1–6.

Malm, T., T. Salmi, I. Marstio, and J. Montonen (2019). Dynamic safety system for collaboration of operators and industrial robots. *Open engineering 9*(1), 61–71.

Park, K. M., J. Kim, J. Park, and F. C. Park (2021). Learning-based real-time detection of robot collisions without joint torque sensors. *IEEE Robotics and Automation Letters 6*(1), 103–110.

Quigley, M., K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, Volume 3, pp. 5. Kobe.

Siebert, J., D. Seifert, P. Kelbert, M. Kläs, and A. Trendowicz (2023). Badgers: generating data quality deficits with python.

Vrontis, D., M. Christofi, V. Pereira, S. Tarba, A. Makrides, and E. Trichina (2023). Artificial intelligence, robotics, advanced technologies and human resource management: a systematic review. *Artificial intelligence and international HRM*, 172–201.