Proceedings of the 33rd European Safety and Reliability Conference (ESREL 2023)
Edited by Mário P. Brito, Terje Aven, Piero Baraldi, Marko Čepin and Enrico Zio
 Published by Research Publishing, Singapore.
doi: 10.3850/978-981-18-8071-1_P207-cd

Performance Index Modeling from Fault Injection Analysis for an Autonomous Lane-Keeping System

Parthib Khound

Chair of Reliability of Technical Systems and Electrical Measurement, University of Siegen, 57076 Siegen, Germany. E-mail: parthib.khound@uni-siegen.de

Omar Mohammed

Chair of Reliability of Technical Systems and Electrical Measurement, University of Siegen, 57076 Siegen, Germany. E-mail: omar.mohammed@uni-siegen.de

Peng Su

Unit of Mechatronics and Embeded Control Systems, KTH Royal Institute of Technology, 100 44 Stockholm, Sweden. E-mail: pensu@kth.se

Dejiu Chen

Unit of Mechatronics and Embeded Control Systems, KTH Royal Institute of Technology, 100 44 Stockholm, Sweden. E-mail: chen@md.kth.se

Frank Gronwald

Chair of Reliability of Technical Systems and Electrical Measurement, University of Siegen, 57076 Siegen, Germany. E-mail: frank.gronwald@uni-siegen.de

A faulty sensor data could not only undermine the stability but also drastically compromise the safety of autonomous systems. The reliability of the functional operation can be significantly enhanced, if any monitoring modules can evaluate the risk on the system for a particular fault in a sensor. Based on the estimated risk, the system can then execute the necessary safety operation. To develop a risk evaluating algorithm, the relation between the faults and the effects should be known. Therefore, to establish such cause-and-effect relationship, this paper presents a performance indexing method that quantifies the effects caused by given fault types with different intensities. Here, the considered system is a lane keeping robot and the only sensor used for the functional operation is a red, green, and blue (RGB) camera. The lane keeping algorithm is modeled using a supervised artificial intelligence (AI) learning method. To quantify the effects with performance indices (PIs), different faults are injected to the RBG camera. For an injected fault type, the system's PI is evaluated from the AI algorithm's (open-loop) outcome and the lane keeping (closed-loop) outcome. The lane keeping/closed-loop outcome is quantified from the trajectory data computed using the strapdown inertial navigation algorithm with the measurement data from a 6D inertial measurement unit (IMU).

Keywords: fault injection, RGB camera fault, performance index, IMU trajectory, strapdown inertial navigation.

1. Introduction

For safety critical and real time applications, such as automated driving, any faulty data from a sensor could not only alter the driving performance but could also lead to fatal consequences. Therefore, to ensure the safety, the risk palliating concepts, such as, fault detection, isolation and recovery (FDIR) methods, are very useful. The fault tolerant methods perform fault diagnosis for executing a remedial action, see e.g., Blanke et al. (2000); Gao et al. (2015). The health monitoring schemes estimate the health of a system in real time and are actively discussed field of research, see e.g., Loureiro et al. (2014); Gomes and Wolf (2021). The safety of an autonomous system can be remarkably enhanced, if a monitoring module can estimate the risk from a detected fault in real time, such that a necessary action for fail-safe and fail-operation can be executed before the occurrence of a system failure. The fault detection and

risk estimation can guarantee reliable functional operations. Note that different faults in the sensors lead to different outcomes. Therefore, the prior information of the cause-and-effect relationship would be very useful for the risk evaluation. In this context, the cause is the fault and the effect is the respective system's performance degradation and failure. For numerical computations, the performance degradation should be quantified. This paper presents a method to evaluate and quantify the performance of a lane keeping robot subjected to different faults injected to the primary sensor, i.e., an RGB camera. Here, this quantified value of the outcome/performance degradation of the robot in regard to safety is termed as the performance index (PI).

There exist many fault injection methods, based on signal conditioning (Yang et al., 2017) or applied to AI networks (Liu et al., 2017; Su and Chen, 2022), sensor signals (Mitra et al., 2018; Secci and Ceccarelli, 2020), etc. In Mitra et al. (2018), perception errors are statistically injected to the bounding box and their corresponding effects on the brake torque outputs of the wheels of an autonomous vehicle are presented. In Secci and Ceccarelli (2020), the analyses of the effects on an autonomous driving application for different fault types in an RGB camera are discussed. Indeed, different fault types have certain effects on the functional operation. Additionally, the intensity of faults may amplify the effects to different degrees. For example, a slightly blurry image would not necessarily alter the lane keeping performance of the robot, but if the image is extremely blurred, the lane keeping operation could completely fail. In this paper, the faults with different types and intensities are injected to the input image of the RGB camera. This emulates the possible hardware and environmental faults on the sensor. Some common examples of the fault types are blur, speckle noise, broken lens, etc. Here, the intensity represents a defined quantification of a fault type categorized into three levels, i.e., slight, medium, and extreme. For the fault injection, our in-house tool is used (Mohammed, 2022a). The fault detection and classification method to evaluate the fault type and intensity developed for the RGB camera is separately presented in Mohammed et al. (2023).

The final PI for each fault type and intensity is computed by combining two quantities. The first one is the deviation of the actuation output of the AI algorithm from that of during the normal operation, i.e., without any faults. This is an offline/static/open-loop analysis. The second quantity is the deviation of the final trajectory of the robot from that of the normal operation and this is an online/dynamic/closed-loop analysis. This trajectory is estimated from 6D IMU sensor data, i.e., the data from three orthogonal gyroscopes and accelerometers, using the strapdown inertial navigation algorithm (S-INA) (Woodman, 2007). The dead reckoning solely based on a 6D IMU is highly prone to errors and the results significantly drift over time. Therefore, sensor fusion with other sensors based on the extended Kalman filter (EKF) is used for the inertial navigation, see e.g., Malyavej et al. (2013). Currently, for the presented system, only the 6D IMU sensor is available for the trajectory estimation. Thus, considering this increase in drift and inaccuracy, the trajectory data will be obtained only for a small section of the whole track within a short duration of time. In Mohammed et al. (2023), the preliminary concept for a risk index evaluation is proposed for the same lane keeping robot. The method partly uses an intuitive quantification of lane keeping performance based on experimental observation, which is definitely not an optimal way from a technical context. The proposed method overcomes this by using the 6D IMU data for assessing the quality of the lane keeping operation instead of any partial intuitive assessment.

The article is structured as follows: In Section 2, the system's architecture, functionality, and the fault injection method are described. Section 3 presents the open-loop analysis method, demonstrating some experimental results for some faults and evaluating the relevant PIs from this method. Similarly, the closed-loop analysis is presented in Section 4. In Section 5, the final PI is evaluated by combining the outcomes from the open-loop and the closed-loop analyses. Finally, this paper is concluded in Section 6.

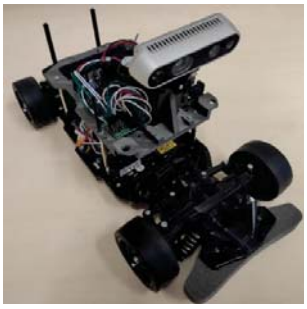

Fig. 1. The lane keeping robot.

2. Background

2.1. *Hardware Architecture*

Fig. 1 shows the lane keeping robot, which was built at the University of Siegen during a master thesis (Mohammed, 2022b). The primary hardware components of the system are: TAMIYA 1:10 RC TT-02 (chassis), NVDIA Jetson Nano Developer Kit (processing unit), Intel RealSense depth camera D435i (sensor module), and PCA9685 (PWM motor driver). The sensor module includes an RGB and depth camera sensors with a 6D IMU. The IMU comprises of three orthogonal gyroscopes and accelerometers. The RGB camera is the main sensing unit of the system that is used for the lane keeping algorithm. The IMU is only used for the closed-loop performance evaluation.

2.2. *Lane-Keeping Algorithm*

The main function of the system is to keep the robot between two lane marks while driving on a prescribed path. The lane marks are fixed in a laboratory. The lane keeping algorithm is developed using the JetRacer project, see NVDIA-AI-IOT (2021). This uses a residual neural network (ResNet) with 18 deep learning layers, termed as ResNet-18 (He et al., 2016). The camera sensor feeds the RGB image data to the AI model in real time. Then the algorithm accordingly evaluates a suitable steering value that actuates a motor connected to the front steering axle. The steering value is in the range $[-1, 1]$, where -1 and 1 are the maximum actuation input to turn the front wheel steering axle to the left and right direction,

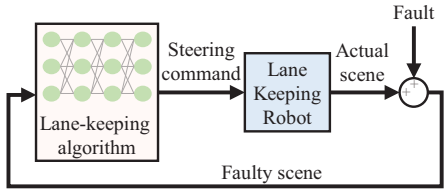

Fig. 2. The closed-loop system.

respectively. This steering action keeps the robot between the lane marks. The robot is trained with several scenarios to optimally keep itself between the lane marks, e.g., when it is outside the lane, at the extreme corner of a lane mark, within a curve track, etc.

2.3. *Fault Injection*

Fig. 2 illustrates the overview of closed-loop system with a fault. An in-house fault injecting tool is developed for injecting different environmental and hardware related faults to the RGB camera, see Mohammed (2022a). The tool provides a graphical user interface (GUI) where a fault type and intensity can be selected from the console. It generates a distorted form of the original RGB image corresponding to the selected fault type with a specified intensity.

Fig. 3 shows examples of some faulty RGB images generated by the tool with different fault types and intensities. Here, only three levels of intensity are used, i.e., slight, medium, and extreme. These intensity levels for the fault types blur, speckle noise, and no chromatic aberration correction (NCAC) are implemented with the help of some pre-implemented libraries to distort the image data by manipulating certain image parameters, see Mohammed, 2022a. However, the rain and crack are injected using the image overlapping method. Thus, the overlapping images for each intensity level are selected based on intuition. In the future work, a suitable alternative approach will be investigated for these overlapping fault types.

3. Open-loop Analysis

The trained lane keeping AI algorithm of the robot computes a suitable steering input for the

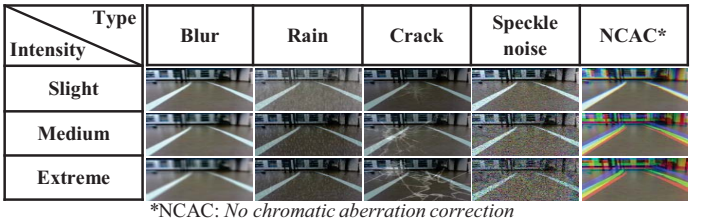

Fig. 3. Some results from the fault injecting tool.

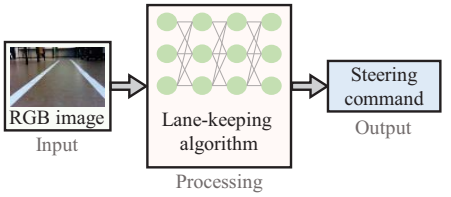

Fig. 4. The lane keeping algorithm: the open-loop system.

actuation based on the input RGB image obtained from the camera sensor, see Fig. 4. For the open-loop analysis, the robot is first driven through a prescribed course, recording all the RGB input images, in the sequence of occurrence, without injecting any faults. Later, all the recorded images are distorted by injecting each fault type and intensity shown by Fig. 3. The original and the distorted images by each fault are then separately fed to the AI algorithm and the associated steering commands are stored in the same sequence of occurrence, without actually running the robot. Thus, this part analyzes the open-loop AI function illustrated by Fig. 4 for different faults.

The deviation of the steering values (algorithm's outputs) corresponding to an injected fault from that of the normal operation is evaluated using the root mean square error (RMSE) (Mohammed et al., 2023), given by

$$\Theta_{T,I}(n) = \sqrt{\frac{\sum_{i=1}^{n} \{\theta_{T,I}(i) - \theta(i)\}^2}{n}}\,, \quad s.t. \quad 1 \leq n \leq N\,, \tag{1}$$

where $\theta(i)$ is the steering value (algorithm's output) corresponding to the i-th input frame without any faults. The variable $\theta_{T,I}(i)$ also denotes the steering value but output from the distorted input image with a fault type T and intensity I. The parameter N represents the total number of images arranged in the sequence of occurrence. Hence, the variable $\Theta_{T,I}(n)$ denotes the RMSE for n samples in connection to the fault type and intensity T and I, respectively. The higher the RMSE value, the higher the deviation and, hence, the worse the performance. The RMSE profiles for all the combinations of the considered fault types and intensities are shown in Fig. 5. The faults with speckle noise and NCAC have consistently the high values for all the intensity levels.

The final RMSE values for the entire N samples, i.e., $\Theta_{T,I}(N)$, is normalized and used for the PI evaluation, given by

$$PI^{OL}_{T,I} = \frac{\Theta_{T,I}(N)}{\max\limits_{\forall T \& I} \{\Theta_{T,I}(N)\}}\,. \tag{2}$$

Here $PI^{OL}_{T,I}$ represents the open-loop performance index for a fault type and intensity T and I, respectively. In Table 1, all the values of $PI^{OL}_{T,I}$ for the considered faults are listed. This analysis is static, hence, for the same set of inputs the evaluated PIs with respect to a fault type and intensity will be the same for any number of tests.

4. Closed-loop Analysis

The closed-loop analysis studies the outcome of whole system in action, illustrated by Fig. 2. For a driving application, the trajectories data provides the outcome of the system. A reliable dead-reckoning method is necessary to accurately estimate the trajectory. Moreover, the accuracy of a dead reckoning relies on several sensors. However, the presented lane-keeping robot has only a 6D IMU sensor for the trajectory estimation. The 6D refers to three orthogonal gyroscopes and

Table 1. The open-loop performance indies.

Faults	Slight Intensity	Medium Intensity	Extreme Intensity
Blur	0.2168	0.3584	0.6446
Rain	0.3100	0.5817	0.5046
Crack	0.2955	0.1875	0.3671
Speckle noise	0.7832	0.7786	0.8533
NCAC	0.9910	1.0000	0.9747

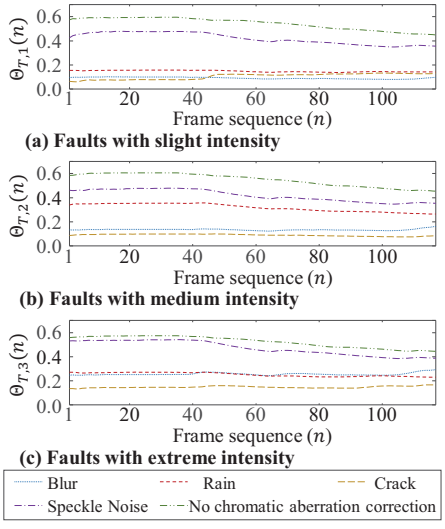

Fig. 5. The open-loop RMSE profiles.

accelerometers. The S-INA is used to estimate the trajectory from the rotational velocity and the acceleration data (Woodman, 2007). Note that, the accuracy of a trajectory cannot be expected only from the 6D IMU, due to drift, quantization errors, random noises, etc., see Woodman (2007). Here, this inaccurate estimation is used only to illustrate the concept of the PI modeling. It is worth noting that in Brossard et al. (2020), the accuracy of the dead-reckoning only from a 6D IMU is significantly enhanced by combining the results from the S-INA and a pseudo measurement using an invariant EKF (IEKF). The noise parameters of the IKEF is adapted in real time based on some deep neural networks. In the future work, the focus will be to enhance the localization method by either adding more sensors to the robot or exploiting the method presented in Brossard et al. (2020).

4.1. *Trajectory Estimation Method*

The S-INA involves updating the rotation matrix with single integration and then applying double integration to obtain the position data. The rotation matrix $\mathbf{R_{sb}}(t)$ from the body-fix to the static inertial coordinate frame is updated at each time-step dt as

$$\mathbf{R_{sb}}(t+dt) = \mathbf{R_{sb}}(t)\, \exp\left\{\mathbf{\Omega_b}(t)\, dt\right\}\,, \quad (3)$$

where $\mathbf{\Omega_b}(t)$ is the skew symmetric representation of the angular rate $\underline{\omega}_b$. The vector $\underline{\omega}_b$ is indeed the gyroscope data corresponding to the x_b, y_b, and z_b axes of the body-fix frame, given by

$$\underline{\omega}_b(t) = \begin{bmatrix}\omega_{b,x}(t) & \omega_{b,y}(t) & \omega_{b,z}(t)\end{bmatrix}^T\,. \quad (4)$$

It is important to be consistent with the order of the rotational convention. Here, the yaw-pitch-roll rotational order is considered to transform from the inertial to the body-fix frame, hence,

$$\mathbf{\Omega_b}(t) = \begin{bmatrix} 0 & -\omega_{b,z}(t) & \omega_{b,y}(t) \\ \omega_{b,z}(t) & 0 & -\omega_{b,x}(t) \\ -\omega_{b,y}(t) & \omega_{b,x}(t) & 0 \end{bmatrix}\,, \quad (5)$$

see Woodman (2007). Following the matrix update and stating the initial state as stationary, the acceleration of the robot with respect to the static inertial reference frame is obtained using

$$\underline{a}(t) = \mathbf{R_{sb}}(t)\, \underline{a}_b(t) - \underline{a}_g(0)\,, \quad (6)$$

where $\underline{a}_b(t)$ is a vector containing the accelerometer data corresponding to the x_b, y_b, and z_b axes of the body-fix frame, i.e.,

$$\underline{a}_b(t) = \begin{bmatrix}a_{b,x}(t) & a_{b,y}(t) & a_{b,z}(t)\end{bmatrix}^T\,. \quad (7)$$

Here $\underline{a}_g(0)$ is the initial acceleration of the static frame due to gravity. Finally, the position with respect to the static inertial frame, i.e., $(x(t), y(t), z(t))$, and from a stationary initial condition is obtained by

$$\underline{x}(t) = \int_0^t \left(\int_0^{\tau_2} \underline{a}(\tau_1) d\tau_1 \right) d\tau_2 \,. \tag{8}$$

For the above computations, the initial conditions should be first assigned. The static coordinate frame is transformed when the robot is stationary, such that

$$\underline{a}_g(0) = \begin{bmatrix} 0 & 0 & -|\underline{a}_b(0)| \end{bmatrix}^T \,. \tag{9}$$

The modulus operator $|\cdot|$ over a vector denotes the magnitude of the vector. So, $|\underline{a}_b(0)|$ is the magnitude of the acceleration due to gravity measured by the accelerometer during the stationary condition. Following this, the initial rotation matrix is

$$\begin{aligned} \mathbf{R_{sb}}(0) \;=\; & \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ & \times \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \\ & \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} . \end{aligned} \tag{10}$$

Here, γ, β, and α denote the initial yaw, pitch, and roll angles, respectively, that are required to transform the static inertial coordinate frame to the initial body-fix frame such that the acceleration due to the gravity in the static inertial frame is given by Eq. (9). Considering the yaw-pitch-roll rotation sequence, γ is equal to zero and the other parameters are evaluated by solving

$$\begin{bmatrix} -\sin\beta \\ (\cos\beta\ \sin\alpha) \\ (\cos\beta\ \cos\alpha) \end{bmatrix} = -|\underline{a}_b(0)|^{-1}\, \underline{a}_b(0) \,, \tag{11}$$

see Pedley (2013).

4.2. *Trajectory Deviation Evaluation*

For the closed-loop PI evaluation, the deviation of trajectory data due to an injected fault from that of without any fault is obtained using the RMSE as well. The RMSE at a time t is given by

$$\Delta_{T,I}(t) = \sqrt{\frac{\int_0^t \left| \underline{x}_{T,I}(\tau) - \underline{x}(\tau) \right|^2 d\tau}{t}} \,, \quad s.t. \qquad t \neq 0 \,. \tag{12}$$

The vector $\underline{x}_{T,I}(t)$ denotes the position of the robot with respect to the static inertial frame when a fault with type T and intensity I is injected to the input image data. Similar to the open-loop PI evaluation, the closed-loop PI ($PI^{CL}_{T,I}$) is evaluated by normalizing the RMSE for a total time t_F, i.e., when the trajectory does not drift significantly. This is given by

$$PI^{CL}_{T,I} = \frac{\Delta_{T,I}(t_F)}{\max\limits_{\forall T \& I} \{\Delta_{T,I}(t_F)\}} \,. \tag{13}$$

Ideally, t_F should cover the entire track. But as already mentioned, the S-INA based only on a 6D IMU are prone to errors and the results drift over time. Therefore, here, this value is around 6 s which only covers a small section of the entire track. The RMSE profiles from the closed-loop experiments are shown in Fig. 6. The calculated values of the closed-loop PI, i.e., $PI^{CL}_{T,I}$, are organized in Table 2. The limitations of a 6D IMU adversely affect the repeatability of the results with random drifts. Therefore, just for the presentation of the concept, each PI value in Table 2 is obtained from a single test.

5. The Final Performance Index

The open-loop deviation described by $PI^{OL}_{T,I}$ outlines the variation in the actuation commands. While the closed-loop deviation quantified by $PI^{CL}_{T,I}$ describes the fluctuation in the transmission dynamics. Both these parameters measure the performance of the fault injected system compared to the ideal operation. Therefore, the final PI is defined by combining these two parameters, given by

$$PI_{T,I} = \frac{b\, PI^{OL}_{T,I} + (1-b)\ PI^{CL}_{T,I}}{\max\limits_{\forall T \& I} \left\{ b\, PI^{OL}_{T,I} + (1-b)\ PI^{CL}_{T,I} \right\}} \,, \quad b \in \{0\,, 0.25\,, 0.50\,, 0.75\,, 1.00\} \,, \tag{14}$$

Table 2. The closed-loop performance indies.

Faults	Slight Intensity	Medium Intensity	Extreme Intensity
Blur	0.3228	0.5417	0.4872
Rain	0.4561	0.5373	0.5310
Crack	0.3067	0.4197	0.4430
Speckle noise	0.3462	1.0000	0.6196
NCAC	0.1664	0.5197	0.4602

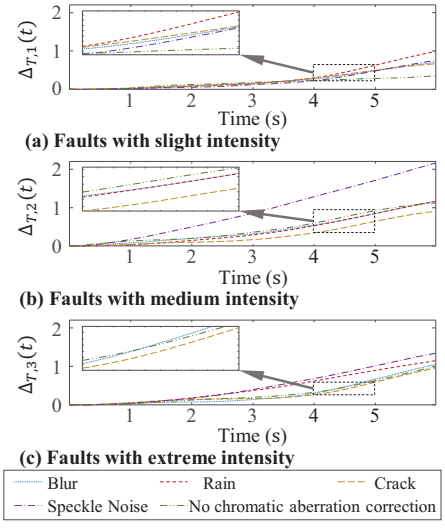

Fig. 6. The closed-loop RMSE profiles.

where b is a bias factor, with four possible values. This factor is equal to zero when the bias is completely towards the closed-loop results. The value 0.25 means that the majority of the weight is towards the closed-loop analysis. This should be the ideal preference when the trajectory data can be accurately obtained. Thus, 0.50 implies that results from both the analyses are equally weighted, and vise-versa. The trajectory data computed by the S-INA only from a 6D IMU sensor data is not accurate. Therefore, here the bias is 0.75. The final PI values corresponding to all the considered fault types and intensities are shown in Table 3. The higher the value of the PI, the worse the performance.

6. Conclusion

This contribution presents a performance indexing method to quantify the performance of a lane keeping robot subjected to different faults to the primary sensor, i.e., an RGB camera. The faults injected to the RGB image data are categorized into different types and intensities. The PI is evaluated by combining the results from open-loop and closed-loop analyses using a biasing factor. Based on this factor, the final PI is inclined more towards one of the analysis results. The open-loop analysis involves computing the deviation of the steering actuation based on the fault injected input data from that of the normal data. While in the closed-loop analysis, the deviation of the final trajectory data of the fault injected system from that of the normal one is calculated. Indeed, the closed-loop approach applies to the system run-time and the open-loop approach lacks the ground-truth. Therefore, ideally the bias should be more towards the closed-loop approach. However, here, the trajectory data is estimated only from a 6D IMU sensor using the S-INA. The dead-reckoning solely based on a 6D IMU suffers from extreme drift, quantization errors, and random noises. Considering this limitation of the presented system, the bias is set more towards the open-loop results, which is definitively not ideal. Hence, in the future work, focus will be on enhancing the trajectory estimation method using sensor fusion techniques.

Acknowledgement

This work is part of the TRUST-E project (TRUST-E, 2021) and funded by *Bundesministerium für Bildung und Forschung* (BMBF), under contract PENTA 16ME0320K -16ME0329. The authors would like to thank all the members of the project for useful discus-

Table 3. The final performance indies.

Faults	Slight Intensity	Medium Intensity	Extreme Intensity
Blur	0.2765	0.4594	0.6879
Rain	0.3938	0.6485	0.5810
Crack	0.3390	0.2790	0.4388
Speckle noise	0.7660	0.9477	0.9033
NCAC	0.8919	1.0000	0.9615

sions which stimulated the writing of this paper.

References

Blanke, M., W. C. Frei, F. Kraus, J. R. Patton, and M. Staroswiecki (2000). What is fault-tolerant control? *IFAC Proceedings Volumes 33*(11), 41–52.

Brossard, M., A. Barrau, and S. Bonnabel (2020). AI-IMU dead-reckoning. *IEEE Transactions on Intelligent Vehicles 5*(4), 585–595.

Gao, Z., C. Cecati, and S. X. Ding (2015). A survey of fault diagnosis and fault-tolerant techniques—part I: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics 62*(6), 3757–3767.

Gomes, I. P. and D. F. Wolf (2021). Health monitoring system for autonomous vehicles using dynamic bayesian networks for diagnosis and prognosis. *Journal of Intelligent & Robotic Systems 101*(19), 1–21.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Liu, Y., L. Wei, B. Luo, and Q. Xu (2017). Fault injection attack on deep neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 131–138.

Loureiro, R., S. Benmoussa, Y. Touati, R. Merzouki, and B. O. Bouamama (2014). Integration of fault diagnosis and fault-tolerant control for health monitoring of a class of MIMO intelligent autonomous vehicles. *IEEE Transactions on Vehicular Technology 63*(1), 30–39.

Malyavej, V., W. Kumkeaw, and M. Aorpimai (2013). Indoor robot localization by RSSI/IMU sensor fusion. In *10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, pp. 1–6.

Mitra, P., A. Choudhury, V. R. Aparow, G. Kulandaivelu, and J. Dauwels (2018). Towards modeling of perception errors in autonomous vehicles. In *21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3024–3029.

Mohammed, O. (2022a). Fault injecting tool. `https://github.com/omarMohammed-USI/Faults-injecting-tool-USI`. (accessed Mar. 10, 2023).

Mohammed, O. (2022b). *Faults and fault injection in camera based systems for assisted driving*. Master thesis, University of Siegen, Siegen, Germany.

Mohammed, O., P. Khound, P. Su, D. Chen, and F. Gronwald (2023). Multilevel artificial intelligence classification of faulty image data for enhancing sensor reliability. In *Proceedings of the 33rd European Safety and Reliability Conference (ESREL)*.

Mohammed, O., P. Khound, B. Vandevelde, and F. Gronwald (2023). Health index modeling for trustable electronic sensor systems in an autonomous application. In *Smart Systems Integration (SSI)*.

NVDIA-AI-IOT (2021). JetRacer — an autonomous AI racecar using NVIDIA Jetson Nano. `https://github.com/NVIDIA-AI-IOT/jetracer`. (accessed Mar. 10, 2023).

Pedley, M. (2013). Tilt sensing using a three-axis accelerometer. *Freescale Semiconductor application note 1*, 1–22.

Secci, F. and A. Ceccarelli (2020). On failures of RGB cameras and their effects in autonomous driving applications. In *31st International Symposium on Software Reliability Engineering (ISSRE)*, pp. 13–24.

Su, P. and D. Chen (2022). Using fault injection for the training of functions to detect soft errors of DNNs in automotive vehicles. In *Proceedings of the 17th International Conference on Dependability of Computer Systems DepCoS-RELCOMEX*, pp. 308–318.

TRUST-E (2021). Project overview. `https://www.edacentrum.de/en/projects/TRUST-E`. (accessed Mar. 10, 2023).

Woodman, O. J. (2007). An introduction to inertial navigation. Technical report, University of Cambridge, Computer Laboratory.

Yang, C., C. Yang, T. Peng, X. Yang, and W. Gui (2017). A fault-injection strategy for traction drive control systems. *IEEE Transactions on Industrial Electronics 64*(7), 5719–5727.