

Surface Corrosion Detection for Ferrous-metal Parts: Application of Artificial Intelligence, Python and Microscopic Images

Roman WADOWIK^{1*}, Artur BELZO¹, Regita BENDIKIENE²

¹ Rzeszów University of Technology, al. Powstańców Warszawy 12, 35-959 Rzeszów, Poland

² Kaunas University of Technology, Studentu st. 56, 51424, Kaunas, Lithuania

<http://doi.org/10.5755/j02.ms.41377>

Received 2 May 2025; accepted 19 June 2025

This paper presents a novel method for the identification of surface damage, in particular corrosion, in ferrous metals based on generative artificial intelligence (GenAI), showing how to automate damage identification and corrosion recognition. The methodology involved using optical microscopy to capture electrochemical corrosion patterns, followed by image preprocessing and classification using AI algorithms implemented in Python. High-quality microscopic images have been recorded, based on selected ferrous metals. Python code lines were generated using ChatGPTTM based on queries created by the authors, and this method was applied to the corrosion analysis. Quantitative evaluation confirmed Python code parameters-dependent detection accuracy and repeatability, demonstrating the robustness of the proposed technique. The results were discussed in terms of possible industrial applications. In addition, the limitations of the results obtained, which sometimes fall short of the claims inspector's expectations, were discussed. Compared to traditional corrosion detection methods such as visual inspection and non-destructive testing, AI-based methods are a faster and more cost-effective solution that can process large volumes of images in real time and produce consistent results. Further research directions are also suggested, including the analysis of other types of damage and improving the accuracy of the model. In addition to technical efficiencies, the broader impact of these studies is that they can contribute to predictive maintenance, reduce downtime and improve safety in industries with high ferrous metal use.

Keywords: damage identification, corrosion, surface, steel, artificial intelligence.

1. INTRODUCTION

The surfaces of steel products have to meet the design-related requirements for specific functional properties, such as tribological properties, leak tightness, good adhesion of coatings, external appearance, etc. The exploitation of machines and steel products leads to wear of untreated or coated surfaces due to normal wear processes (e.g., abrasion caused by friction between interoperating machine parts or contact with operating media) and impact of external factors which are difficult to predict during the design process (e.g., environmental impact such as electrochemical corrosion caused by unexpected exposure to water or chemical substances, mechanical impacts, exposure to high temperature, etc.). Identification of such damages linked to normal exploitation is crucial to plan maintenance tasks or even part repair. It also corresponds to quality assurance tasks. Literature underlines the importance of approaches that help to analyse the damages. The main method, which has been utilized for decades, is based on microscopic observations that often depend on human experience, which can lead to inconsistency and subjectivity.

For instance, Wdowik and Świrad [1] analysed selected tools of the software using a microscope, coupled with focus variation technology. It enabled them to analyse selected features of ceramic chips. Xing F. et al. [2] presented a machine learning-based tool for microscopic image analysis. Cross S. in his review [3] discusses the advantages and problems of fractal geometric analysis and its current

applications in the field of microscopy. In the paper of Long F. et al. [4] the authors present basic methods for visualizing typical 3D images and related multi-scale, multi-timepoint, multi-color data sets. Merchant F. and Castleman K. [5] present investigations regarding image processing and analysis with the tools to improve the visual quality of images and extract quantitative information. Developments in microscopy, such as light sheet microscopy, digital microscopy, and the use of deep learning techniques for image segmentation and analysis with big data image informatics and management, were discussed in the abovementioned work too. Li W. and co-authors [6] developed an automated defect recognition tool in electron microscopy based on a computer vision approach. Object detector, convolutional neural network, and local image analysis methods are applied. Their proposed approach works for images of varying contrast, brightness, and magnification. In another paper [7], the authors present several popular machine learning techniques and their applications in microscopic image analysis. Machine learning can effectively and efficiently deal with the complexity and diversity of microscopic images and plays an important role in image-based computer-aided diagnosis. Connolly C. in his paper [8] investigates technologies that improve image quality in microscopy. A new technique for high-content analysis is described. The abovementioned paper describes a 3D visualisation package, hardware, and software for viewing of high-resolution microscopic images online. In the work of Sanaei N. et al. [9] the number of

* Corresponding author: R. Wdowik
E-mail: rwdowik@prz.edu.pl

defects content in metal additive manufacturing (AM) specimens and the correlations between the defect characteristics (size, sphericity/circularity, aspect ratio) were investigated using 2D and 3D defect characterisation methods. They analyse the location-based distributions of defect characteristics in AM samples and study the variations of defect characteristics in these samples. Satorres Martínez S. and co-authors [10] present a machine vision system to reveal, detect, and characterise defects on non-planar transparent surfaces. The cornerstone of this machine's vision is an innovative lighting system. A new adaptive threshold selection algorithm is proposed. The lighting system is compared with other commercial lighting systems. The performance of the system is evaluated through a series of tests using a commercial headlamp lens model [10]. Garcia-Perez A. et al. [11] present a tool wear monitoring system using computer vision that analyses tool inserts after the machining process is completed. They perform research by installing a camera and a tool cleaning device in the robot magazine room to remove chips and cooling residue, allowing for a neat image of the tool. Xie X. in the paper [12] presents a review of methods of surface inspection using computer vision and image processing techniques. The same work mentioned above focuses on visual surface inspection as a texture analysis problem to distinguish features extracted from normal and defective regions. Liu Z. et al. [13] and Chen Y. et al. [14] present a review of the applications of deep learning algorithms in microscopy image analysis and surface defect detection, which is a key part of quality inspection of industrial products. The latest and traditional methods are compared and the common problems and their solutions on industrial surfaces defect detection are summarized. Wang M. and co-authors in the paper [15] focus on the detection of rail surface defects. The authors of this paper propose a novel model for detection rail surface defects. The mechanism includes a sharpening filter for defects on railway tracks to enhance the details of the underlying features of fine defects. Their second aim is to collect data from real rail images to train an object detection network and use it as open-source. Giorgio Cavaliere's doctoral dissertation [16] focuses on the use of advanced vision systems for surface defect detection, applying vision systems and artificial intelligence. Automated inspection systems utilize high-speed cameras, image processing algorithms, architectural neuronal networks, and artificial intelligence (AI). The results reveal that automated inspection systems offer improved quality assurance processes and the integration of 2D/3D technologies in industrial applications [16].

The microscopic studies play a crucial role in damage analysis, and due to the intensive progress of cheap and accessible AI-based tools, the authors believe it is important to test the potential applications of this novel and cheap (from user's perspective) technology, also taking into account well-known data acquisition methods such as optical microscopy. In this context, the analysis presented in the study is performed to investigate potential applications of the generative artificial intelligence along with the Python language to support the detection of damages of ferrous-metal parts. In the case of the presented study, the authors focus mainly on corrosion detection and the calculation of selected quantitative indicators using AI. The

goal of this study is to develop a reliable system capable of automatically and accurately detecting corrosion, which could eventually be integrated into real-time monitoring systems for industrial applications.

2. EXPERIMENTAL DETAILS

Experiments presented in the study are based on the analysis of corrosion on ferrous-metal products. Corrosion can be recorded in photographs (images) taken by the use of USB microscope produced by DinoLite company (Fig. 1). The colour of corroded areas must be clearly different than colour of other areas. In the case of ferrous products corroded areas are usually red (rusty) while not damaged surfaces (without corrosion) are silver in colour. Data recording software installed on a PC by Dino and the Generative Pre-trained Transformer (GPT) (ChatGPT™) by OpenAI company were used to support the analysis of the efficiency of detecting and calculating damaged surfaces of ferrous-metal products. Moreover, Visual Studio Code™ was used to compile the Python code in the programming environment and the LibreCAD software was utilized too. LibreCAD software was used to verify the correctness of calculations performed by using GPT-based approach.

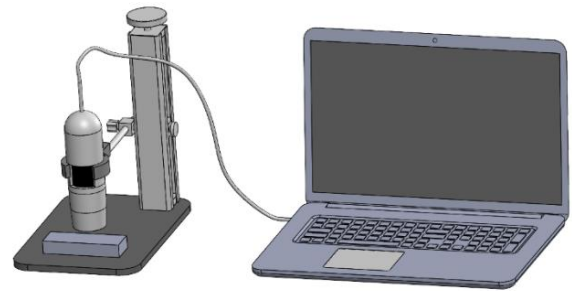


Fig. 1. Image recording test stand configuration: USB microscope and PC-software set

The details of the developed research procedure that was utilized by authors is presented in Fig. 2. It should be emphasized that this procedure utilizes software, hardware, AI assistance and human design of subsequent steps, prompts design and further analysis of obtained results.

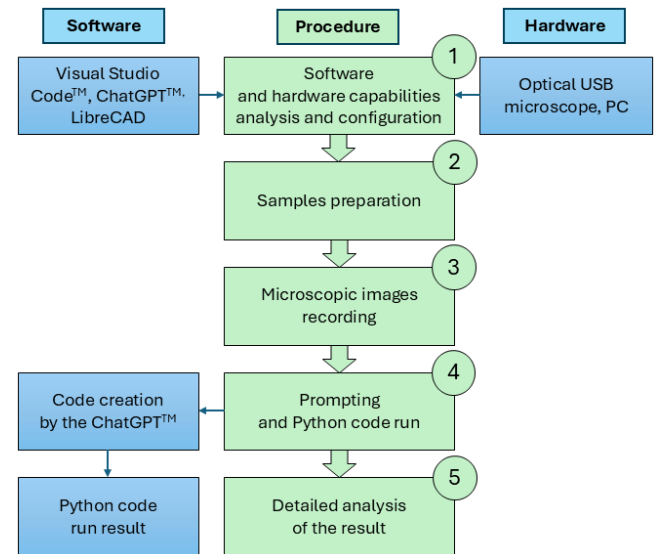


Fig. 2. Experimental procedure

Before presenting the results in this study, the authors also tested other types of damage recognition with AI-assistance, such as scratches on metal parts and temperature damage of tool holders, and decided to focus on corrosion at this stage of development. The following areas of investigation were selected for the aims of the study:

1. Measurements of corroded (rusty) area.
2. Automation of corrosion recognition.

3. RESULTS

This chapter presents research that lead to a damage detection. The authors selected three different microscopic images of ferrous-metal parts and tested a filtering methodology based on the use of artificial intelligence.

3.1. Corrosion detection

The electrochemical corrosion detection is presented below. The image captured by the microscope was linked to the Python code (based on the hard drive source indication) that enables marking the areas which are supposed to be corroded. The original images of corroded surfaces are presented in Fig. 3, Fig. 4 and Fig. 5. A ruler is added to all images presented in the study. The comparison of the original photo and a new photo generated based on the Python code (see Table 1) shows that the corroded areas are marked according to the code statements. The Visual Studio Code™ software was utilized within the code run, however, libraries required by the code and presented in the first lines (*cv2*, *numpy*, *matplotlib*) had to be installed. To recognize corroded areas authors tested different area marking approaches. The first approach was based on outlining the area with a red line (see Fig. 6, Fig. 7 and Fig. 8). The authors also tested hatching the corroded areas.

The final result was displayed in the Visual Studio Code™ as a comparison of the original image and labelled image. It can be noted that in the case of corrosion outlining, the areas damaged cannot be easily recognized if the original image is removed and outlines are left (please see an example which is presented in Fig. 8). The better result shall be reached if the corroded area is filled out with hatches (filled in) as it is presented in Fig. 10, Fig. 11, and Fig. 12.

Table 1. Exemplary prompt and exemplary Python code generated by the ChatGPT™ based on authors' prompting

Prompt: Please generate the code in the Python language that will mark in red the areas of the surface which are susceptible to electrochemical corrosion.
Code <pre>import cv2 import numpy as np import matplotlib.pyplot as plt def highlight_corrosion(image_path): # Load the image image = cv2.imread(image_path) image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert to HSV for better color segmentation hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV) # Define a range for corroded areas (e.g., rusty brown/orange) lower_rust = np.array([5, 50, 50]) # Adjust based on sample images upper_rust = np.array([25, 255, 255])</pre>

```
# Create a mask to detect rusty areas
mask = cv2.inRange(hsv, lower_rust, upper_rust)
# Apply morphological operations to refine the mask
kernel = np.ones((3, 3), np.uint8)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE,
kernel, iterations=2)
# Find contours of the detected corroded areas
contours, _ = cv2.findContours(mask,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
# Draw the contours on the original image in red
result = image_rgb.copy()
cv2.drawContours(result, contours, -1, (255, 0, 0), 2) # Red
color marking
# Display the original and processed images
fig, ax = plt.subplots(1, 2, figsize=(12, 6))
ax[0].imshow(image_rgb)
ax[0].set_title("Original Image")
ax[0].axis("off")
ax[1].imshow(result)
ax[1].set_title("Highlighted Corrosion Areas")
ax[1].axis("off")
plt.show()
# Example usage
image_path = "corroded_surface.jpg" # Replace with your image
path
highlight_corrosion(image_path)
```

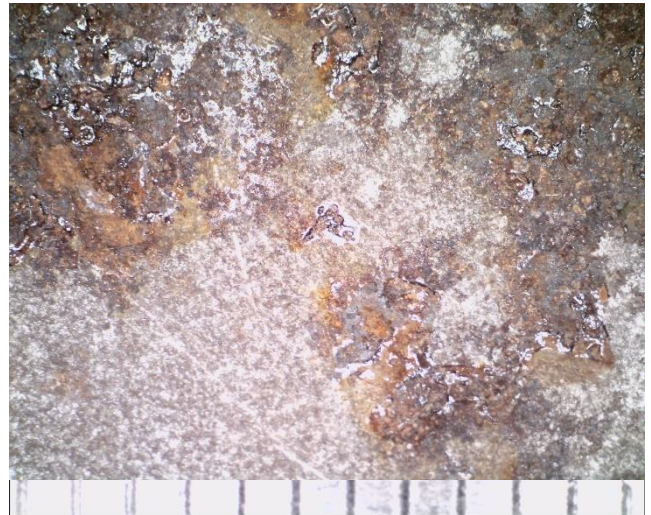


Fig. 3. Original image of the first corroded surface (distance between vertical lines is ~1 mm)

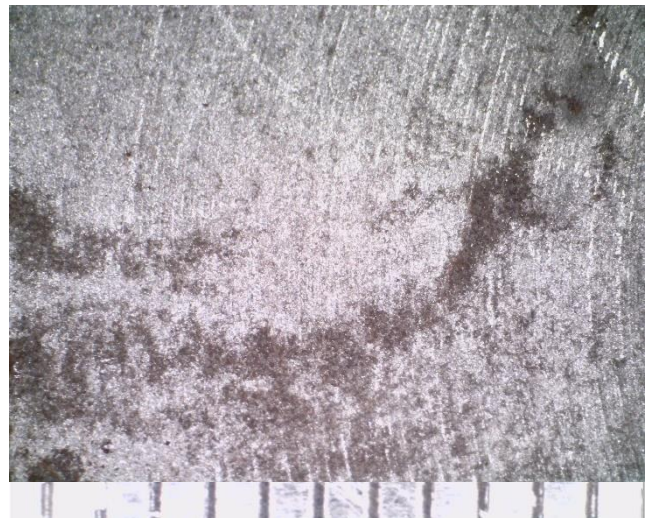


Fig. 4. Original image of the second corroded surface (distance between vertical lines is ~1 mm)

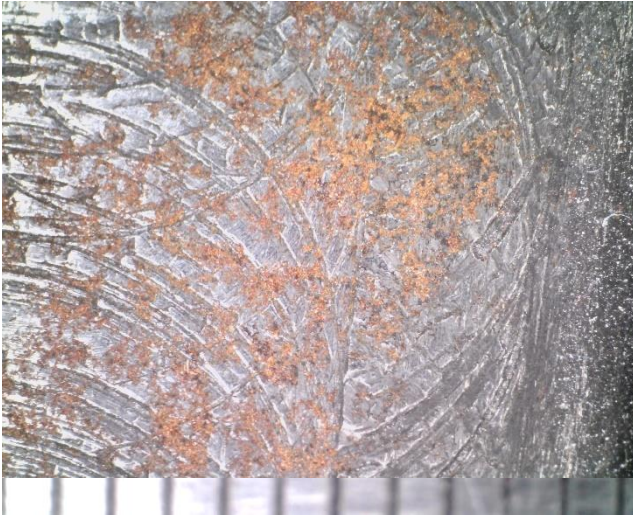


Fig. 5. Original image of the third corroded surface (distance between vertical lines is ~ 1 mm)

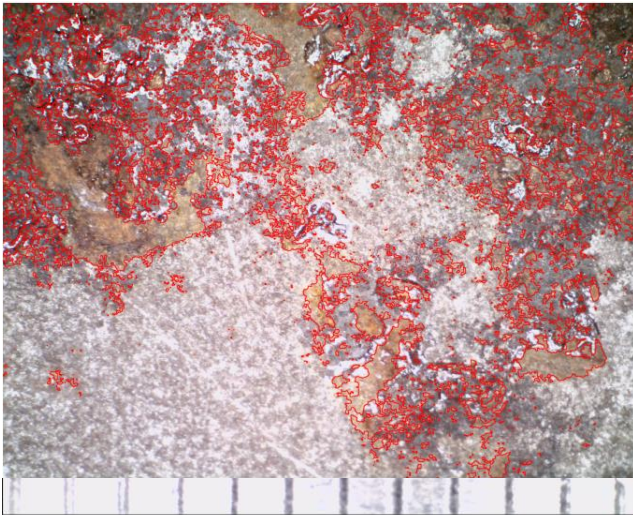


Fig. 6. Outlined corroded areas of the first corroded surface (distance between vertical lines is ~ 1 mm)

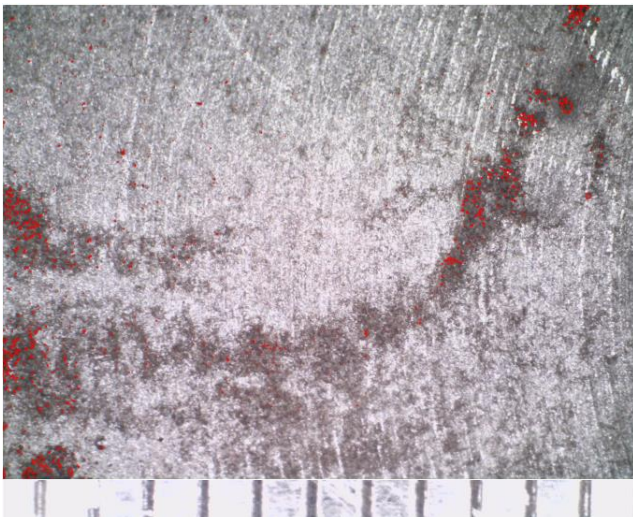


Fig. 7. Outlined corroded areas of the second corroded surface (distance between vertical lines is ~ 1 mm)

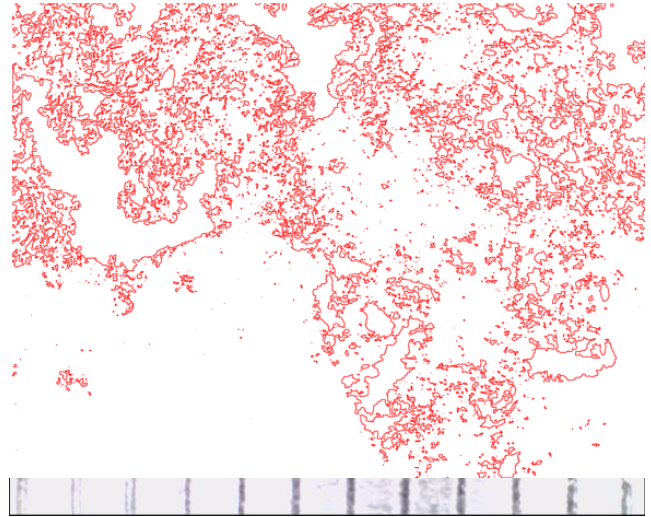


Fig. 8. Outlines of the corroded surface presented in Fig. 3 (distance between vertical lines is ~ 1 mm)

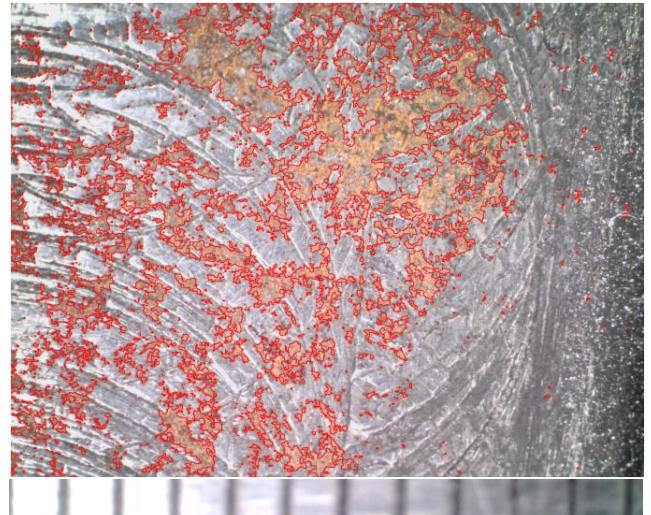


Fig. 9. Outlined corroded areas of the third corroded surface (distance between vertical lines is ~ 1 mm)

However, during the detection authors noticed that if the distance between the corrosion-damaged areas is greater, it makes them easier to be recognized after outlining. The approach which is based on filling areas in, enables easier recognition of corroded areas regardless of the distance between them. The initial analysis of the result (simple comparison) indicates that defined variation of colour (defined in the Python code) leads to the detection of corrosion-damaged areas, however, the change of settings in the code (i.e. `lower_rust = np.array([5, 50, 50])` and `upper_rust = np.array([25, 255, 255])`) changes the displayed result. This algorithm behaviour is presented in Fig. 13 where the parameters are as follow: `lower_rust = np.array([1, 1, 50])`; `upper_rust = np.array([25, 255, 255])`.

As mentioned above, a ruler was added under each analysed image and the distance between the vertical lines of the ruler equals approximate 1 mm (the width of vertical lines and copying process have slight impact on the accuracy). The ruler was recorded under the same photo shot of the original image and copied to the analysed image.

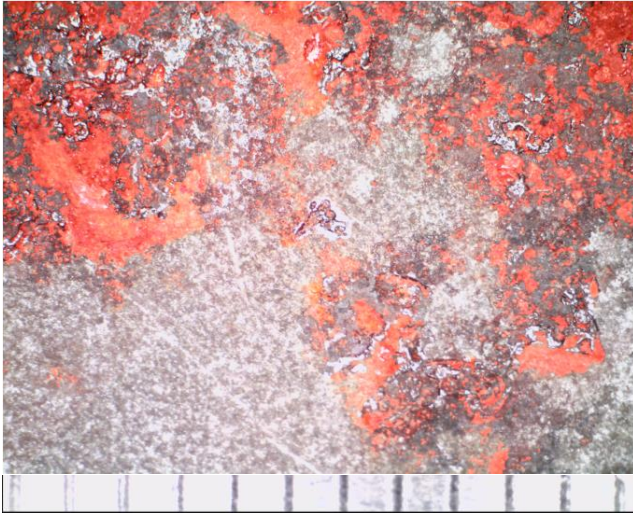


Fig. 10. Hatched corroded areas of the first corroded surface (distance between vertical lines is ~1 mm)

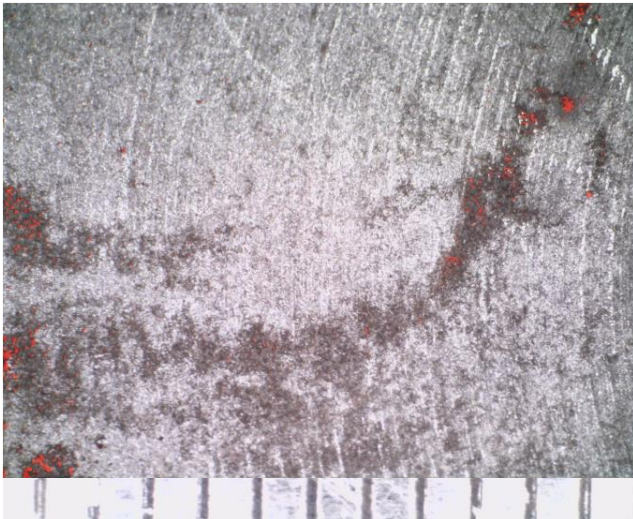


Fig. 11. Hatched corroded areas of the second corroded surface (distance between vertical lines is ~1 mm)

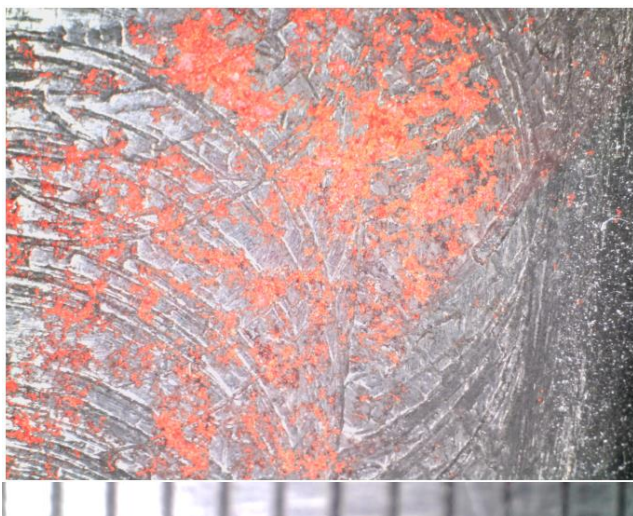


Fig. 12. Hatched corroded areas of the third corroded surface (distance between vertical lines is ~1 mm)

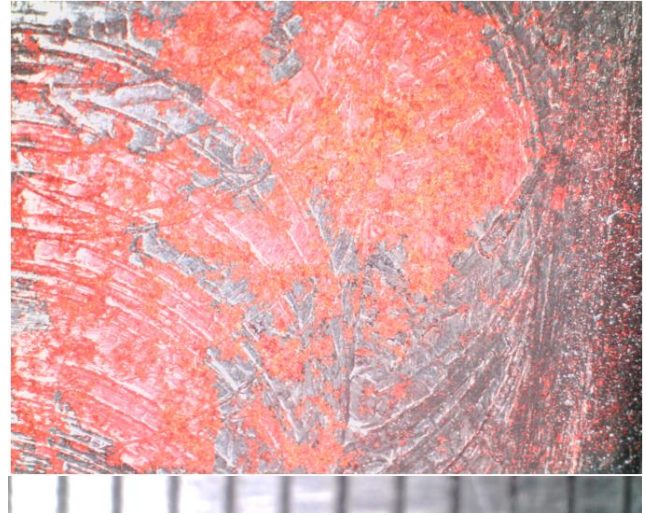


Fig. 13. Hatched corroded areas of the third corroded surface (distance between vertical lines is ~1 mm) – changed hatching parameters in the Python code

3.2. Quantitative analysis

In the next stage, the possibilities of corrosion-damaged area calculation were verified. The procedure used is presented in Fig. 14. The prompt developed by the authors requested to calculate the area damaged by corrosion as a percentage of the entire image area. This approach can be presented by the following equation:

$$A_{cs\%} = \frac{A_{cs}}{A_{es}} \times 100\%, \quad (1)$$

where A_{cs} is the corroded area; A_{es} is the area of entire image.



Fig. 14. Basic steps regarding quantitative analysis

Table 2 presents the prompts created by authors and the Python code generated by the ChatGPT™.

Table 2. Exemplary prompt, correcting prompts and exemplary Python code generated by the ChatGPT™ used for hatched area calculations

Prompt 1

Please generate the code in the Python language that will mark in red the areas of the surface which are susceptible to electrochemical corrosion.

Correcting prompt 1

change the code to hatch the corroded area!

Correcting prompt 2

update a code to calculate the hatched area

Correcting prompt 3

please calculate it as a percentage of the total area of the image

Code

```

import cv2
import numpy as np
import matplotlib.pyplot as plt
def apply_hatching(image, mask, spacing=10):
    """
  
```

```

    Applies a diagonal hatching effect on the given image in
    areas defined by the mask
    and calculates the hatched area in pixels.
    """

    # Create a copy of the original image
    hatched_image = image.copy()
    # Get contours of the mask to fill the detected regions
    contours, _ = cv2.findContours(mask,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    # Create a blank mask for drawing hatching lines
    hatching = np.zeros_like(image, dtype=np.uint8)
    # Calculate the total hatched area (sum of nonzero pixels in
    mask)
    hatched_area = np.count_nonzero(mask)
    # Loop through each detected corrosion area
    for contour in contours:
        x, y, w, h = cv2.boundingRect(contour)
        # Create diagonal hatching lines within the detected
        region
        for i in range(0, w + h, spacing):
            start_point = (x + i, y)
            end_point = (x, y + i)
            cv2.line(hatching, start_point, end_point, (255, 0, 0),
1) # Blue hatching
            # Fill the detected area with hatching
            cv2.drawContours(hatching, [contour], -1, (255, 0, 0),
thickness=cv2.FILLED)
            # Combine hatching with the original image using a bitwise
            operation
            hatched_image = cv2.addWeighted(hatched_image, 1,
hatching, 0.5, 0)
        return hatched_image, hatched_area
def highlight_corrosion_with_hatching(image_path):
    # Load the image
    image = cv2.imread(image_path)
    image_rgb = cv2.cvtColor(image,
cv2.COLOR_BGR2RGB)
    # Convert to HSV for better color segmentation
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    # Define a range for corroded areas (e.g., rusty
    brown/orange)
    lower_rust = np.array([15, 50, 50]) # Adjust based on
sample images
    upper_rust = np.array([25, 255, 255])
    # Create a mask to detect rusty areas
    mask = cv2.inRange(hsv, lower_rust, upper_rust)
    # Apply morphological operations to refine the mask
    kernel = np.ones((3, 3), np.uint8)
    mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE,
kernel, iterations=2)
    # Apply hatching effect and calculate the hatched area
    hatched_image, hatched_area = apply_hatching(image_rgb,
mask)
    # Calculate total image area
    total_area = image.shape[0] * image.shape[1]
    # Calculate corrosion percentage
    hatched_percentage = (hatched_area / total_area) * 100
    # Display the original and processed images
    fig, ax = plt.subplots(1, 2, figsize=(12, 6))
    ax[0].imshow(image_rgb)
    ax[0].set_title("Original Image")
    ax[0].axis("off")
    ax[1].imshow(hatched_image)
    ax[1].set_title(f"Hatched Corrosion Areas\n{hatched_area}
pixels ({hatched_percentage:.2f}%) of total area")
    ax[1].axis("off")
    plt.show()
    # Print the total hatched area in pixels and percentage

```

```

print(f"Total hatched corrosion area: {hatched_area}
pixels")
print(f"Percentage of image affected by corrosion:
{hatched_percentage:.2f}%")
# Example usage
image_path = "corroded_surface.jpg" # Replace with your
image path
highlight_corrosion_with_hatching(image_path)

```

The change of the lower_rust and upper_rust values change the calculation results.

Table 3 presents the result obtained after the generated Python code run.

Table 3. Exemplary results of $A_{cs\%}$ parameter measurements in the Python-based environment on the basis of Python code generated by the ChatGPT™

Figure number	$A_{ca\%}$ parameter, %
Fig. 10	21.48
Fig. 11	0.55
Fig. 12	17.89

Fig. 15 presents the area measurement technique in the LibreCAD software. It utilizes polygonal area measurement tool. The clear area outlined in pink colour was calculated and other three areas (two outlined in blue and one outlined in green) were subtracted from it. The $A_{ca\%}$ parameter was calculated in the next step as the ratio of the value obtained based on the abovementioned calculation to the measured area of the entire image (according to Eq. 1). The result of $A_{ca\%}$ is 60,89 % and this value differs from the results presented in Table 3. Recognition results strongly depend on inspector's approach to the outlined area such as outlining precision and proper recognition of colours indicating corrosion. The results may also depend on LibreCAD tools characteristics, precision-related computer mouse/pen mouse characteristics, etc. The presented values clearly demonstrate that in the case of Python-based calculation the results depend on code settings linked to the filtered colour corresponding to corrosion, however, in the case of manual outlining in the LibreCAD software they depend on the abovementioned factors.

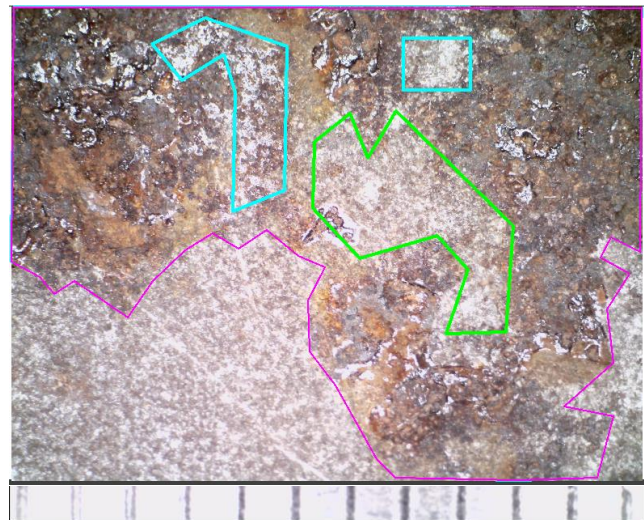


Fig. 15. Approximate calculation of the area by the use of the LibreCAD software tool for area measurement

3.3. Capabilities regarding automated damage detection

Automated recognition of damages enables to find features of microscopic images that indicate a specific damage. If photos are copied to a folder, the algorithm created as the result of prompting can help to filter them and copy the images characterized by existing evidence of the damage to another folder. The script created in Python language shall indicate both folders and it ought to describe the parameters of damages. Fig. 16 presents the abovementioned automated detection in which corroded surfaces are filtered from the set of surfaces recorded under a microscope by using colour settings defined in the code which are specific for corroded areas. The script recognizes images that present corroded surfaces and also surfaces without corrosion. Initial experiments indicate that this low-cost technique enables to separate images by less experienced Python users and it may be very useful in authors opinion to automate damage recognition in existing manufacturing environments.

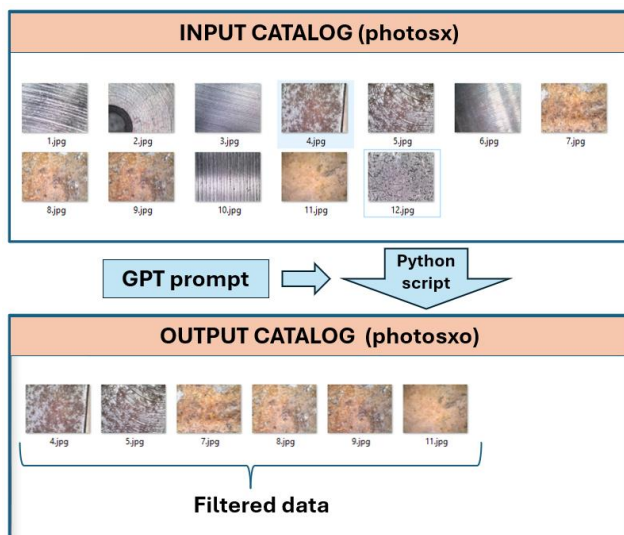


Fig. 16. Automated classification of corroded surfaces

4. DISCUSSION

Traditional inspection techniques usually do not require any damage recognition automation. Microscopic images or the results from other examination methods (e.g. penetrant, chemical, ultrasonic, etc.) are the basis for analysis and drawing final conclusions. For the scope of this study, the proposed low-cost techniques of damage analysis can simplify and automate final analysis even for corrosion-related image-based data sets.

Based on the authors' experience with generative artificial intelligence conversations, it can be tentatively stated that prompting is a key area that needs deeper understanding to obtain good results generated by the conversation. Chat responses are in fact, unpredictable. At this stage of development, corrosion recognition still requires access to high-quality microscopes or cameras and a PC (hardware), and the verification of the final results of the Python code, which is run by experienced users. However, the examples provided show that the detection

was performed, and calculations of quantitative indicators are also possible. Moreover, the final result depends on the code construction and specific values that are used for e.g. colour definition. In the case of manual outlining in the LibreCAD software they depend on characteristics of the measurement process linked to inspector's indications and software-hardware parameters.

The results of the presented tests may also be supplemented by the inspection of other mechanical surface damage. The authors performed several tests on the use of AI-assistance in the case of mechanical scratches, but the generated Python algorithm did not produce satisfactory results (e.g. scratches were not properly labelled).

According to the authors, automatic corrosion recognition is a very promising and low-cost technique that can help filter large data sets and help classify damages to make further decisions regarding part usage. However, based on existing experience with scratch recognition and the unpredictable character of chat response, authors suggest the user to verify the results generated by the chat and make the final human decisions at this stage of AI development.

5. CONCLUSIONS

The study concerned the potential application of selected AI, and Python techniques to analyse microscopic images of damages of ferrous-metals products. The results show that artificial intelligence-based methods can significantly improve the speed of damage detection, with applications across a variety of industries. Despite some challenges, including limitations related to image quality and code adaptation, this method is a promising and low-cost solution for increasing the efficiency of corrosion monitoring systems. The appropriate procedures have been presented and validated for electrochemical corrosion that can be captured using optical microscopy.

Beyond the technical improvements, the broader implications of this research are noteworthy. The integration of AI into corrosion detection systems may contribute to more efficient predictive maintenance strategies, reduced operational downtime, and improved safety in industries that rely on ferrous-metal components. With the advancement of artificial intelligence (AI) and machine learning, automated corrosion detection has become a promising alternative, offering consistent, fast, and scalable solutions. These findings suggest that AI-based methods could significantly transform traditional approaches to materials diagnostics in various industrial contexts.

Further research into expanding the application of artificial intelligence may be focused on surface geometry analysis, further scratches analysis, etc. As AI technologies continue to develop, additional applications may include real-time corrosion assessment and integration with industrial Internet of Things (IIoT) platforms for continuous monitoring.

Acknowledgments

This work was developed within the Erasmus project Teamwork-based Education and Digitalization as an Approach for the Interdisciplinary Engineering Training:

<https://erasmus-plus.ec.europa.eu/projects/search/details/2024-1-PL01-KA220-HED-000257156>, funded by the European Union.

Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the Foundation for the Development of the Education System. Neither the European Union nor the Foundation for the Development of the Education System can be held responsible for them.

Artificial Intelligence usage statement and software usage statement

Authors developed the paragraphs within the chapters and literature analysis without AI assistance. The artificial intelligence was used for Python code creation in the ChatGPT™ by the OpenAI company based on prompts written by authors. Moreover, the exemplary abovementioned code generated by the ChatGPT™ is presented in Table 1 and in Table 2. Fig. 3, Fig. 4 and Fig. 5 were recorded by authors by the use of USB microscope. Figures from Fig. 6 to Fig. 13 were developed based on the generated Python code run in the Visual Studio Code™, however, the original Fig. 3, Fig. 4 and Fig. 5 were used for this run. Fig. 15 was developed using the original image obtained under the USB microscope, and outlines were created in the LibreCAD software. Other figures were developed by authors by using CAD software (Fig. 1) or PowerPoint™ (Fig. 2) and Word™ (Fig. 14), also by using printscreen of the Microsoft Windows™ hard drive explorer (Fig. 16).

REFERENCES

1. **Wdowik, R., Świrad, S.** Application of Focus-variation Technique in the Analysis of Ceramic Chips *Materials Science Forum* 957 2019: pp. 187–194. <https://doi.org/10.4028/www.scientific.net/MSF.957.187>
2. **Xing, F., Xie, Y., Su, H., Liu, F., Yang, L.** Deep Learning in Microscopy Image Analysis: A Survey *In IEEE Transactions on Neural Networks and Learning Systems* 29 (10) 2018: pp. 4550–4568. <https://doi.org/10.1109/TNNLS.2017.2766168>
3. **Cross, S.** The Application of Fractal Geometric Analysis to Microscopic Images *Micron* 25 (1) 1994: pp. 101–113. [https://doi.org/10.1016/0968-4328\(94\)90057-4](https://doi.org/10.1016/0968-4328(94)90057-4)
4. **Long, F., Zhou, J., Peng, H.** Visualization and Analysis of 3D Microscopic Images *PLoS Computational Biology* 8 (6) 2012: pp. e1002519. <https://doi.org/10.1371/journal.pcbi.1002519>
5. **Merchant, F., Castleman, K.** Microscope Image Processing. Academic Press, 2022. eBook ISBN: 9780128210505
6. **Li, W., Field, K.G., Morgan, D.** Automated Defect Analysis in Electron Microscopic Images *NPJ Computational Materials* 4 2018: pp. 36 <https://doi.org/10.1038/s41524-018-0093-8>
7. **Xing, F., Yang, L.** Chapter 4 – Machine Learning and its Application in Microscopic Image Analysis, Editor(s): Guorong Wu, Dinggang Shen, Mert R. Sabuncu, In The Elsevier and MICCAI Society Book Series, Machine Learning and Medical Imaging, Academic Press, 2016: pp. 97–127, ISBN 9780128040768. <https://doi.org/10.1016/B978-0-12-804076-8.00004-9>
8. **Connolly, C.** Analysis of Microscopic Images *Sensor Review* 29 (1) 2009: pp. 20–27. <https://doi.org/10.1108/02602280910926724>
9. **Sanaei, N., Fatemi, A., Phan, N.** Defect Characteristics and Analysis of their Variability in Metal L-PBF Additive Manufacturing *Materials & Design* 182 2019: pp. 108091. <https://doi.org/10.1016/j.matdes.2019.108091>
10. **Satorres Martínez, S., Gómez Ortega, J., Gámez García, J., Sanchez Garcia, A.** A Machine Vision System for Defect Characterization on Transparent Parts with Non-Plane Surfaces *Machine Vision and Applications* 23 (0) 2012: pp. 1–13 <https://doi.org/10.1007/s00138-010-0281-0>
11. **García-Perez, A., Gomez-Silva, M.J., Escalera-Hueso, A.** A Generative AI Approach to Improve In-Situ Vision Tool Wear Monitoring with Scarce Data *Journal of Intelligent Manufacturing* 36 2024: 3165–3183. <https://doi.org/10.1007/s10845-024-02379-2>
12. **Xie, X.** A Review of Recent Advances in Surface Defect Detection using Texture Analysis Techniques *ELCVIA: Electronic Letters on Computer Vision and Image Analysis* 7 (3) 2008: pp. 1–22. <https://raco.cat/index.php/ELCVIA/article/view/150223>
13. **Liu, Z., Jin, L., Chen, J., Fang, Q., Ablameyko, S., Yin, Z., Xu, Y.** A Survey on Applications of Deep Learning in Microscopy Image Analysis *Computers in Biology and Medicine* 134 2021: pp. 104523 <https://doi.org/10.1016/j.combiomed.2021.104523>
14. **Chen, Y., Ding, Y., Zhao, F., Zhang, E., Wu, Z., Shao, L.** Surface Defect Detection Methods for Industrial Products: A Review *Applied Sciences* 11 2021: pp. 7657. <https://doi.org/10.3390/app11167657>
15. **Wang, M., Li, K., Zhu, X., Zhao, Y.** Detection of Surface Defects on Railway Tracks Based on Deep Learning *IEEE Access* 10 2022: 126451–126465. <https://doi.org/10.1109/ACCESS.2022.3224594>
16. **Cavaliere, G.** Comparative Use of Systems to Detect Surface Defects in Die-Cast Components Using Advanced Vision Systems Applying Artificial Intelligence. Free University of Bozen-Bolzano, 2024, PhD thesis.



© Wdowik et al. 2025 Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.