ARCHIVES
of
FOUNDRY ENGINEERING

10.24425/afe.2024.149252

Published quarterly as the organ of the Foundry Commission of the Polish Academy of Sciences

ISSN (2299-2944)
Volume 2024
Issue 1/2024

58 – 65

8/1

Image Processing Techniques for Crack Detection in MPI of Springs

M.M. Marciniak
Rzeszow University of Technology, Poland
Corresponding author. E-mail address: m.marciniak@prz.edu.pl

Received 10.11.2023; accepted in revised form 14.01.2024; available online 11.03.2024

Abstract

This study investigates image processing techniques for detecting surface cracks in spring steel components, with a focus on applications like Magnetic Particle Inspection (MPI) in industries such as railways and automotive. The research details a comprehensive methodology that covers data collection, software tools, and image processing methods. Various techniques, including Canny edge detection, Hough Transform, Gabor Filters, and Convolutional Neural Networks (CNNs), are evaluated for their effectiveness in crack detection. The study identifies the most successful methods, providing valuable insights into their performance. The paper also introduces a novel batch processing approach for efficient and automated crack detection across multiple images. The trade-offs between detection accuracy and processing speed are analyzed for the Morphological Top-hat filter and Canny edge filter methods. The Top-hat method, with thresholding after filtering, excelled in crack detection, with no false positives in tested images. The Canny edge filter, while efficient with adjusted parameters, needs further optimization for reducing false positives. In conclusion, the Top-hat method offers an efficient approach for crack detection during MPI. This research offers a foundation for developing advanced automated crack detection system, not only to spring sector but also extends to various industrial processes such as casting and forging tools and products, thereby widening the scope of applicability.

Keywords: Non-destructive testing, Magnetic particle inspection, Coil spring, Image processing, Crack detection

1. Introduction

Spring steels find applications in various industries, including railways, automotive, and machine tools. A classic spring is an elastic component designed to deform under a load and return to its original shape once the load is removed. Springs must adhere to stringent quality, environmental, and flexibility standards and ensure reliable performance over many years. A comprehensive understanding of the application and its specific criteria is essential for designing springs correctly. Expertise in materials and spring manufacturing technology is required to determine stress levels accurately and evaluate factors like creep, relaxation, and fatigue [1]. Different methodologies are employed in various areas, including the analysis of operational data at different stages of product development [2], non-destructive evaluation of samples and semi-products [3], mechanical property assessment [4], forging and rolling simulation [5], data-driven parametric analysis, and thermodynamic study [6,7].

The primary risk associated with spring products is material fatigue, resulting from errors that occur during production or operation. The appearance of defects on the material surface can lead to the formation of surface cracks, which propagate under unfavorable loading conditions, ultimately reducing the fatigue life and causing failure [1]. This relationship holds true for springs with surface faults and subsurface inclusions.

In the railway industry, preventing defects in suspension parts is of utmost importance, primarily for safety and reliability reasons. Suspension spring fractures (fig. 1) can lead to a reduction in the load on the boogie wheel, making it susceptible to flange climb. This situation can result in track collapse and wagon derailment when the wheel ends up on top of the rail (fig. 2) under unsatisfactory technical conditions like railway turnout or during curve passing [8].

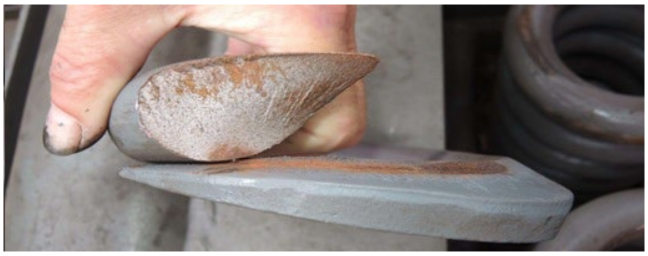
Fig. 1. Locomotive bogie spring fracture example [9]

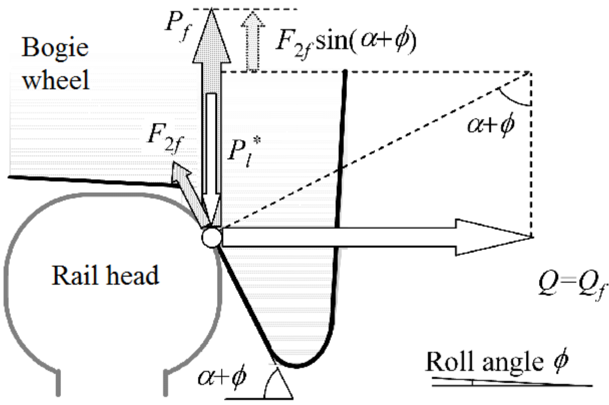

Fig. 2. Forces acting between wheel and rail during flange climbing [8]

Springs are typically manufactured from high-quality materials such as 51CrV4, 52CrMoV4, 61SiCr7 due to their specific properties and features, including high strength, relatively good resistance to fatigue, and resistance to overheating [7].

Fig. 3. Spring forming from hot rod (hot coiling).

Spring production can be divided into five steps: **Coiling:** Wire is coiled around a shaft, which can be done with either cold or heated wire (fig. 3). **Heat Treatment:** The coiled spring is heat-treated to relieve stress and harden the steel. **Shot Peening:** The spring undergoes shot peening to strengthen the steel and prevent metal fatigue.
Setting: The spring is compressed multiple times to ensure it functions correctly and remains stable at a specific length. **Coating:** A protective coating is applied to prevent corrosion which occurs either locally or throughout the section in the form of small shallow pits (pitting corrosion). These act as stress raisers under alternating stress conditions during service.

In the first stage of producing the springs, a preheated rod is used, which is oil-hardened or, in the case of the design option, annealed/unhardened. Annealed wire springs, after the coiling operation, are heat-treated (hardened, tempered) to acquire the necessary high strength. There are few steel mills operating on the market specialized enough to manufacture steel meeting the requirements of springs used in high-speed trains, which are exposed to great stress levels during operation. The long products (hot rolled billets, bars, and rods) that are made through a continuous casting route and subsequently thermo-mechanical processing seem inconsistent in meeting their quality. Surface cracks are mainly revealed during inspection after the final stage of hot rolling at the steel producer's end or before forming at the customer's site. Moreover, subsurface cracks, which are present in steel billets, may evolve after forming at the customer's site. This leads to the subsequent rejection of steel billets.

Quenching cracks are usually caused by an inadvertent use of the wrong grade of steel for a given heat treatment procedure, wrong heat coefficient of quenching medium, high hardening temperature, insufficient soaking period, surface defects such as seams, laps, clusters of non-metallic inclusions occurring at or near the surface, sharp grooves, or dents on the surface. In service, they act as stress raisers resulting in premature failure. Therefore, the evolution of surface cracks in long steel products and ready springs is one of the critical problems for producers [6].

The magnetic particle inspection (MPI) is a technique to the evidence of cracks in or close the surface of ferromagnetic materials conducted before shot peening. After heat treatment during inspection spring must be magnetized (fig. 4) [10]. The lines formed by the magnetization run parallel to the spring profile surface. Lying crosswise to created magnetic lines surface defects such a cracks generate opposite magnetic field. The effect of variable polarity on the surface discontinuity boundaries causes an accumulation of iron powder in the cavity. This makes it possible to catch even the smallest defects in the light of an ultraviolet lamp revealing the dye with which the powder is sprinkled. Then the springs without defect can be subjected to the processes of peening, setting, coating like: powder coating oiling, galvanizing, wet painting with primer paints, passivation, phosphating.

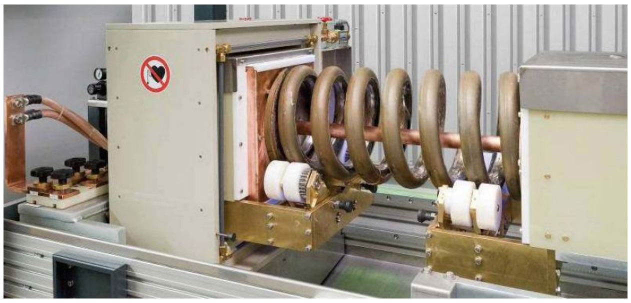
Fig. 4. MPI station (Bombardier Siegen) [10]

Traditionally, the detection of cracks in coil springs has relied on manual inspection, which can be time-consuming, laborintensive, and subject to human error. To address these challenges and enhance the efficiency and accuracy of quality control, modern manufacturing processes have increasingly turned to the integration of vision technology and image processing techniques.

In the pursuit of effective defect detection, a range of image processing techniques beyond the Canny edge detection method mentioned in the abstract has been explored. These techniques have proven to be valuable tools in identifying defects [11,12], including cracks, on the surface of steel parts. Some of these techniques include, but are not limited to:

Canny Edge Detection focuses on identifying sharp changes in pixel intensity, which can often indicate the presence of edges, including cracks, in an image. The algorithm enhances the visibility of these edges, making them easier to detect.

Hough Transform (HT) is a powerful tool for detecting straight lines, which can be crucial in identifying linear defects like cracks. It is especially effective when combined with edge detection methods. **Gabor Filters** are widely used for texture analysis in image processing. They can be applied to detect variations in texture that may be associated with surface defects like cracks. **Thresholding methods** segment an image into different regions based on pixel intensity. By setting appropriate thresholds, one can isolate areas that may contain defects, including cracks. **Deep learning techniques**, such as Convolutional Neural Networks (CNNs), have gained prominence in defect detection. They can learn and recognize complex patterns in images, making them suitable for various defect types.

These image processing techniques, when used in combination or in specific contexts, contribute to the robustness of defect detection systems. The choice of technique depends on factors such as the type of defect being targeted, image quality, and computational resources.

2. Methodology

2.1. Data collection/software

The initial phase of the work encompassed the acquisition of a comprehensive dataset consisting of spring images. These images were captured by Logitech Hd Pro Webcam C920 placed on tripod at a high resolution of 2448x3264 pixels and were stored in the PNG format. The image capture process was conducted during the Magnetic Particle Inspection (MPI) procedure using KD DEUTROFLUX UWS after rod coiling and spring heat treatment, as indicated in Figure 5. The captured images featured the spring's surface under varying conditions: some images displayed visible cracks, while others depicted the surface without any cracks. Furthermore, the imaging process involved capturing the object from various angles and under different UV lighting conditions emanating from a lamp. This was achieved while maintaining a standardized distance from the object, approximately 150 mm. Consequently, this approach yielded a diverse set of images that presented the object from multiple perspectives and exhibited variations in lighting.

In addition to this image capture procedure, the project involved the utilization of specific software and hardware tool. The author selected open-source program Python 3.12.0 and Win 10 Intel Pentium Quad Core 1,6 GHz DDR3L 4GB RAM hardware. To facilitate the execution of the code, several essential libraries were imported, including **NumPy**, a fundamental library renowned for its support of numerical operations in Python, including the management of arrays and matrices, as well as an array of mathematical functions. The image processing tasks were carried out using **OpenCV** (cv2), which played a crucial role in image manipulation and analysis. Furthermore, the **Matplotlib** plotting library was employed to create 2D and 3D visualizations, proving invaluable for data visualization and displaying images throughout various algorithmic stages, ultimately revealing the results. **Tkinter**, a standard GUI library, was also integrated into the authors' code to facilitate the creation of graphical user interfaces for desktop applications. Additionally, the **Time** library was incorporated to include the "time" module within the program. This module was instrumental in measuring the execution time of the proposed image processing solutions, providing valuable insights into the algorithm's performance.

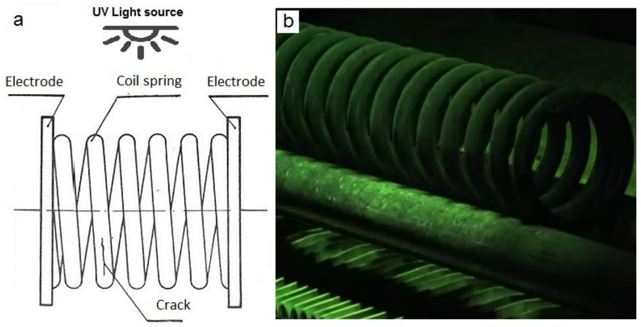

Fig. 5. MPI coil spring producer test stand a) scheme, b) view

2.2. Image processing methods

The implemented Python code adheres to a well-defined methodology that comprises several integral steps. These steps encompass image acquisition, grayscale conversion, edge detection, boundary extraction, and the superimposition of crack boundaries onto the original image. To facilitate a comprehensive understanding of the image processing results, these steps are effectively visualized using Matplotlib. This code possesses the capability to conduct in-depth analysis of cracks. Notably, the example code for image processing involving the Canny filter is detailed in the appendix A.

The process begins with the user selecting an image from the designated folder. Following this selection, the image undergoes conversion to grayscale, a fundamental step aimed at simplifying subsequent processing. This conversion of the original image is illustrated in Figure 6a. Grayscale images (fig. 6b) are characterized by the utilization of a single intensity value for each pixel, rendering them more manageable for further processing. The conversion operation is executed through the application of `cv2.cvtColor`.

During the Edge Detection step, a prominent approach is applied, on the grayscale image. Methods serves to accurately pinpoint edges and contours. The results, in the form of edges, are meticulously stored in the "Edge Image" like in Figure 7a were Prewitt filter was used.

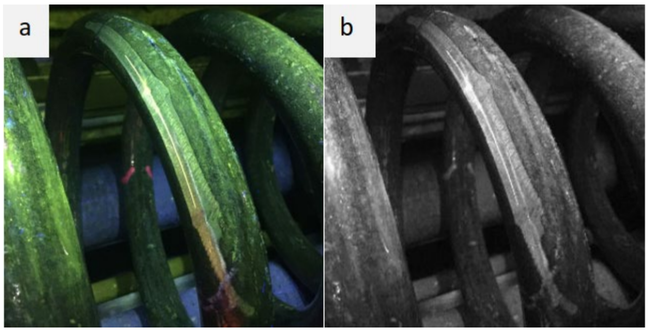

Fig. 6. a) Original image with crack on surface, b) grayscale image

The edge image obtained in the previous step serves as the basis for contour detection. Contours in this context represent the outlines of objects or shapes within the image. To identify these contours, the cv2.findContours function is employed. In this process, small contours, which may represent noise or irrelevant details, are filtered out based on a predefined minimum area threshold, defined as min_contour_area. This crucial step isolates significant features that effectively represent the boundaries of the detected cracks.
Once the contours have been successfully filtered, the code proceeds to draw them on the output_image in red color to provide a clear visualization of the “Crack boundaries” (Fig. 7b)..

These identified crack areas are distinctly marked in red, achieved either by outlining them with lines or filling them as regions. The processed images, which include the "Original Image," "Gray Image," "Edge Image," and "Crack Boundaries," are presented using Matplotlib. Each image is displayed in a separate Matplotlib window, each appropriately titled for enhanced visualization. This strategic choice enables users to make adjustments to the parameters for subsequent samples at the program's conclusion, allowing them to optimize the detection of cracks based on their findings.

The primary objective is to attain the best possible outcome, ensuring that the cracks are prominently marked and accurately detected.

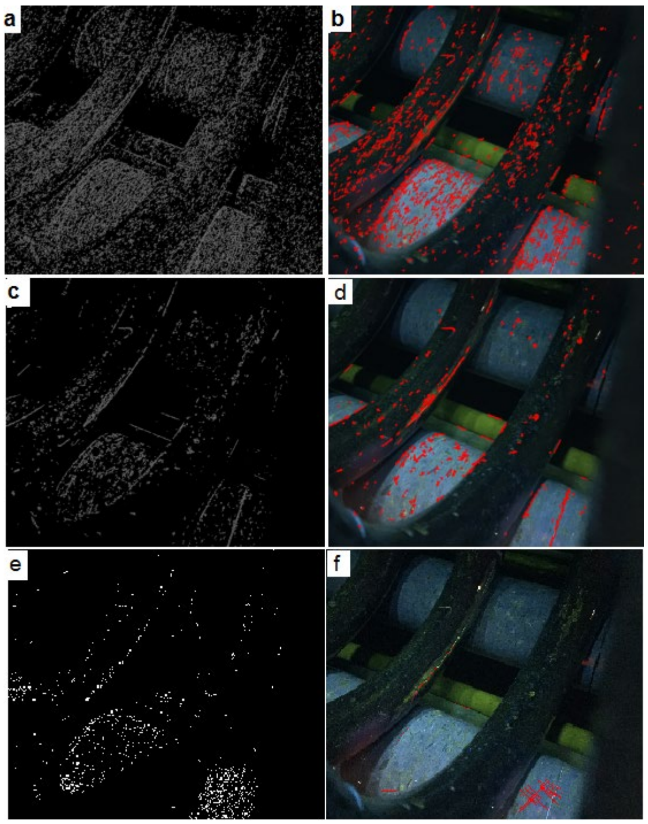

Fig. 7. Filtered grey images-"Edge Image" Filters used: a) Prewitt, c) Sobel, e) Canny, and "Crack Boundaries" in order b), d), f)

Throughout the testing phase, a diverse array of image enhancement techniques were explored to accentuate crack-like structures. This comprehensive exploration encompassed the application of various filters, like: Gaussian, Laplacian, Prewitt (fig. 7a), Gabor, and Sobel (fig. 7c), Canny (fig. 7e). Canny edge detection is widely recognized for its effectiveness in identifying abrupt shifts in intensity within the image. The key parameters for edge detection, namely threshold_low and threshold_high, are meticulously configured, taking into consideration the unique characteristics of the image under analysis. These parameters play a crucial role in the accuracy of edge detection as shown in Figure 7f influencing pointing lined outside spring contour. Additionally, reaserch involved the implementation of morphological operations such as skeletonization, morpho top-hat, and different edge-preserving techniques.

2.3. Crack detection from a set of images

The subsequent phase encompassed the development of a program with the capacity for set of image processing.

The program's initiation begins with the importation of essential libraries, followed by the user's selection of a folder containing the target images. These images are assumed to be in formats such as PNG, JPG, JPEG, or BMP.

For each image, the program validates the loading process, ensuring that the image is successfully loaded. If an image is empty or cannot be loaded, the program seamlessly proceeds to the subsequent image. Once loaded, the image undergoes conversion to grayscale.

During the Edge Detection step, a chosen approach is applied, on the grayscale image. Methods serves to accurately pinpoint edges and contours. The results, in the form of edges, are meticulously stored in the "Edge Image" or are filtered once more time for better results.

Subsequently, the program superimposes the filtered contours onto a copy of the original image in red. This visualization serves to accentuate the detected cracks. Additionally, the program conducts time measurements for each image, storing this data for future analysis. This data is invaluable for profiling the efficiency of the algorithm.

If filtered contours are present, signifying the presence of cracks, the processed image that includes the detected cracks is displayed using Matplotlib. To conclude, the program calculates and displays the total execution time for processing all images in the folder.

2.4. Method selection

In the course of this research aimed at implementing the most effective image processing methodology for detecting cracks in the MPI process, several widely employed methods were systematically examined. Upon converting images to grayscale, various filters and their respective parameters were explored, yielding images that revealed the presence of cracks. However, the outcomes of most filters proved to be unsatisfactory. Exemplary results are depicted in figure 8.

The bilateral filter, known for its edge-preserving properties in image processing, endeavors to smooth images while retaining essential edges. It is engineered to reduce image noise without compromising the integrity of edges and fine details. As illustrated in Figure 8a, it is evident that the bilateral filter tends to preserve features that share characteristics with cracks, such as coil contours or structural elements.

Skeletonization, a process that iteratively erodes the boundaries of objects until only one-pixel-wide representations remain, referred to as skeletons, aims to maintain the core structure while eliminating the object's thickness. Unfortunately, skeletonization is susceptible to image noise, manifesting as clusters of glowing pixels, which are non-crack elements. This phenomenon leads to erroneous representations, as evident in Figure 8b. Gabor filter is designed to capture specific spatial frequency information and orientation characteristics in images.

Figure 8c showcases the results following the application of the Gabor filter. However, the complex and irregular nature of cracks on a spring coil renders them ill-suited to description by simple sinusoidal waveforms, as demonstrated by the mismatch between the filter's characteristics and the actual cracks.

Two methods were revealed to have potential for true positive crack detection adopting Canny and Top-hat filter. The first method employs Canny edge filtering with specific parameters, notably threshold_low = 300 and threshold_high = 500, for effectively detecting edges in specified images. This method is supplemented by a min_contour_length = 250, as elaborated in Appendix A.

The second method involves binary image creation, image filtering, and boundary extraction, referred to as the morphology top-hat method, which is described in Appendix B.

Subsequently, the top-hat filter is applied to the grayscale image. In essence, this operation computes the disparity between the input image and its opening, where the opening operation involves erosion followed by dilation. The result of this operation is an enhancement of crack-like structures or small bright regions within the image.

A binary threshold is subsequently applied to the enhanced image, thereby segregating regions of interest from the background. In the code, a threshold of 50 is set, implying that pixel values exceeding 50 are designated as 255 (white), while values below 50 are denoted as 0 (black). The program then identifies contours within the binary thresholded image. Contours effectively delineate the boundaries of connected regions within the image. The program specifically identifies external contours while disregarding internal contours, an essential step in detecting regions that may potentially contain cracks. Contours with an area smaller than a predefined minimum contour area, herein specified as 1000, are excluded. This minimum contour area serves as a threshold that can be adjusted in accordance with the characteristics of the processed images, effectively filtering out contours that are considered noise due to their diminutive size.

3. Results

3.1 Time Execution

Time library and written code allows to measure time from the moment when user picks folder with images and until last images is completed, program also shows in the window execution time for each image. Succeeding the analysis of batch image applications using a set of 41 images (captured as described in chapter 2.1) including 10 images with visible cracks on the outer spring (200 M1342 0019 for Standard bogie UIC 517 Axle load 22.5 t) surface was used., the following conclusions were derived. Thanks to The morphological Top-hat method exhibited an average total execution time of 34.81 seconds for processing all 41 images, which is marginally longer than the execution time of the Canny edge filter method, averaging 29.12 seconds. This disparity can be attributed to the additional computational load introduced by the morphological operation, as it entails extra thresholding following the top-hat filter. In Figure 9a, we observe the results of the Canny filter, with edges converted into lines that delineate the cracks on the original image (fig. 9b). Meanwhile, the application of thresholding in the morphological method results in extended lines covering nearly the entire crack area (fig. 9c), enhancing precision, as seen in Figure 9d.

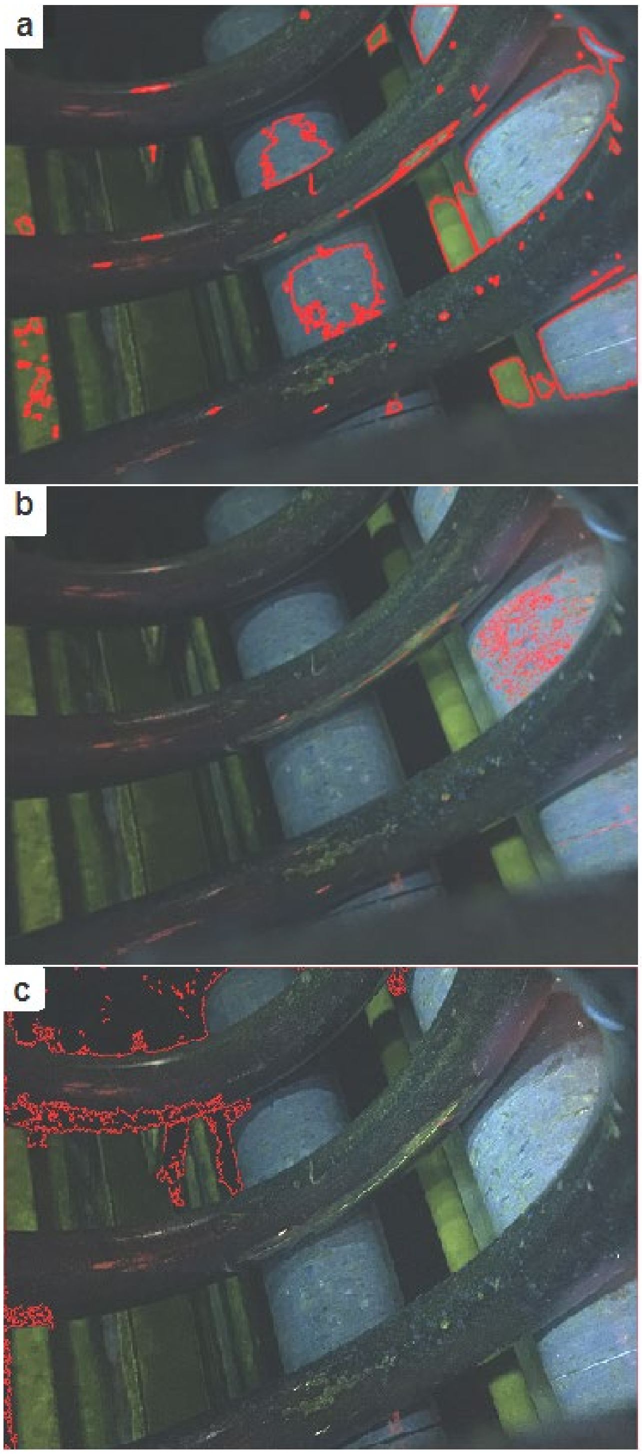
Fig. 8. Original images after crack marking:
a) Bilateral filter, b) Skeletonization, c) Gabor filter

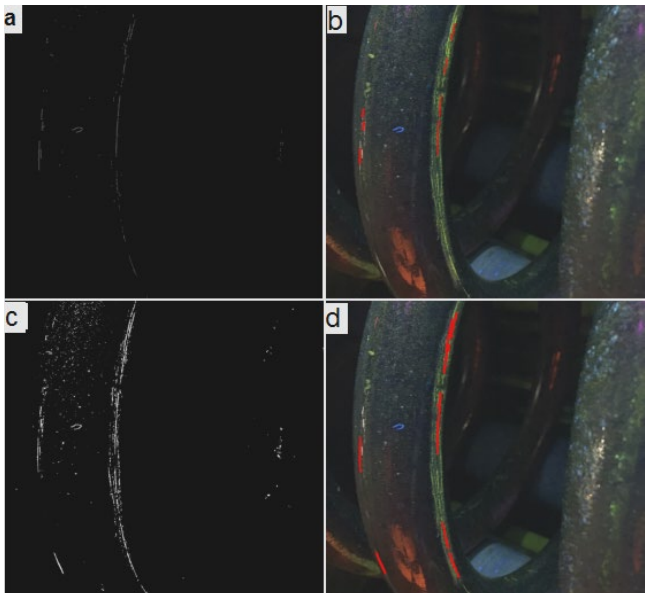
Fig. 9. a) Canny method after filter and b) draw craks, c) Morphological Top-hat after filter and threshold and d) draw cracks

3.2 Accuracy of Crack Detection

During 41 image set check test the Top-hat method identified 12 images as potentially containing cracks, of which 9 were confirmed by user to exhibit actual cracks, so three images were inaccurately labeled as containing cracks, yielding false positives (like fig. 10b). The Canny edge filter method successfully identified 10 images with cracks, while registering also three false positives (fig. 10a,c).

In the case of Canny, varying filter parameters fails to yield comparable results. Increasing the threshold parameters of the filter allows for the elimination of the undesirable effect of false edges that are not actual cracks (comparing fig. 7f and fig. 10e).

However, simultaneously, the resulting edge image can be incomplete, resulting in minimal highlighting of the crack and corresponding to only a small portion of its length as visible in the original image (fig 10 e,g) or cracks won’t be pointed on the image.

This filter experiences challenges, particularly in images with bright backgrounds where clusters of white pixels may appear. The appeared in test difficulty of establishing parameters for consistent crack detection is evident when dealing with images with pollution in the form of light-reflecting pollen (fig. 10a).

During tests using bath program new approach was made by the author. After modification in the Top-hat code (min contour area= 1300) to avoid pointing pollutions the specific false positive images (like fig. 10b) and after increasing by 10% the brightness of image (fig. 10f) which was not initially indicated by the program the Top-hat method started to successfully detect only images containing visible cracks. After changes in the code for bath processing all 10 images from 41 images with cracks, where registered with 0 false positives. Top-hat employed a different approach compared to the Canny method, offering improved accuracy, as depicted in Figure 10 f,h. This enhancement is attributable to described earlier thresholding after filtering, which broadens the edges and allows them to be superimposed on the original image.

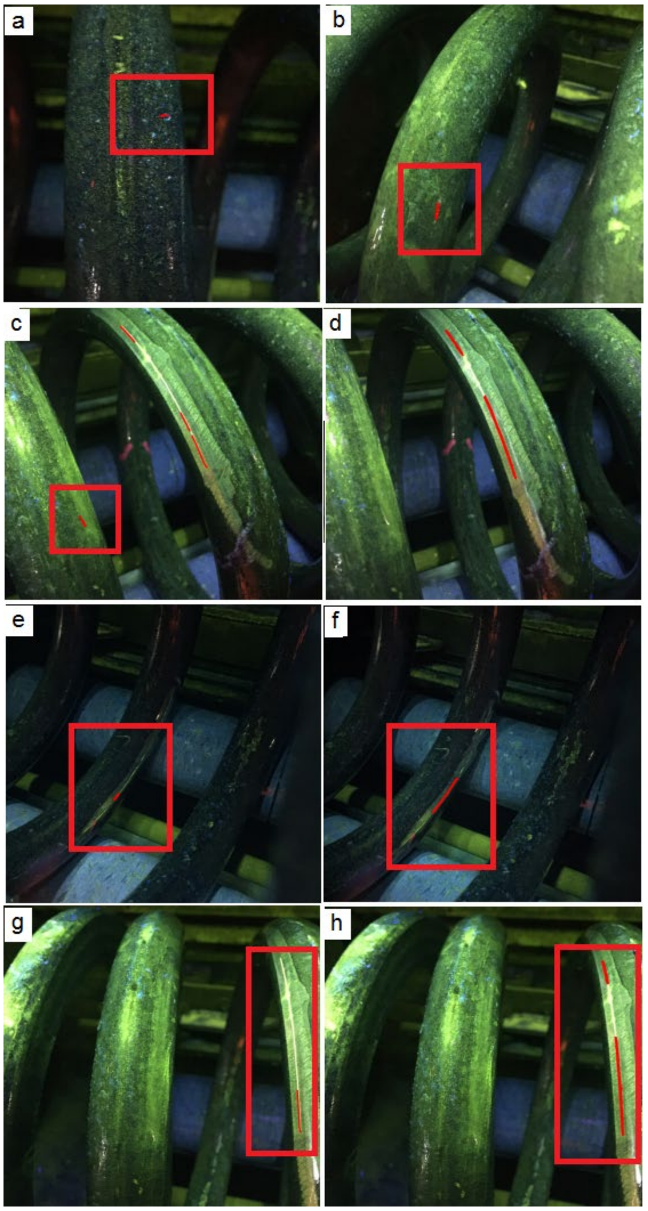

Fig. 10. Original images after crack marking:
a,c,e,f) Canny method; b,d,f,h) Morphological Top-hat

4. Conclusions

This single-image processing program, equipped with the capability to implement various methods and observe images at each stage, proved instrumental in identifying suitable parameters for the Canny and Top-hat methods. Through testing, these methods were selected from a pool of popular crack detection techniques.

Additionally, experimentation with different thresholds like showed chapter 2.2 for Canny edge detection revealed that parameters such as threshold_low = 300 and threshold_high = 500 were conducive to detecting edges effectively, particularly when combined with min_contour_length = 250. The experiments emphasized the significance of adjusting parameters, such as kernel size and minimum contour area, to align with the unique characteristics of the images at hand.

Assessing the accuracy of crack detection across the entire dataset enabled the evaluation of the method's effectiveness in identifying and delineating cracks within the selected image folder. Moreover, the batch processing capability streamlined the workflow, rendering it practical for the analysis of substantial datasets. In the next phase, these methods can be further tested on a larger sample of images, emulating the conditions of a vision system analyzing extensive datasets during standard operations. Incorporating time measurement facilitated insights into the computational efficiency of selected methodology.

While the Top-hat method exhibited potential for enhancing true positive detection it incurred a slightly longer execution time compared to the Canny edge filter method. This underlines the trade-off between accuracy and execution speed, necessitating further optimization to minimize the incidence of false positives in Canny method. In conclusion, the choice between these two methods should be contingent on the specific requirements of the application.

For the tested set of images, the Top-hat method, thanks to its enhanced accuracy in crack proper indication, constitutes a valid and efficient approach for detecting cracks in images captured during MPI. Future work may explore the development of more advanced techniques for automated crack detection and analysis.

References

[1] Gubeljak, N., Predan, J., Senčič, B. & Chapetti, M. (2014). Effect of residual stresses and inclusion size on fatigue resistance of parabolic steel springs. *Materials Testing.* 56(4), 312-317. DOI:10.3139/120.110567.

[2] Xu, C., Yilong L., Ming Y., Jiabang Y. & Xiang P. (2021). Effects of the ultra-sonic assisted surface rolling process on the fatigue crack initiation position distribution and fatigue life of 51CrV4 spring steel. *Materials.* 14(10), 2565, 1-19. DOI:10.3390/ma14102565.

[3] Yun, J.P., Choi, Dc., Jeon, Yj. et al., (2014). Defect inspection system for steel wire rods produced by hot rolling process. *The International Journal of Advanced Manufacturing Technology.* 70, 1625-1634. DOI:10.1007/s00170-013-5397-8.

[4] Perichiyappan, S. & Jagadeesha, T. (2021). Modelling and simulation of primary suspension springs used in Indian railways. *Materials Today: Proceedings.* 46(17), 8450-8454. DOI: 10.1016/j.matpr.2021.03.478.

[5] Kumar, S., Kumar, V., Nandi, R.K. et al. (2008). Investigation into surface defects arising in hot-rolled SUP 11A grade spring billets. *Journal of Failure Analysis and Prevention.* 8(6), 492-497. DOI:10.1007/s11668-008-9169-y.

[6] Filipović, M., Eriksson, C. & Överstam, H. (2006). Behaviour of surface defects in wire rod rolling. *Steel research international.* 77(6), 439-444, DOI:10.1002/srin.200606411.

[7] Matjeke, V.J., Van Der Merwe, J.W., Mukwevho, G. & Phasha, M.J. (2019). Thermal characteristics of spring steels used in railway bogies. *SN Applied Sciences*. 1, 1548, 1-8. DOI:10.1007/s42452-019-1546-5.

[8] Nagumo, Y., Tanifuji, K. & Imai, J. (2010). A basic study on wheel flange climbing using model wheelset. *International Journal of Railway*. 3(2), 60-67. DOI:10.1299/kikaic.74.242.

[9] The Rail Safety Inspection Office. (2021). *Accident and incident investigation report: Derailment of the regional passenger train No. 21209 between Chvalkov and Vcelnicka operating control points.* Retrieved November 7, 2023, from https://www.dicr.cz/files/uploads/Zpravy/MU/DI_Chvalkov_Vcelnicka_210715.pdf.

[10] Maass, M., Deutsch, W.A., Bartholomai, F. (2014). Magnetic Particle Inspection on train components. In 11th European Conference on Non-Destructive Testing, 6-11 October 2014 (pp. 1-9). Prague, Czech Republic.

[11] Deng, J., Singh, A., Zhou, Y., Lu, Y. & Lee, V.C.S. (2022). Review on computer vision-based crack detection and quantification methodologies for civil structures. *Construction and Building Materials.* 356, 129238. DOI:10.1016/j.conbuildmat.2022.129238.

[12] Mohan, A. & Poobal, S. (2018). Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal.* 57(2), 787-798. DOI:10.1016/j.aej.2017.01.020.

Appendix A

```
gray_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2GRAY)
  plt.figure()
  plt.title(f"Grayscale Image: {image_file}")
  plt.imshow(gray_image, cmap='gray')
  plt.axis("off")
  threshold_low = 300
  threshold_high = 500
  edges = cv2.Canny(gray_image, threshold_low, threshold_high)
  plt.figure()
  plt.title(f"Canny Edge Detection: {image_file}")
  plt.imshow(edges, cmap='gray')
  plt.axis("off")
  output_image = original_image.copy()
  contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
  min_contour_length = 250
  filtered_contours = [contour for contour in contours if
cv2.arcLength(contour, closed=True) > min_contour_length]
  # Draw the filtered contours on the output image
  cv2.drawContours(output_image, filtered_contours, -1, (0, 0, 255), 10)
```

Appendix B

```
  gray_image = cv2.cvtColor(original_image,
cv2.COLOR_BGR2GRAY)
  plt.figure()
  plt.title(f"Grayscale Image: {image_file}")
  plt.imshow(gray_image, cmap='gray')
  plt.axis("off")
  kernel_size = 15
  kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (kernel_size,
kernel_size))
  top_hat = cv2.morphologyEx(gray_image, cv2.MORPH_TOPHAT,
kernel)
  plt.figure()
  plt.title(f"Top-Hat Filtered Image: {image_file}")
  plt.imshow(top_hat, cmap='gray')
  plt.axis("off")
  _, binary_image = cv2.threshold(top_hat, 50, 255,
cv2.THRESH_BINARY)
  plt.figure()
  plt.title(f"Thresholded Image: {image_file}")
  plt.imshow(binary_image, cmap='gray')
  plt.axis("off")
  contours, _ = cv2.findContours(binary_image,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  min_contour_area = 1000
  filtered_contours = [contour for contour in contours if
cv2.contourArea(contour) > min_contour_area]
  output_image = original_image.copy()
  cv2.drawContours(output_image, filtered_contours, -1, (0, 0, 255),
thickness=10)
```