



Progressive Automatic Method for Annotation of Concrete Crack Images

Donghui Xie^{1,a}, Dandan Shi^{2,b}, Qingning Chen^{1,c,*}

¹Ningbo Yuanshui Co., Ltd., Ningbo, China

²Ningbo Reservoir Management Center, Ningbo, China

^a1139381698@qq.com, ^b1371003097@qq.com,

^cseventyoranges@163.com

Abstract. Cracks are the most common manifestation of diseases in concrete dams. For dam cracks, many projects still use traditional measurement methods for detection, which are inefficient and subjective. To improve the accuracy and efficiency of crack detection, this paper presents a progressive automatic annotation algorithm that uses a three-stage process to annotate sample images of cracks. Firstly, draw black lines to simulate cracks on white paper, followed by application of edge detection to find crack contours. Secondly, the detected crack contours and sample information are integrated to generate an annotation file for training, thus obtaining the first-order weight file. Thirdly, calculate the Euclidean distance between the background area and the RGB components of the pixels in the detection area to optimize the mask and extract the crack coordinates. An 8-neighbor mask and the shared number are used at each coordinate point to systematically extract crack contours. And the crack sample information is integrated to automatically generate the image annotations for training, thus obtaining the second-order weight file for batch detection of concrete cracks. Finally, after optimizing and extracting mask contours, crack sample information is integrated to generate the image annotations for training and the subsequent batch detection, thereby, producing the third-level weight file. When used for the classification of images of black lines on white paper, black cracks on white rendered concrete, and cracks in concrete, the trained Mask region-based convolutional neural network (RCNN) model had a comprehensive evaluation index of 95.2%, 83.3%, and 79.2% respectively. The high detection rate shows that this model can be used effectively for fast detection of cracks in concrete structures.

Keywords: Cracks; automatic annotation; progressive; optimization; contour extraction; deep learning

1 Introduction

Deep learning techniques have been successfully implemented in crack detection, and many important contributions have been made in this rapidly developing field. Zhang et al^[1]. used small patches extracted from pavement images to differentiate crack pixels

© The Author(s) 2025

Y. Zhang et al. (eds.), *Proceedings of the 2024 10th International Conference on Architectural, Civil and Hydraulic Engineering (ICACHE 2024)*, Advances in Engineering Research 259,

https://doi.org/10.2991/978-94-6463-658-1_52

from background pixels. Zeiler and Fergus^[2] constructed a fast region-based convolutional neural network (RCNN), which required only 0.03 s per image to detect cracks in concrete. Yokoyama and Matsumoto^[3] classified training samples into five categories, namely, cracks, chalk writing, joints, surface, and other parts and developed a neural-network based detector with a consistent classification accuracy rate of approximately 79%.

The accuracy of crack measurements depends on the quality and quantity of image samples, but using a large number of samples requires a lot of time and energy to complete the labelling process. Xu et al^[4]. proposed a hybrid attention-based model for automatic image-caption generation based on the convolutional neural network (CNN) and recurrent neural network (RNN) models; the proposed model could detect and select objects correctly, but the selection frame contained a large amount of background information. As a result, the detected object was not segmented very accurately. Maier-Hein et al^[5]. carried out the annotation of endoscopic images based on the concept of crowd-algorithm collaboration; the proposed method is well suited in medical image annotation, but its universality has not yet been established. Zhou et al^[6]. used a large number of unlabelled medical images to establish a set of self-supervised learning models, known as Models Genesis. Although Models Genesis outperforms conventional models, it also learns a lot of redundant information related to image background.

Current research in the field of automatic image annotation focuses on improving the accuracy of object detection, while speeding up the sample annotation process^[6-19]. However, these two goals cannot be well balanced. In this study, we propose a progressive learning method for fully automated image annotation. The proposed method combines morphological techniques with deep learning technology to allow the learning process to gain more traction. Images of black lines on white background, black cracks on white rendered concrete, and cracks in concrete are used for crack detection, and the detection results are optimized and converted into image annotations. The training process is repeated iteratively until accurate segmentation of the crack image is achieved, which is fully automated, reducing the subjectivity of human factors while improving computational efficiency and accuracy.

2 Model Training Process

The training process is as follows.

Stage 1: Black lines, simulating cracks, are drawn on white paper, which is then cut into fixed-size samples. Internal voids (points located within crack contours) are eliminated during image preprocessing using morphological operations of binarisation, corrosion, and expansion, followed by application of edge detection to find crack contours. Next, the detected crack contours and sample information are integrated to generate an annotation file for training, thus obtaining the first-order weight file.

Stage 2: The first-order weight file is used to detect cracks in images, in which concrete is rendered in white. Next, the Euclidean distance between the background area and the RGB components of the pixels in the detection area is calculated and used to optimize the mask and extract the crack coordinates. An 8-neighbour mask and the

shared number are used at each coordinate point to systematically extract crack contours. Finally, the crack sample information is integrated to automatically generate the image annotations for training, thus, obtaining the second-order weight file.

Stage 3: The style transfer technique is used to combine the white rendered concrete and regular concrete images. The base64 code in the original annotation file is modified for training, and the upgraded second-level weight file is used for batch detection of concrete cracks. After optimizing and extracting mask contours, crack sample information is integrated to generate the image annotations for training and the subsequent batch detection, thereby, obtaining the third-level weight file.

The process of progressive self-training algorithm is shown in Figure 1.

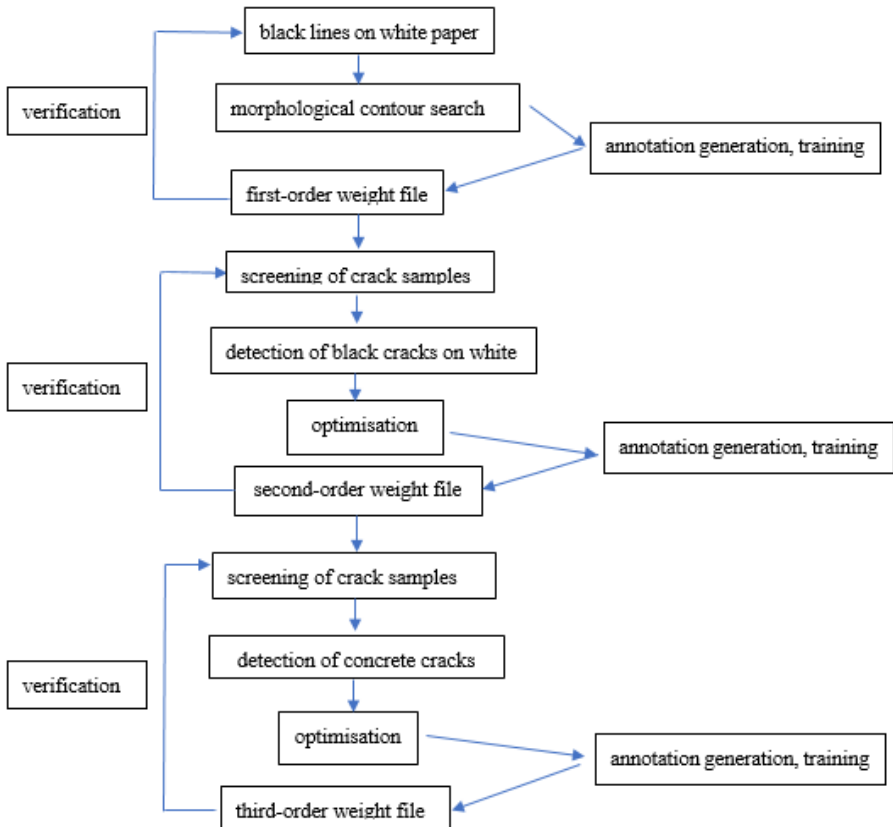


Fig. 1. Flowchart of the progressive self-training algorithm^[20]

3 Annotation Generation Method

Generally, a weight file generated in one stage is used in the following stage to detect cracks and generate mask coordinates. After optimization, the mask information is

transformed into a new JSON file through a series of operations: first, the contour coordinates in the detection region are extracted and arranged sequentially; next, the relevant crack image information is given as an input to the JSON file.

3.1 Mask Contour Extraction

The mask is composed of a series of coordinate points. The mask contour extraction is essentially an analysis of a series of coordinate points to eliminate any point located between the crack contours and retain only the contour points.

The void number of a given coordinate point is the number of pixels among its 8 neighbour cells unoccupied by other coordinate points. In Figure 2, there are 3 unoccupied pixels around contour point P1, so its void number is 3 whereas, all 8 neighbour pixels of contour point P2 are occupied, so its void number is 0.

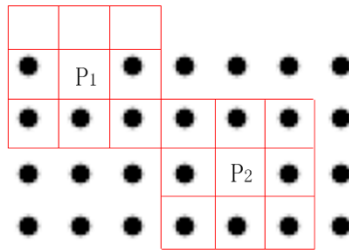


Fig. 2. Determination of the void number

The shared number is the number of unoccupied pixels shared by two adjacent mask points. In Figure 3, points P3 and P4 share 2 common void pixels (shown in green), thus, the shared number is 2.

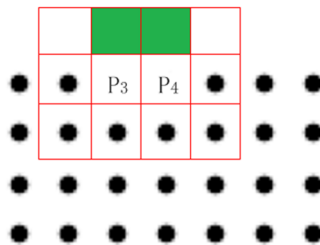


Fig. 3. Determination of the shared number

Contour extraction is carried out based on the following principles:

1. Before contour detection, go over the mask points and delete any scattered point, that is, the point with shared number exceeding 6 out of 8 (Figure 4).

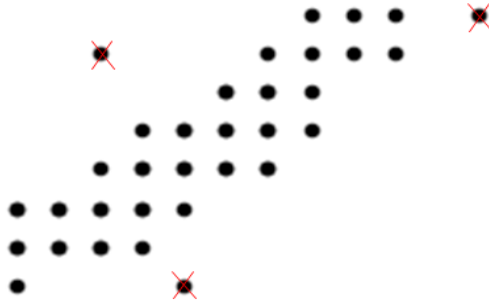


Fig. 4. Elimination of scattered points

- Starting point A of the contour is selected by finding the point with the highest void number and the smallest x and y coordinates (Figure 5).

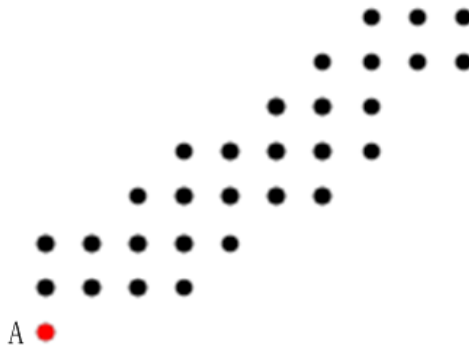


Fig. 5. Selection of the contour starting point

- Among the points adjacent to P, find the one with the highest void number and set it as the next contour point, ensuring that the new point is not the same as the current contour point (Figure 6).

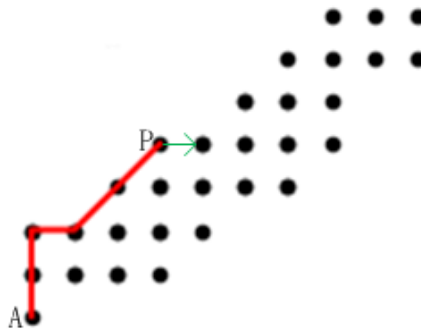


Fig. 6. Contour point selection

7. Find the coordinate points in the original file and draw the contour, until all coordinate points are deleted (Figure 10).

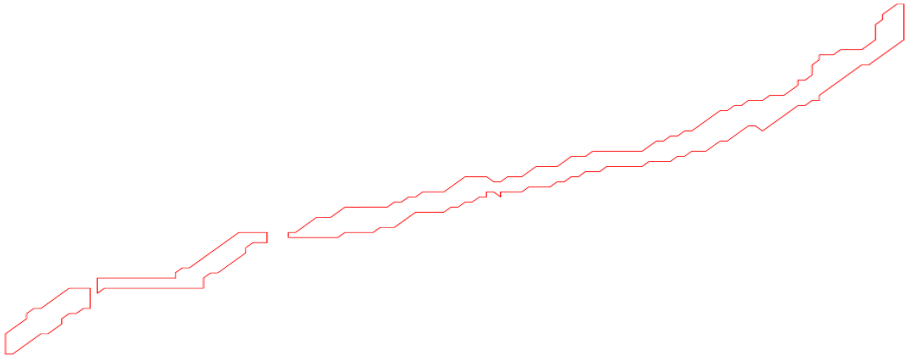


Fig. 10. Newly detected crack contour

3.2 JSON File Generation

The number of closed contours is counted to establish a fixed framework for the annotations. The file is divided into five sections: LabelMe version information, contour coordinates, image path, encoding, and size.

The adjusted contour coordinates, the image path, and its encoding format are imported into the JSON file based on the fixed annotation framework. The image is then loaded into a binary format, and its raw bytecode is compiled and encoded into base64 format. The obtained data is then inserted into the corresponding position of the JSON file, and the complete JSON file is provided as an input to the Mask RCNN model. The training process is continued using the previous-order weight file to generate a new weight file.

4 Training Process

4.1 Black Lines on White Paper

A 12B pencil is used to draw black lines resembling cracks on white paper, which are photographed, as shown in Figure 11(a). The original image is cut into 512×512 tiles (Figure 11(b)), so that 20,000 image tiles are obtained. The subsequent image processing consists of image binarisation and pixel corrosion and expansion (Figure 11(c)), followed by crack contour detection (Figure 11(d)). Each image tile and its corresponding JSON file are given as inputs to the Mask RCNN model for training to obtain the first-order weight file.

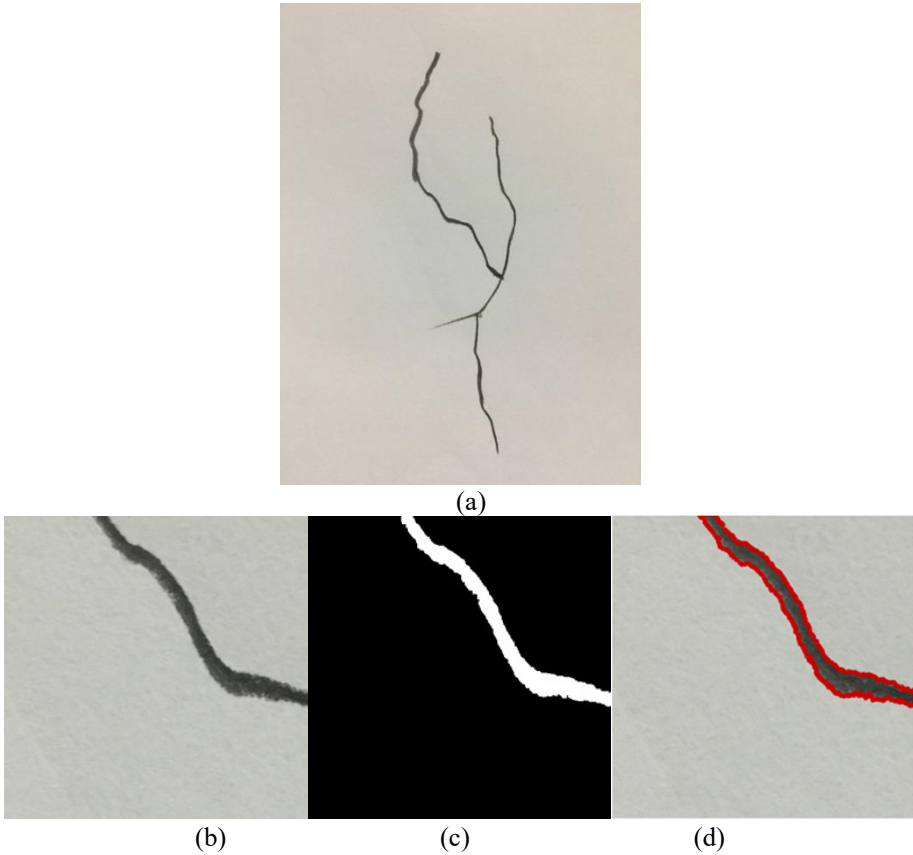


Fig. 11. Processing of black lines on white paper images

4.2 Black Cracks on White Rendered Concrete

The appearances of black lines on white paper and black cracks on white rendered concrete are similar. The test results show that the model can easily transition between the two types of samples; the first-order weight file can be used effectively to filter and detect cracks in white rendered concrete. The optimized mask coordinates are converted into a JSON file using the annotation algorithm, and the file is given as an input to the model for further training, thereby, obtaining the second-order weight file.

4.2.1 Screening of Annotation Results.

During the crack detection process, false results often occur in shaded or blistered areas of the concrete. The following principles are used to eliminate false detection results:

1. Delete any detection result with a confidence level < 0.95 (Figure 12).

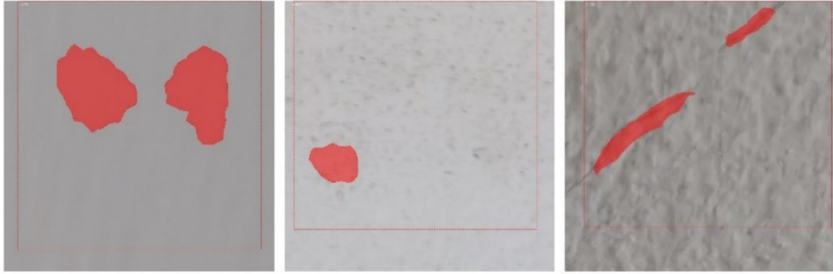


Fig. 12. Image elimination based on confidence level

2. Delete any image with a detected mask containing less than 750 points (Figure 13).



Fig. 13. Image elimination based on the number of mask points

3. Rotate the coordinate system and project the mask coordinates on the x and y axes, respectively, and then calculate the ratio L of the average number of x and y coordinates using the following formula:

$$L = \frac{k \sum_{i=1}^j n_{x_i}}{j \sum_{i=1}^k n_{y_i}} \tag{1}$$

(Note: n_{x_i} represents the number of coordinates when $x=x_i$, and n_{y_i} represents the number of coordinates when $y=y_i$)

If L is close to 1, the cluster of points detected erroneously will be eliminated (Figure 14).

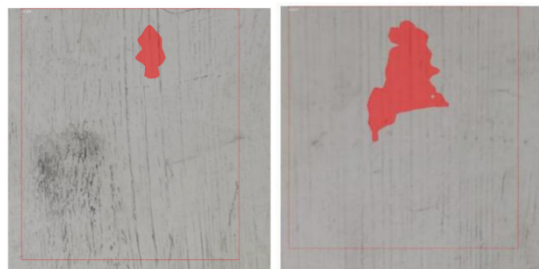


Fig. 14. Image elimination based on the mean ratio

4. Determine the propagation direction of the crack, and eliminate any crack with a width greater than 100 pixels (Figure 15).



Fig. 15. Image elimination based on the crack width anomaly

5. After converting the image to grayscale, calculate the difference C between the classification result and the average value of the background pixels using the following formula:

$$C = \frac{\sum_{i=1}^m x_i}{m} - \frac{\sum_{i=1}^n y_i}{n} \quad (2)$$

(Note: m is the number of background pixels, n is the number of points classified as crack pixels, x_i is the pixel value of the i -th background pixel, and y_i is the pixel value of the detected i -th pixel)

When C is in the range of $[-10, 10]$, find and eliminate the detection errors and cracks with very light colour (Figure 16).

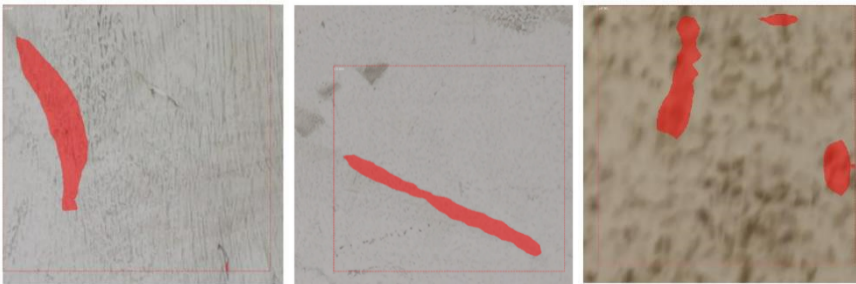
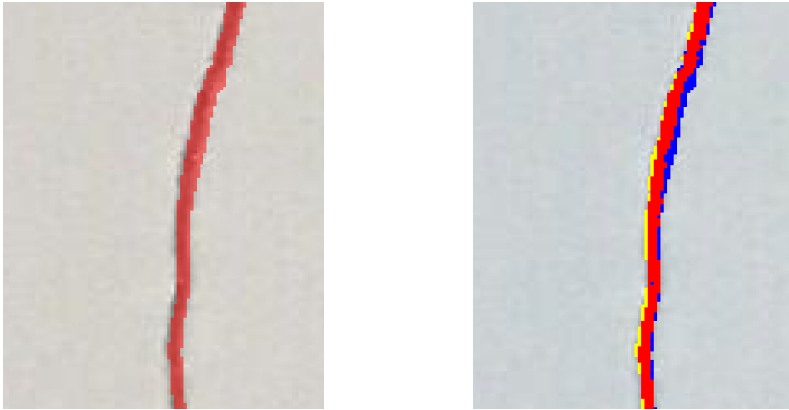


Fig. 16. Image elimination based on the pixel difference value

4.2.2 Optimization Algorithm.

To obtain more accurate crack mask coordinates, the deep learning-based classification results (Figure 17(a)) are optimized with the aid of the RGB components of the raw image. The classification results are divided into three types: accurate (red), erroneous (blue), and missed (yellow), as shown in Figure 17(b).



(a) Classification results (b) Comparison of accurate, erroneous, and missed detection areas

Fig. 17. optimization results

All the pixels in the rectangular frame are categorized as belonging to either the crack area or the background area. The average RGB values of the two types of areas are used as the discrimination indices in the subsequent processing. The average RGB values of the crack pixels are given by the following:

$$\overline{R}_c = \frac{1}{NUM_c} \sum_{i=1}^{NUM_c} R_{ci} \tag{3}$$

$$\overline{G}_c = \frac{1}{NUM_c} \sum_{i=1}^{NUM_c} G_{ci} \tag{4}$$

$$\overline{B}_c = \frac{1}{NUM_c} \sum_{i=1}^{NUM_c} B_{ci} \tag{5}$$

where $\overline{R}_c, \overline{G}_c$ and \overline{B}_c are the average RGB component values, respectively, of the pixels in the crack area; NUM_c is the total number of pixels in the crack area; and, R_{ci}, G_{ci} and B_{ci} are the colour components of the i -th pixel in the crack area. The subscripts b and c indicate the background and crack pixels, respectively. The average RGB values of the pixels in the background area can be obtained in a similar way.

4.2.3 Screening of the Mask Area.

Traverse all pixels in the mask area and calculate the Euclidean distances between the RGB values of each pixel and the average RGB values of the crack area and between the RGB values of each pixel and the average RGB values of the background area, using the following formulas:

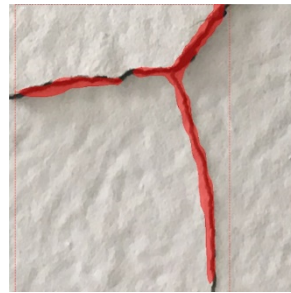
$$D_c = \sqrt{(R_0 - \overline{R}_c)^2 + (G_0 - \overline{G}_c)^2 + (B_0 - \overline{B}_c)^2} \tag{6}$$

$$D_b = \sqrt{(R_0 - \overline{R}_b)^2 + (G_0 - \overline{G}_b)^2 + (B_0 - \overline{B}_b)^2} \tag{7}$$

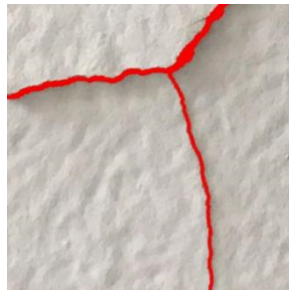
D_c and D_b represent the Euclidean distances between the RGB values of the selected pixel and the average RGB values of the crack and the background areas, respectively; and B_0 represent the RGB components, respectively, of the selected pixel. If, the Euclidean distance between the selected point and the average pixel value of the background area is smaller than that between the selected point and the average pixel value of the crack area, which means that the RGB value of this point is closer to that of the background area; therefore, the selected point is eliminated; otherwise, this point is retained. After all the mask points are traversed, a new set of mask coordinates is obtained.

4.2.4 Expansion of the Mask Area.

The mask points are expanded outwards along the mask boundary, and the newly acquired pixels are screened using the method described in the previous section. After each screening and expansion, the average RGB values of the mask area and the background area are recalculated. When the mask coordinates before and after the iteration are consistent, it is considered that the screening and expansion in this iteration have not modified the mask area. Thus, the iteration process is stopped, and the final optimized crack mask is obtained. The original image and the images before and after the mask optimization are shown in Figure 18.



(a) Black cracks on white rendered concrete (b) Crack mask before optimization



(c) Crack mask after optimization

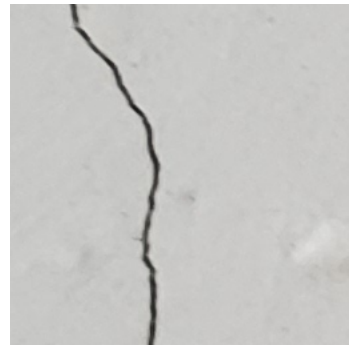
Fig. 18. Image optimization process

4.3 Concrete Cracks

Firstly, images of plain concrete are cut into 512×512 tiles, and from these, 2,000 crack-free background images (Figure 19(a)) are selected. Secondly, the second-order weight file is used to detect black cracks on white rendered concrete (Figure 19(b)) and obtain the crack mask coordinates, which are then plotted on the crack-free background images. Finally, the original JSON file of the black cracks on the white rendered concrete is modified: the image code is replaced, and the image path and annotation file name are adjusted; the contour coordinates remain unchanged. The newly generated image is shown in Figure 19(c).



(a) Concrete background image



(b) Black cracks on white rendered concrete



(c) Cracks on concrete background (using style transfer technique)

Fig. 19. Image generation process

After transitioning from the black cracks on white rendered concrete to the cracks in concrete successfully, the preceding stage is repeated, starting with the sample screening, followed by crack detection (Figure 20(a)), mask optimization (Figure 20(b)), extraction of contours (Figure 20(c)), image information generation, and further training.

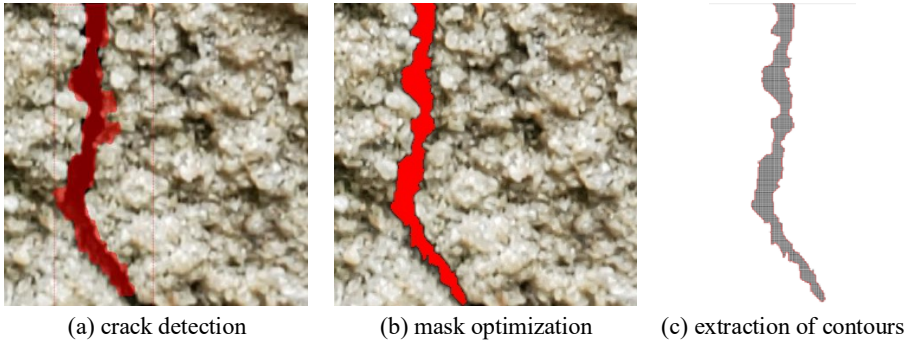


Fig. 20. Image optimization process

5 Evaluation

The data sets of black lines on white paper, black cracks on white walls, and cracks on concrete are divided into three types. The training set consists of 80% images, and the verification set consists of 20% images. According to the results in Tables 1 and 2, it can be seen that this method has good accuracy for crack identification.

Table 1. Testing results (average value)^[21]

Stage	Accuracy	Recall	Evaluating indicator
Black lines on white paper	95.502	92.729	94.055
Black cracks on white rendered concrete	81.025	79.178	76.294
Concrete cracks	78.003	79.500	76.675

Table 2. Testing results (median)^{[21],[22]}

Stage	Accuracy	Recall	Evaluating indicator
Black lines on white paper	96.7	93.6	95.2
Black cracks on white rendered concrete	86.1	85.1	83.3
Concrete cracks	77.8	83.1	79.2

6 Conclusion

This paper presents a progressive automatic annotation algorithm that uses a three-stage process to annotate sample images of cracks. The high detection rate shows that this model can be used effectively for fast detection of cracks in concrete structures. Specifically, it mainly includes the following aspects:

1. The optimization algorithm can further optimize the crack mask to obtain the complete crack mask coordinates.
2. The principle of choosing cracks can effectively find crack images from a large number of background images.

3. The contour extraction algorithm can extract the mask coordinations to complete marking of cracks automatically.
4. The application of edge detection can find crack contours accurately on black lines, simulating cracks that are drawn on white paper.
5. After JSON files made from black lines on white paper are given as inputs to the Mask RCNN model for training, the first-order weight file can be used effectively to filter and detect cracks in white rendered concrete, and the optimized mask coordinates are converted into new label files further.
6. The crack mask coordinates are transferred to the concrete crack-free background images to generate new JSON files, which can further train the model.
7. The model obtained by automatic method for annotation of concrete crack images has good recognition speed and accuracy in building crack identification, and it only needs 0.02s to detect the crack position; In the comprehensive evaluation index of crack detection accuracy, black lines on white paper reaches 95.2%, black cracks on white rendered concrete reaches 83.3%, and the concrete cracks reaches 79.2%.

The research described in this paper only selected the most common cracks as the research object and studied the image processing methods of cracks. However, in practical engineering, various diseases such as cracks, leaks, and exposed reinforcement are often encountered. In complex practical hydraulic engineering, further research is needed.

Acknowledgments

The research described in this paper was financially supported by the National Key Research and Development Program of China (No. 2017YFC0405005).

The research described in this paper was financially supported by the Science and Technology Plan Project of Water Resources Department of Zhejiang Province (No. RB2035).

References

1. Zhang L, Yang F, Zhang D, et al. Road crack detection using deep convolutional neural network[C]// IEEE International Conference on Image Processing. IEEE, 2016. DOI:10.1109/ICIP.2016.7533052.
2. Zeiler M D, Fergus R. Visualizing and Understanding Convolutional Networks[J]. 2014. DOI:10.1007/978-3-319-10590-1_53.
3. Yokoyama S, Matsumoto T. Development of an Automatic Detector of Cracks in Concrete Using Machine Learning[J]. Procedia Engineering, 2017, 171(Complete):1250-1255. DOI:10.1016/j.proeng.2017.01.418.
4. Xu K, Ba J, Kiros R, et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention[J]. Computer Science, 2015: 2048-2057. DOI: 10.48550/arXiv.1502.03044.
5. Maier-Hein L, Ross T, Grhl J, et al. Crowd-Algorithm Collaboration for Large-Scale Endoscopic Image Annotation with Confidence[C]// International Conference on Medical Image

- Computing and Computer-Assisted Intervention. Springer, Cham, 2016. DOI:10.1007/978-3-319-46723-8_71.
6. Zhou Z, Sodha V, Siddiquee M M R, et al. Models Genesis: Generic Autodidactic Models for 3D Medical Image Analysis[J]. 2019. DOI:10.1007/978-3-030-32251-9_42
 7. Jiang L, Li C, Wang S, et al. Deep feature weighting for naive Bayes and its application to text classification[J]. Engineering Applications of Artificial Intelligence, 2016, 52(Jun.):26-39. DOI:10.1016/j.engappai.2016.02.002.
 8. Liu K, Guo Y, Wang S, et al. Semi-supervised Learning Based on Improved Co-training by Committee[C]// International Conference on Intelligent Science and Big Data Engineering. Springer International Publishing, 2015. DOI:10.1007/978-3-319-23862-3_41.
 9. Yuan S, Ying W, Dazhe Z. Computer-Aided Lung Nodule Recognition by SVM Classifier Based on Combination of Random Undersampling and SMOTE[J]. Computational and Mathematical Methods in Medicine, 2015, (2015-4-6), 2015, 2015: 368674. DOI: 10.1155/2015/368674.
 10. Kallenberg M, Petersen K, Nielsen M, et al. Unsupervised Deep Learning Applied to Breast Density Segmentation and Mammographic Risk Scoring[J]. IEEE Transactions on Medical Imaging, 2016, 35(5):1322-1331. DOI:10.1109/TMI.2016.2532122.
 11. Shen W, Zhou M, Yang F, et al. Multi-crop Convolutional Neural Networks for lung nodule malignancy suspiciousness classification[J]. Pattern Recognition, 2017, 61(61):663-673. DOI:10.1016/j.patcog.2016.05.029.
 12. Wataru, Kitagawa, Atsushi, et al. Objective Function Optimization for Electrical Machine by using Multi-Objective Genetic Programming and Display Method of Its Results[J]. IEEE Transactions on Electronics, Information and Systems, 2019, 139(7):796-801. DOI:10.1541/ieejieiss.139.796.
 13. Xu Y, Jia Z, Ai Y, et al. Deep convolutional activation features for large scale Brain Tumor histopathology image classification and segmentation[C]// 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015. DOI:10.1109/ICASSP.2015.7178109.
 14. Bahrololoum A, Nezamabadi-Pour H. A Multi-Expert based Framework for Automatic Image Annotation[J]. Pattern Recognition, 2017, 61:169-184. DOI:10.1016/j.patcog.2016.07.034.
 15. Xu Y Y. Multiple-instance learning based decision neural networks for image retrieval and classification[M]. Elsevier Science Publishers B. V. 2016. DOI:10.1016/j.neucom.2015.07.024.
 16. Kong T, Yao A, Chen Y, et al. HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016. DOI:10.1109/CVPR.2016.98.
 17. He Y, Wang J, Kang C, et al. Large Scale Image Annotation via Deep Representation Learning and Tag Embedding Learning[C]// Acm on International Conference on Multimedia Retrieval. ACM, 2015. DOI:10.1145/2671188.2749330.
 18. Ronggui, Wang, Kai, et al. A novel method for image classification based on bag of visual words[J]. Journal of Visual Communication & Image Representation, 2016. DOI:10.1016/j.jvcir.2016.05.022.
 19. Gu J, Wang G, Cai J, et al. An Empirical Study of Language CNN for Image Captioning[C]// 2017 IEEE International Conference on Computer Vision (ICCV). IEEE, 2017. DOI:10.1109/ICCV.2017.138.
 20. Rivera, José, Herrera G, Chacón, Mario, et al. Improved Progressive Polynomial Algorithm for Self-Adjustment and Optimal Response in Intelligent Sensors[J]. Sensors, 2008, 8(11):1395-1401. DOI:10.3390/s8117410.

21. Joshi D, Singh T, Sharma G. Automatic surface crack detection using Segmentation-based deep-learning approach [J]. *Engineering Fracture Mechanics*, 2022. DOI: 10.1016/j.engfrac-mech.2022.108467.
22. Wu Y, Li S, Zhang J, et al. Dual attention transformer network for pixel-level concrete crack segmentation considering camera placement[J]. *Automation in Construction*, 2024, 157. DOI: 10.1016/j.autcon.2023.105166.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as 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.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

