Surveyr Part 2: Cracks!!!–Unsupervised.

The Kazakh Team (Part 1) , I. Starikova [1], M. Zyskin†

[1] *Slovak Academy of Sciences*

(*Communicated to* MIIR *on 3 September 2025*)

Study Group: 187th European Study Group with Industry, 14th-18th July 2025, University of Bristol, UK
Industrial Partner: Surveyr
Presenter: Amy Stone.
Team Members: The Kazakh Team -Part 1; Irina Starikova, Slovak Academy of Sciences; Maxim Zyskin, consultant.
Industrial Sector: Computing/Robotics, Data Analysis.

Key Words: Image analysis, machine learning, unsupervised learning, edge detection, Frangi filter, crack detection. Classification, segmentation, feature detection.

Summary

This is part 2 of the report, focusing primarily on image analysis and (unsupervised) machine learning approach to crack detection

Contents

† Corresponding Author, Part 2: maximzyskin@gmail.com

2 Crack detection via unsupervised image analysis

Part 1 of the report is focusing on supervised learning methods, requiring collection and labeling of training sets for training image detection neural network, subsequently used for Yolo-like segmentation search, scoring bounding box jointly with the detection (thus enabling to recognize learned objects in a larger image full with competing details). Some public neural networks detecting cracks are already trained and made available, for example on the Roboflow platform. While Yolo method is published as a research paper, and also can be these days called from Python as an open source library (we have done so as part of the project) — and can be coupled in the same Python code via API with a public trained detection model deployed somewhere. e.g. on Roboflow (we have tried this, though our Part 2 team have not pursue it to the end), creating a possible route to solve the Surveyr problem. Further details and discussion of this line of approach have been explored and presented in the spectacular Part 1 of the report by the Kazakh team.

However , this is really not our point here. In this part 2 of the report, we would like to explore primarily image analysis and (unsupervised) learning methods, in as much as they apply to crack detection, and in as much as they seem promising for developing image analysis and machine learning methods.

2.1 Stand-alone cracks

'Old school' methods of edge detection involve only simple operations on pixilated image, essentially based on discrete version of a derivative, or convolution with a Gaussian kernel and differentiating the result, and then studying regions where the norm of the gradient is large , as well as looking at second derivatives, such as sign of Laplacian, to distinguish narrow black feature from narrow white feature; eigenvalues of the Hessian, to verify that that has one large eigenvalue and one small eigenvalue, the latter corresponding to a direction where gray level does not change much and orthogonal to direction where it changes a lot.

Such edge detection methods would be very effective if a well-isolated crack is seen, for example:

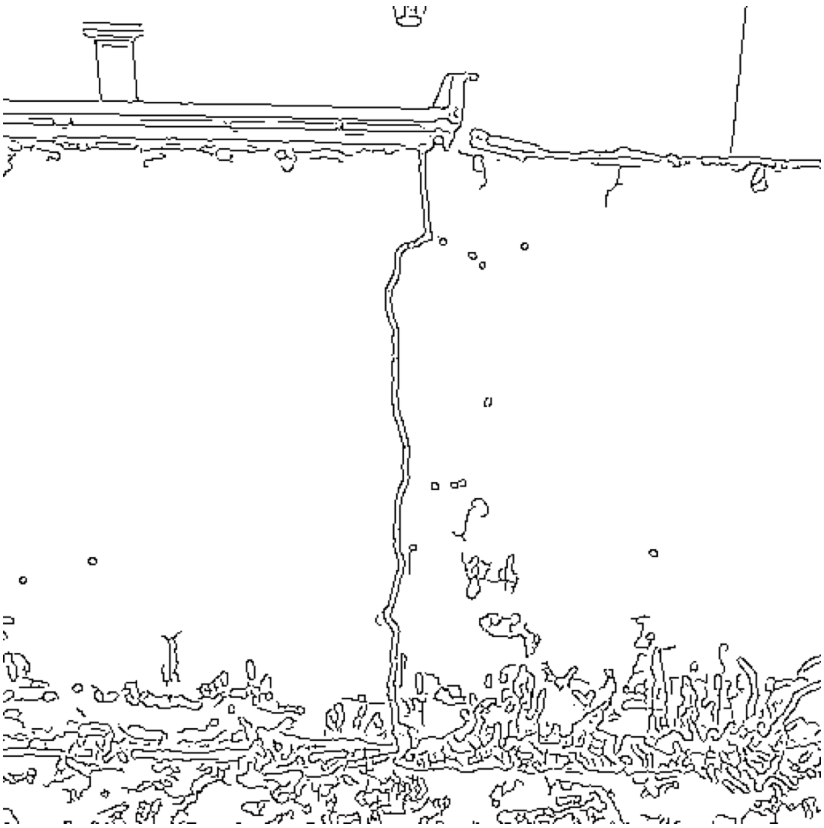

Figure 1. Edge detection applied to gray scale image. Vertical long crack is clearly seen. Building features (top), grass and wall edge (bottom) are also seen

Well-isolated cracks are also picked up by some public trained and deployed models, see for example Fig. 2.

Another public model, Fig. 3, with name ending in ../crack-detection-ypnwo/v3, was also successful, however we have not saved, and could not now find, the link to it now:

The other few models we have also tried, some of them with larger training set, have failed to detect crack in this image. We observed that although image preparation like edge detection helps a lot for a human to see cracks, this does not necessarily extends to models pre-trained on full untreated images. We expect though that training on prepared images could be more successful, as in a way there is less to learn.

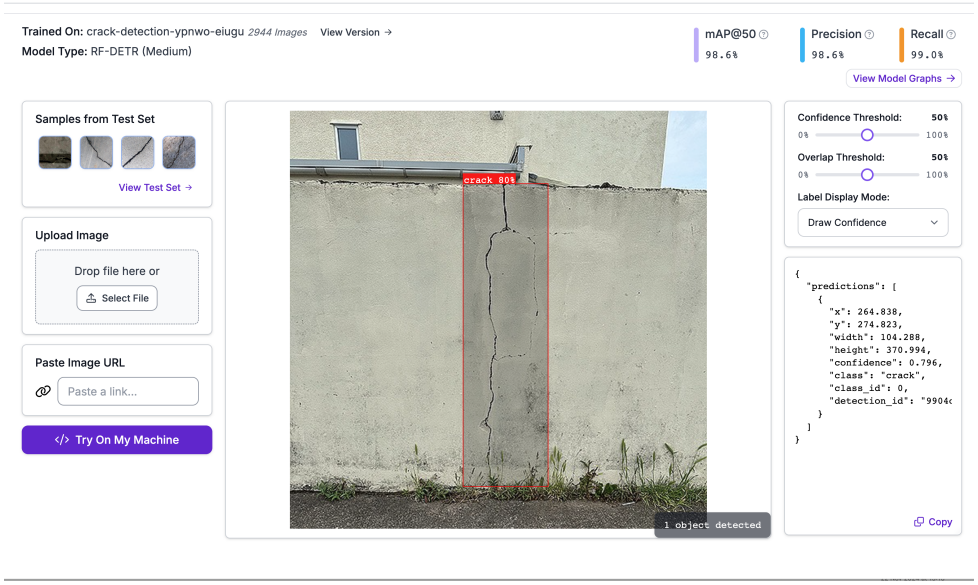

Figure 2. Public trained model deployed on Roboflow finds and identifies the feature as crack, correctly. The model link: https://universe.roboflow.com/roads-fgydm/crack-detection-ypnwo-eiugu/model/1, checked on Sept 1, 2025.

Another public model with name ending in ../crack-detection-ypnwo/v3, was also successful, Fig. 3 (when tested on Aug. 29, 2025), however we have not saved, and could not find the full link to it, at this time.

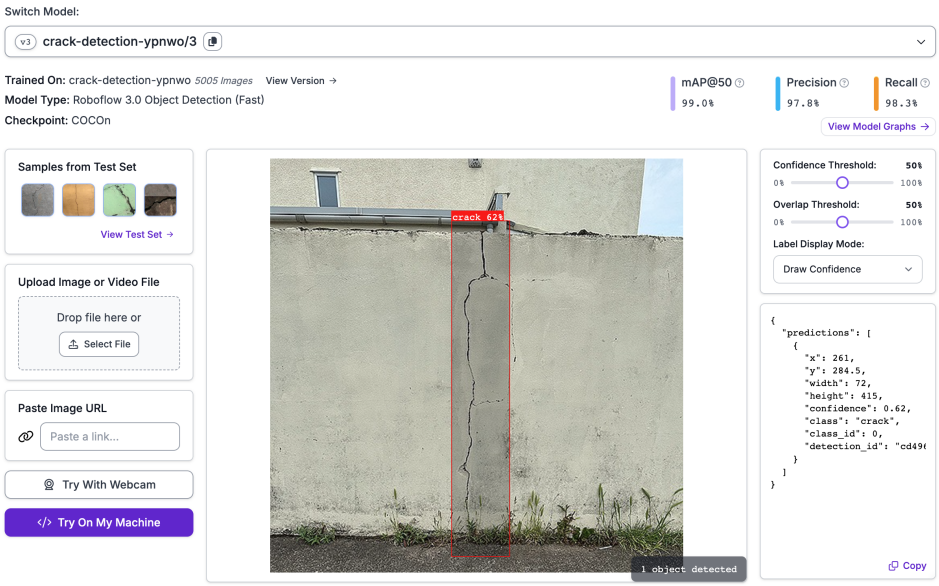

Figure 3

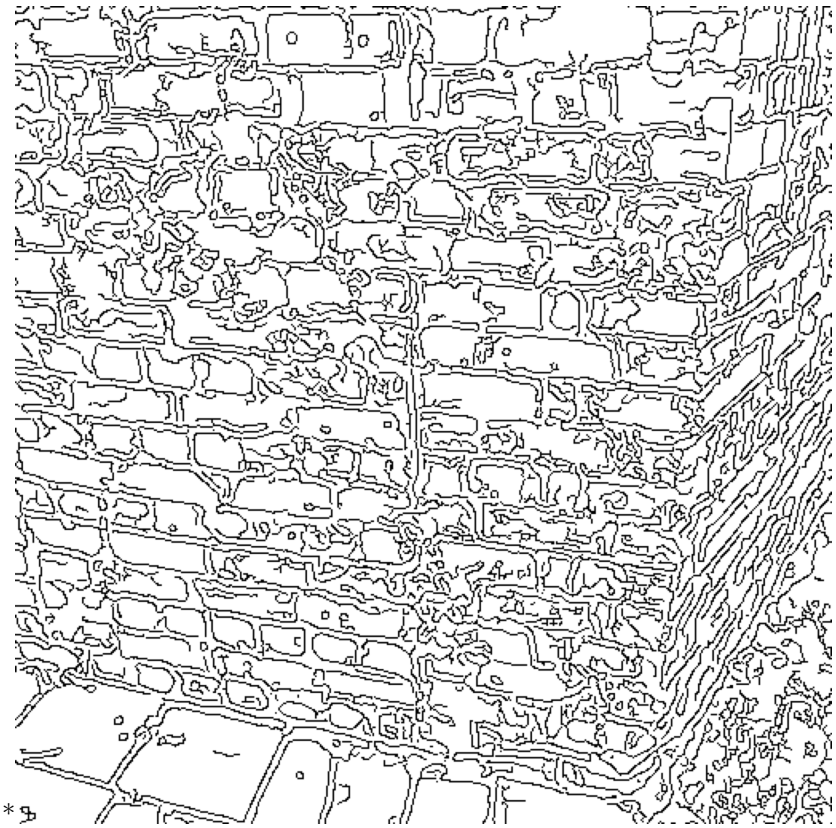

Figure 4. Edge detection on a brick wall seems a bit loaded...Can you see the crack?

2.2 Bricks !!!

2.3 Pre-processing

Cracks through mortar in brick walls provides more challenge. Edge detection will pick up not just the crack, but also bricks/mortar boundaries, as well as other sharp features, and may look as so: We will outline below an unsupervised detection procedure, which would not require extensive collection and labeling of training sets (or alternatively can assist in supervised learning, by simplifying and reducing amount of non-essential features in the imagery. Also, such reduced imagery can be generated synthetically by a computer, with actual raw images not necessarily required. Our approach consists of 2 main steps, broadly described as (i) a filtered edge detection (ii) curves fitting and post-selection.

We have examined several filtered detection approaches. 1. Converted to grayscale image convolution with a Gaussian kernel, averaging over an area on a selected scale. With a good scale choice, this softens/washes out mortar edges, relative to cracks, Fig. 5, right. 2. Follow-up adjusting of the image gray scale to its full 0 to 1 range is suppressing lighter brick and mortar region relative to the crack, Fig. 5, (a). Adjusting contrast also have a similar effect. 3. Further, it proved very effective to suppress lighter gray areas altogether, as the crack is the darkest, as it is not reflecting visual light back. Keeping only

dark areas (we kept values below 0.05), followed by attempted removal of small features (we have not observed a significant effect) is shown in Fig. 6, left. Frangi vesselness filter, based on eigenvalues of the matrix Hessian H, and developed for blood vessel detection, is applied to the result of the previous step.

For a 2D image, the Hessian matrix H at point (x, y) is:

$$H = \begin{bmatrix} \frac{\partial^2 I}{\partial x^2} & \frac{\partial^2 I}{\partial x \partial y} \\ \frac{\partial^2 I}{\partial x \partial y} & \frac{\partial^2 I}{\partial y^2} \end{bmatrix} \tag{2.1}$$

Let λ_1 and λ_2 be the eigenvalues of H with $|\lambda_1| \leq |\lambda_2|$. The "blobness" parameter is defined as $R_B = \frac{|\lambda_1|}{|\lambda_2|}$, and the "structureness" parameter as $S = \sqrt{\lambda_1^2 + \lambda_2^2}$. The filtered result, in the version designed to spot dark cracks (low grayness score) on light background (higher grayness score), is then given by

$$V(s) = \begin{cases} 0 & \text{if } \lambda_2 < 0 \\ \exp\left(-\frac{R_B^2}{2\beta^2}\right)\left[1 - \exp\left(-\frac{S^2}{2c^2}\right)\right] & \text{otherwise} \end{cases} \tag{2.2}$$

In a multiscale implementation, a maximum over scales may be also taken,

$$V_{\text{MS}} = \max_{s_{\min} \leq s \leq s_{\max}} V(s). \tag{2.3}$$

In Fig. 6 (b) we show the result of applying Frangi filter to the raw image converted to gray scale (and slightly smoothened with a Gaussian 3 pixels-wide kernel). In Fig. 6 (c) and (d) we show results of edge detection and Frangi filter respectively post-applied to Fig. 6 (c) which keeps only darkest pixels. We observe that after any of the 4 preprocessed methods in Fig. 6, location of the crack can be seen clearly with a human eye, to a various degree of visual contrast.

Figure 5. Left, convolution with a 2 pixels wide Gaussian kernel is washing out mortar edges. Right, follow-up gray scale adjustment to the full 0 to 1 range.

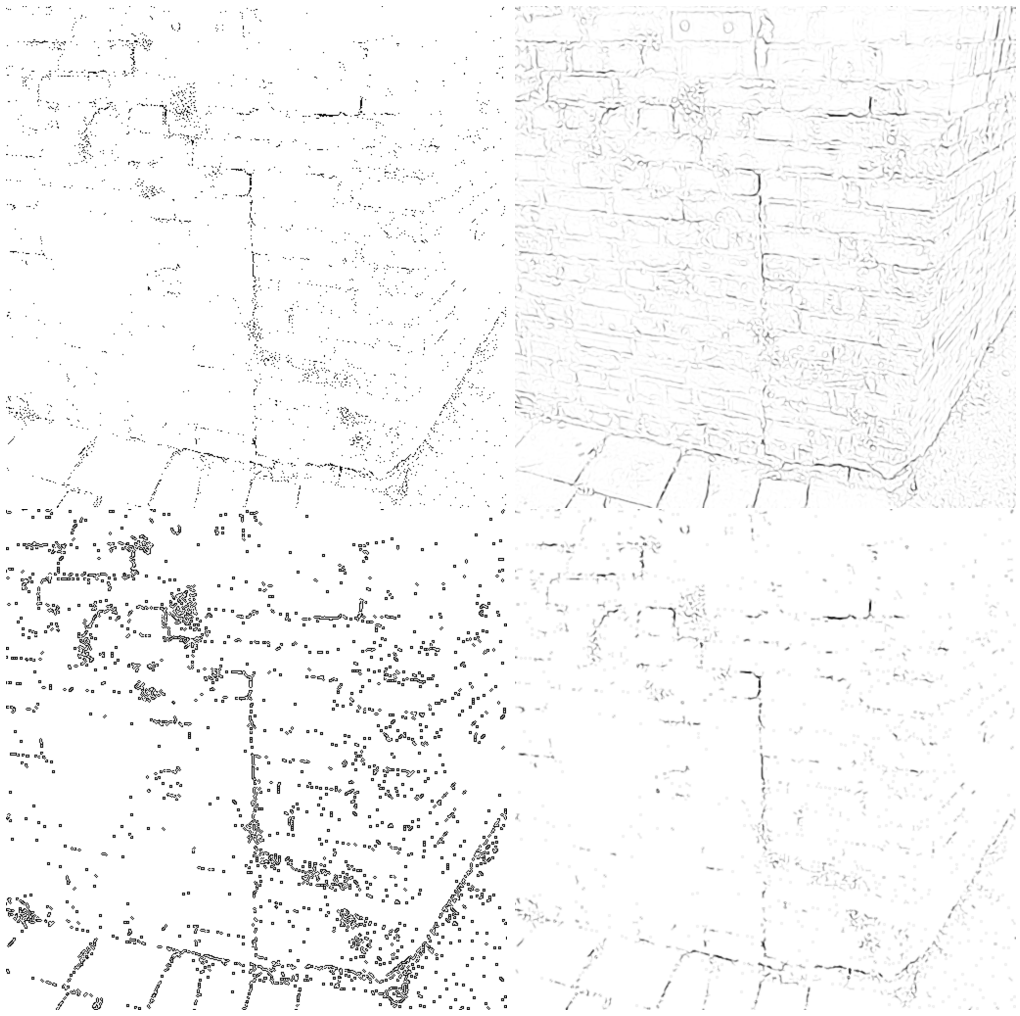

Figure 6. (a) Top left, keeping only dark, gray score ≤ 0.05 regions, followed by small features removal. (b) Top right, Frangi filter with $\beta = 0.5$, $c = 0.1$ applied to the raw image converted to gray and smoothed by 3 pixel Gaussian filter (c) Bottom left, edge detection applied to (a); (d) Bottom right, Frangi filter applied to (a)

2.4 Contours detection

We would like now for the computer to highlight a guess on crack location, by surrounding it by a contour, say. We found that in the preprocessed image the crack pixels are not connected enough for this to work well, so we first dilated dark regions, transforming Fig. 7, left to a thickened version, right. Dilation is done as the Minkowski sum of the negated image with a disk of 4 pixels radius (see the Appendix).

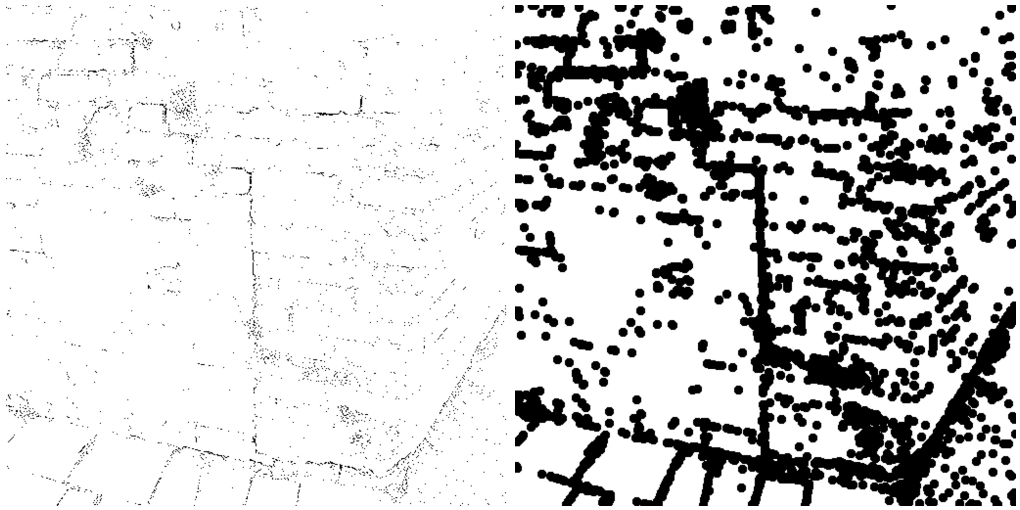

Figure 7.

This is followed by the resulting image binarized into zero and one (black and white, if not like this already, and a contour tracing algorithm is applied to trace the boundary of those by closed polygonal curves (which may involve joining segments, or cutting to avoid self-intersections) Selection is then made among such contours, which may be based on arc length, or largest distance among pixels enclosed. Top contours found by Mathematica, ordered by arc length, are shown in Fig. 8, left. Top contours found using open source vision library and Python are shown in Fig. 8, right.

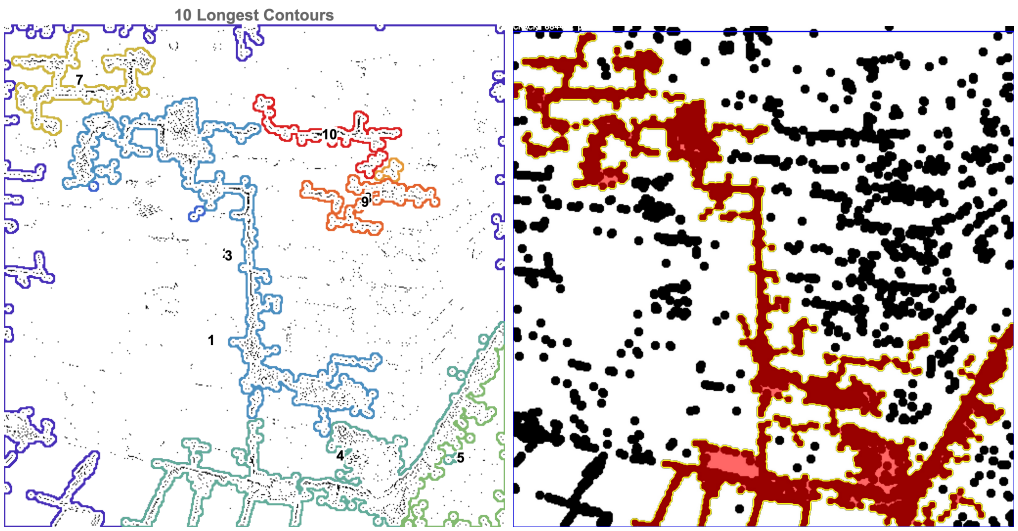

Figure 8. Left: top-length contours fitted on thickened image ;right: final result of an open source Python code contour search; see the Appendix for details.

Further development of the contour tracing method can distinguish algorithmically between contours bounding cracks through bricks mortar (staircase-like) from contours bounding bricks (quadrilateral) , or mortar still present despite pre- processing (lattice-like). We have not tried to do so, in this project of a short duration and no budget, applying already available contour tracing methods instead.

3 Appendix

3.1 Open Source Vision Library

A version of the Python code searching for highest-scoring contours resulted in the finding shown in Fig. 8. Only free tools have been used in the code (Open Source Vision library cv2 and the standard Python libraries).

Figures below are the intermediate steps produced by the code, and the final result.

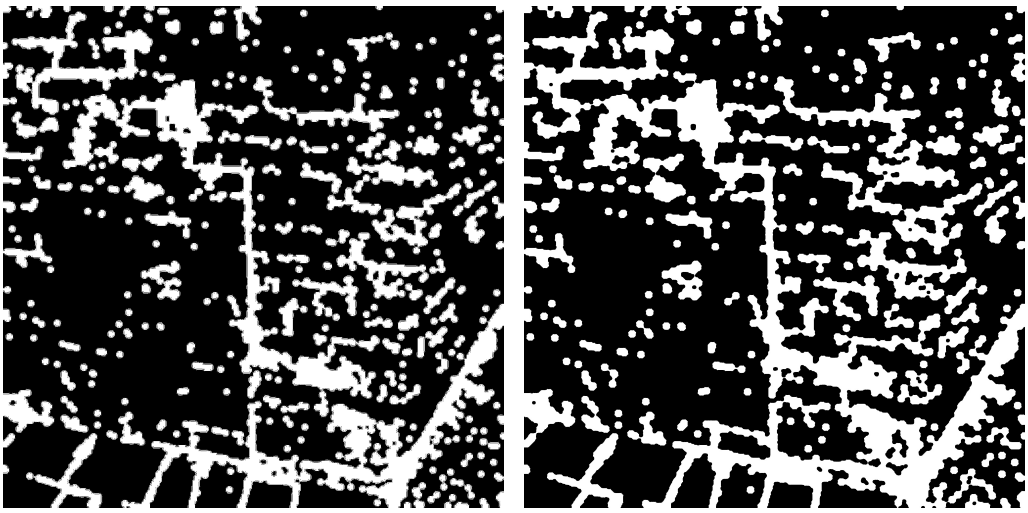

Figure 9. 1. Darkest Regions-left; 2. Connected Regions -right

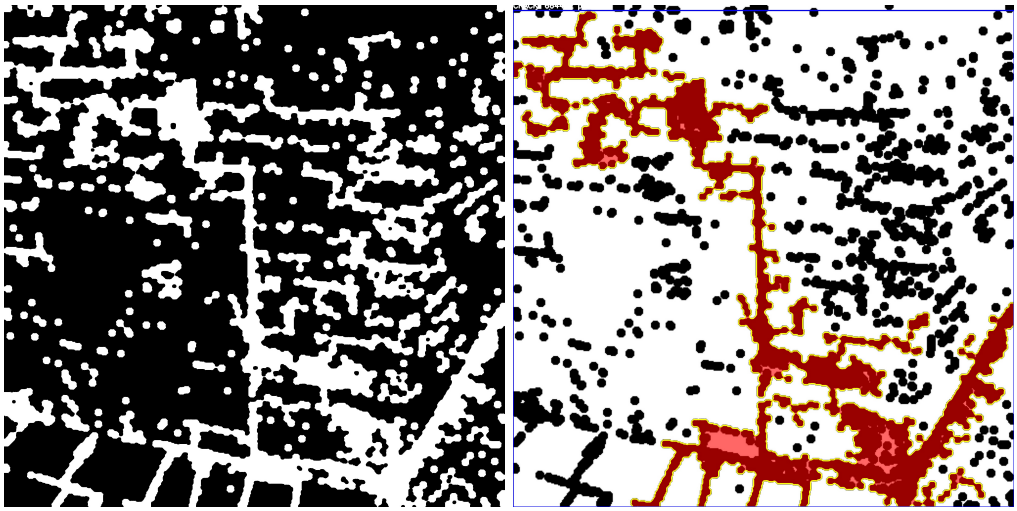

Figure 10. 3. Largest Components-left; final result-right.

3.2 Dilation, by a Minskowski sum with a disk

Let the binary image be represented as a set of foreground pixel coordinates:

$$I = \{\mathbf{p} \in \mathbb{Z}^2 | \text{Pixel at } \mathbf{p} \text{ is foreground}\}$$

The structuring element `DiskMatrix[4]` is a discrete disk of radius 4, defined as the set:

$$B = \{\mathbf{x} \in \mathbb{Z}^2 : \|\mathbf{x}\| \leq 4\}$$

This creates a 9×9 matrix (diameter $= 2 \times 4 + 1 = 9$), where the origin is at the center.

Dilation Operation

The dilation of image I by structuring element B is given by the Minkowski sum:

$$I \oplus B = \{\mathbf{p} + \mathbf{b} | \mathbf{p} \in I, \mathbf{b} \in B\}$$

Equivalently, using the set-theoretic definition for a binary image:

$$(I \oplus B)(\mathbf{s}) = \bigvee_{\mathbf{b} \in B} I(\mathbf{s} - \mathbf{b})$$

where $\bigvee$ denotes the logical supremum (OR) operation.

The procedure was applied to the negated image, negated back after the procedure, so that it was the dark areas which was dilated.

References

[1] M Zyskin. Cracks project code, https://github.com/zyskin/Cracks, 2025.