Decomposing complicated cracks using clustering

Tran Hiep Dinh[*1], **Cong Hieu Le**[1], **Ngoc Khang Nguyen**[1], and **Nguyen Linh Trung**[2]

[1]Faculty of Engineering Mechanics and Automation, VNU University of Engineering and Technology
[2]Advanced Institute of Engineering and Technology, VNU University of Engineering and Technology

Abstract

Simulated cracks have been shown to have a high correlation with the real-world ones and application potentials in vision-based crack detection. However, complicated-structure cracks, such as alligator crackling, are difficult to model. To tackle this challenge, we propose to decompose complicated cracks into simple segments by determining intersections using clustering algorithms. Experimental results have verified the effectiveness of our approach, where the fitting errors *RMSE* and *MAE* between the simulated and real-world cracks can be reduced by half.

[*]Corresponding author. Email: tranhiep.dinh@vnu.edu.vn

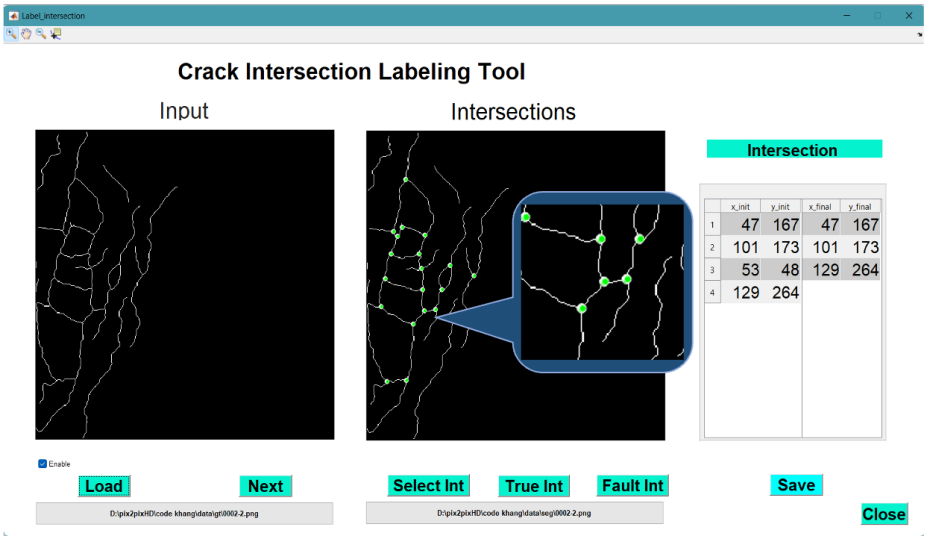

Figure 1: Intersection ground-truth labeling with GUI.

1 Introduction

In this paper, we investigate the employment feasibility of clustering algorithms in decomposing complicated cracks into single segments. The reasons for the selection of clustering algorithms for crack detection are i) the The contributions of our paper are two-fold: (i) a Graphic User Interface (GUI) to label intersections in complicated structure cracks, and (ii) an effective approach for crack structure decomposition using clustering algoirthms.

The paper is structured as follows: The participating clustering algorithms and the testing dataset are introduced in Section 2. Qualitative and quantitative results of the participating algorithms in terms of the intersection detection and correlation are reported in Section 3.

2 Methodology

2.1 Dataset

Here, we use 86 crack images containing complicated structures from the CRKWH100 (15 images), CrackLS315 (32 images), and CrackTree200 (39 images) datasets [7] to form a new dataset for evaluation of participating algorithms. The crack intersections are manually labeled using our developed GUI to speed up the process. The labeling with GUI is demonstrated in Fig. 1. Here, tentative intersections are first calculated with the Crossing number algorithm and demonstrated as red pixels at the concerning point. The region containing the point under investigation is zoomed in for a manual check. The confirmed intersection is then represented as a red cross with a black pixel in the centre for the last time review. The corresponding coordinates are finally extracted for later evaluation of participating algorithms. In total, 982 intersection points are labeled as ground-truth from 86 images.

2.2 Clustering algorithms

The analyzed algorithms include the following, the brief description and parameter number of which are reported in Table 1:

- Density-based spatial clustering of applications with noise (DBSCAN) [2]

Table 1: Brief description and corresponding parameter number of participating algorithms

Algorithm	Brief description	Parameter number
DBSCAN	Clusters are formed from core points if the number of the surrounding neighbors is greater than a pre-defined threshold.	1
DP	Cluster centers are assigned to data points with high local density values and appropriate distances to each other.	2
Denmune	The cluster assignment is based on a voting system framework to identify dense regions, where data points are classified as strong, weak, and noise.	1
HIAC	The high similarity between neighbors is leveraged with added gravitation via a weight probability distribution curve.	1
FINCH	Clusters are first estimated then recursively chain-linked using the first neighbor of each data point	0
DP-Dsets-DBSCAN	Initial clusters are formed based on their dominant density, then expanded using DP to deal with real-world data which can be approximated by Gaussian distribution.	0

- Clustering by fast search and find of Density Peak (DP) [5]
- DENsity peak-based clustering using MUtual NEarest neighbors (DenMune) [1]
- An optimization method that Highly Improves the Accuracy of Clustering algorithms (HIAC) [4]
- FIrst Nearest Neighbor Clustering Hierarchy (FINCH) [6]
- DP-DSets-DBSCAN [3]

Given a binary input image, let $\mathcal{P} = \{p_1, p_2, ..., p_n\}$ contain all white pixels representing crack. K disjoint subsets of $\mathcal{P}$ are then formed using participated clustering algorithms on the elements of $\mathcal{P}$, denoted as $C_1, C_2, ..., C_K$. The centroid of each cluster is calculated as

$$c_K = \frac{1}{N_K} \sum_{n \in C_K} p_n, \tag{1}$$

where N_K is the element number in C_K. To deal with cases where the cluster centroid is not the actual data point, the closest element of C_K to the cluster centroid c_K is employed to evaluate the effectiveness of clustering algorithms in crack intersection detection.

Fig. 2 demonstrates clustering results of participated algorithms, the corresponding centroid, the closest data point, and the ground truth of a sample input.

2.3 Clustering-based Intersection Detection (CID)

The flowchart, pseudo-code, and demonstration of the proposed algorithm are shown respectively in Fig. 3, Algorithm 1, and Fig. 4.

2.4 Evaluation Metrics

2.4.1 Intersection Detection Accuracy

Here, we use the F_1 to evaluate the detection accuracy, the formula of which is given as

$$F_1 = 2 \times \frac{Pr \times Rc}{Pr + Rc}, \tag{2}$$

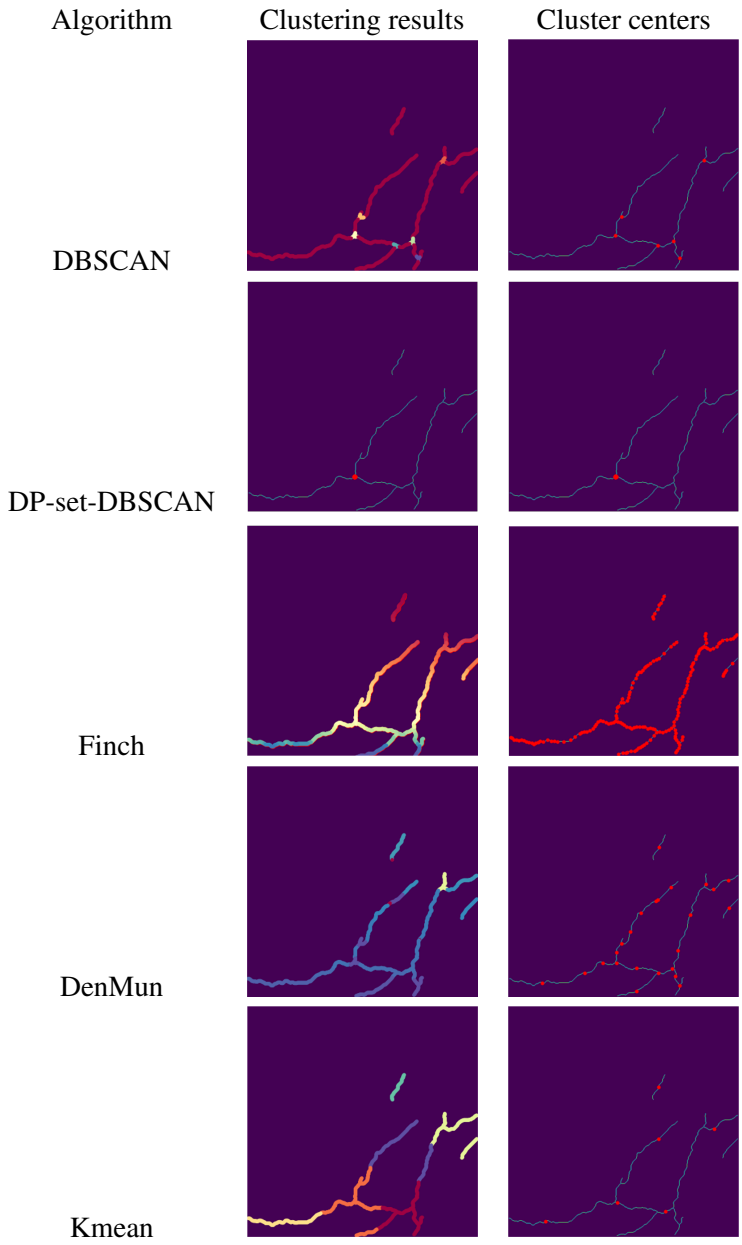

Figure 2: Clustering results and corresponding centers of participating algorithms.

where Pr and Rc are respectively the precision and recall of the detection.

2.4.2 *Correlation Between Simulated and Real-world Cracks*

Here, we use the Root Mean Squared Error (*RMSE*), the Mean Absolute Error (*MAE*), and the Coefficient of Determination R^2. Let x_{02}, $\hat{x}_{02}$ respectively be the actual and estimated crack tip coordinate, N be the number of tips in each crack, the aforementioned metrics can be calculated as

$$RMSE = \sqrt{\frac{1}{N}\sum_{s=1}^{N}(x_{02\,s} - \hat{x}_{02})^2}, \tag{3}$$

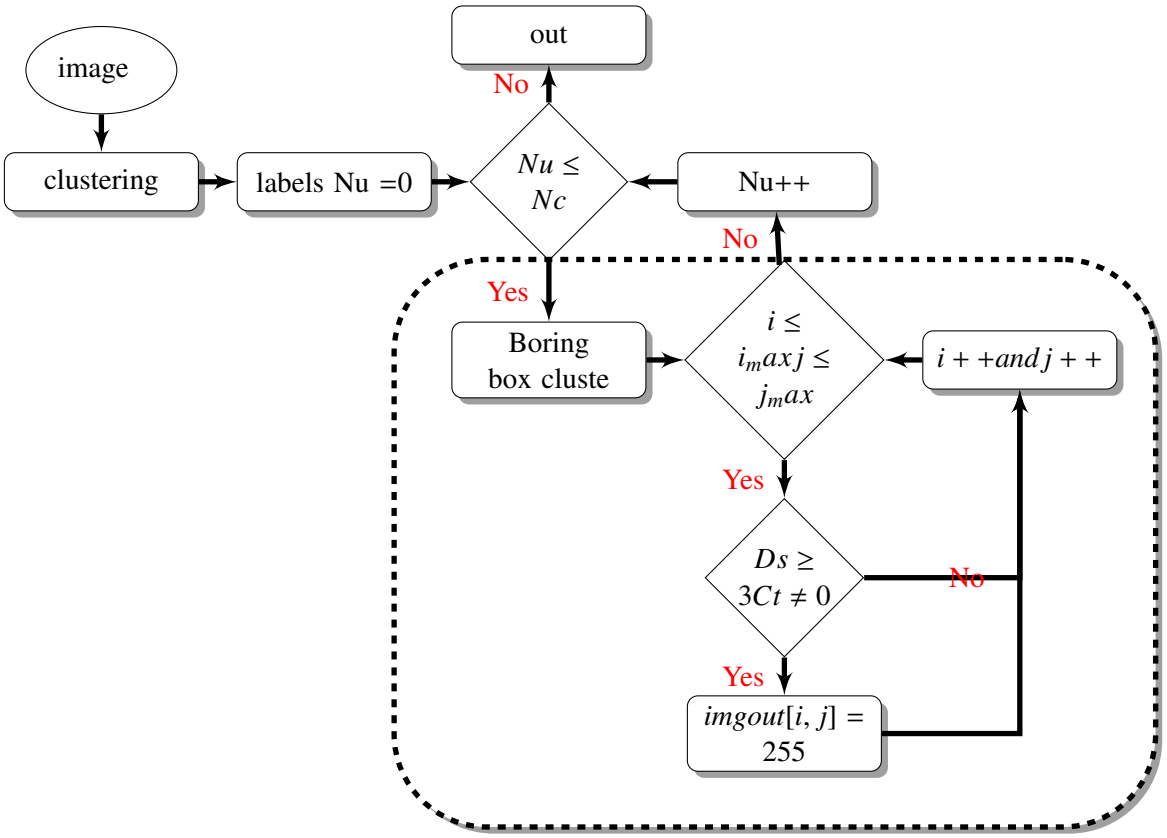

Figure 2: Pipeline of the ranking-based intersection detection

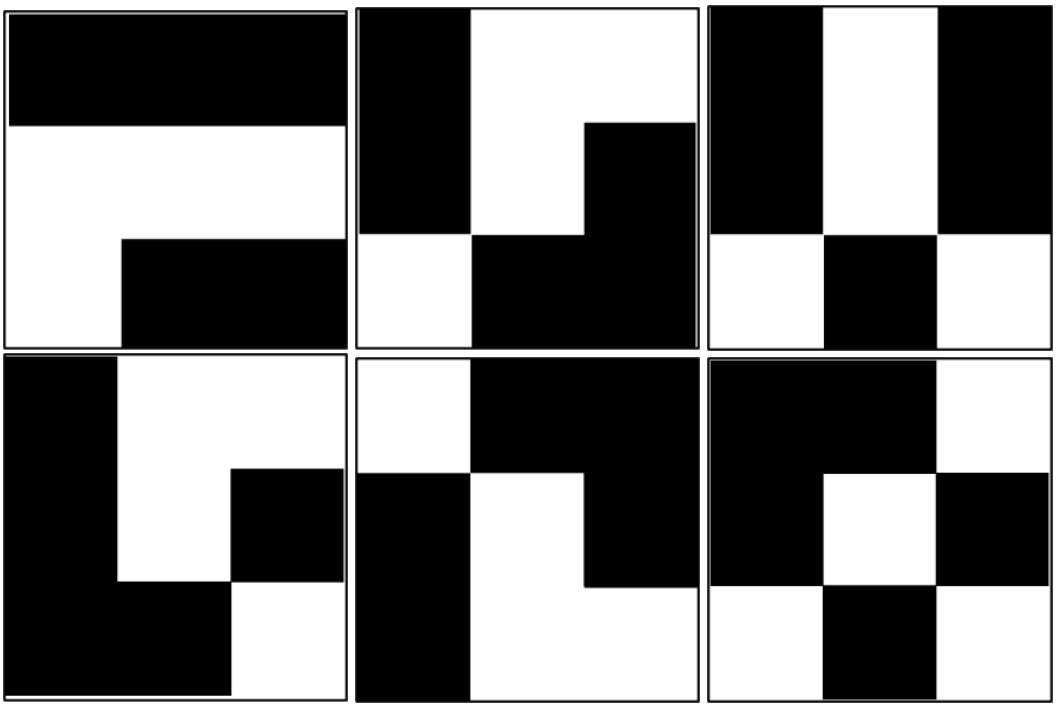

Figure 3: Demonstration of the proposed intersection detection algorithm.

Algorithm 1 Ranking-based clustering for crack intersection detection

matrix_in ← indices of elements in *image* where *image* > 0
labels ← clustering algorithm applied to *matrix_in*
matrix_out ← create zeros matrix like *matrix_in*
window_size ← 3
window_area ← *window_size* × *window_size*
window_center ← ⌊*window_size* ÷ 2⌋
for $label_i$ in unique values of *labels* **do**
    matrix_label ← elements in *matrix_in*1 where *labels* = $label_i$
    matrix_in ← create zeros matrix with size (max(matrix_label[:, 0]) + 1, max(matrix_label[:, 1]) + 1)
    set elements in *matrix_in* at indices (matrix_label[:, 0], matrix_label[:, 1]) to 1
    for *i* in range (min(matrix_label[:, 0]), max(matrix_label[:, 0])) **do**
        for *j* in range (min(matrix_label[:, 1]), max(matrix_label[:, 1]) **do**
            window ← submatrix of *matrix_in* with size 3 × 3 centered at (*i*, *j*)
            density ← sum of elements in *window*
            if *density* > 3 and *matrix_in*[*i*, *j*] = 1 **then**
                set element in *matrix_out* at index (*i*, *j*) to 255
            end if
        end for
    end for
end for

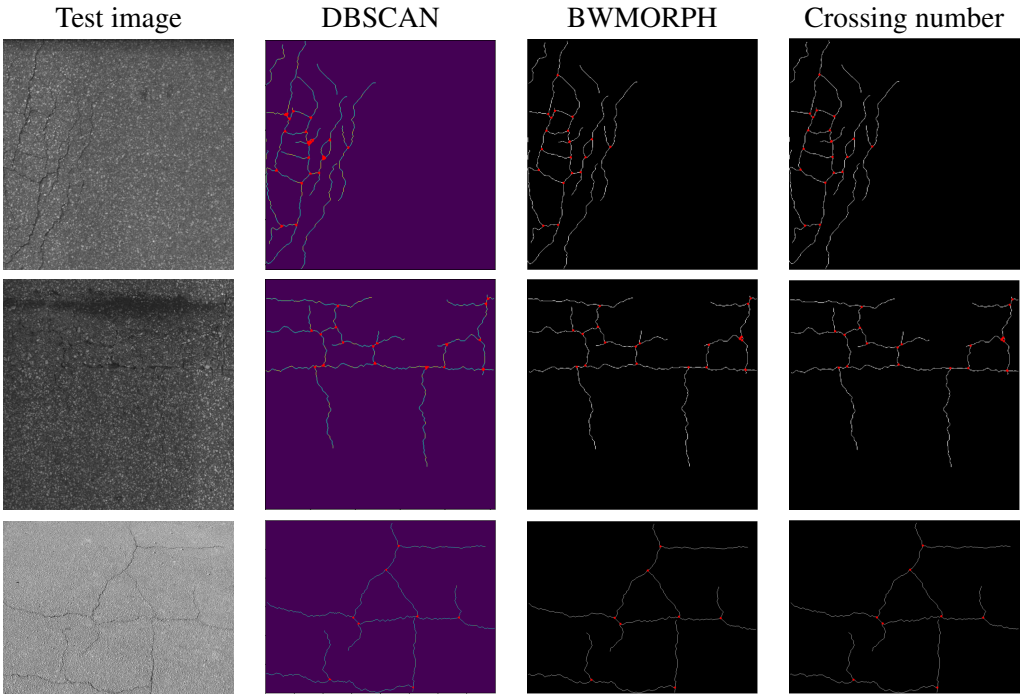

Figure 5: Qualitative results of participating clustering algorithms.

$$MAE = \frac{1}{N}\sum_{s=1}^{N}|x_{02\,s} - \hat{x_{02}}|, \tag{4}$$

$$R^2 = 1 - \frac{\sum_{s=1}^{N}(x_{02\,s} - \hat{x}_{02})^2}{\sum(x_{02\,s} - \overline{x}_{02})^2}. \tag{5}$$

Table 2: Intersection detection accuracy of participating algorithms

	with CID			without CID		
Algorithm	F1	Rc	Pr	F1	Rc	Pr
DenMune	91.68%	94.26%	89.33%	57.98%	54.14%	62.61%
FINCH	87.74%	94.03%	82.72%	54.63%	50.64%	59.80%
TW-FINCH	79.08%	91.25%	70.28%	53.56%	50.44%	57.31%
K-means	91.67%	94.27%	89.30%	50.04%	50.07%	50.02%
DP-Dsets-DBSCAN	75.02%	98.68%	60.57%	50,34%	51,27%	52,37%
DBSCAN	91.67%	94.28%	89.29%	76.64%	75.85%	77.54%
BWMORPH	N/A	N/A	N/A	99,35%	98,79%	99,95%
CN	N/A	N/A	N/A	99,50%	99,14%	99,89%

Table 3: Correlation comparison between before- and after decomposition

Algorithm	*RMSE*	*MAE*	R^2	*MS*
Before decomposition	0.26812	0.21012	-0.18240	1.65380
With BWMORPH	0.01425	0.01132	0.74957	0.27398
With Crossing number	0.01425	0.01132	0.74275	0.27783
With DBSCAN	0.01412	0.01105	0.74121	0.28288
With DP-Dsets-DBSCAN	0.01718	0.01366	0.58994	0.45923
With CFSFD	0.06986	0.05547	0.37140	0.78136
With DenMune	0.01672	0.01327	0.59833	0.44863
With FINCH	0.02200	0.01785	0.54979	0.49697
With TW-FINCH	0.04312	0.03348	0.44074	0.61336
With k-means	0.01658	0.01327	0.60326	0.44444

We also introduce a fitting score, *MS*, calculated as the sum of the above evaluation metrics, normalized in the same [0, 1] range, i.e.

$$|MS| = |RMSE| + |MAE| + |1 - R^2|. \tag{6}$$

3 Results

3.1 Crack intersection detection

Qualitative results of participating clustering algorithms are presented in Fig. 4 where the vulnerability of DBSCAN on intersections that are close to each other is demonstrated (first row). On the other hand, BWMORPH and Crossing number perform equally with the same number of true positives returned on all test images. This observation is also verified in Table 2 showing the out-performance of BWMORPH and Crossing Number against DBSCAN.

3.2 Change in correlation after decomposition

The change in correlation between real-world cracks and the simulated ones is reported in Table 3. Here, the correlation after decomposition is calculated as the average of the correlation of each crack segment and the best-fit model. Notably, the fitting errors *RMSE* and *MAE* are reduced by half while the fitting score R^2 is significantly increased. This has confirmed our hypothesis that decomposing complicated crack structures into single crack segment could overcome the current simulation limitation, where only numerical models of single crack can be obtained.

References

[1] Mohamed Abbas, Adel El-Zoghabi, and Amin Shoukry. Denmune: Density peak based clustering using mutual nearest neighbors. *Pattern Recognition*, 109:107589, 2021.

[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[3] Jian Hou, Huaqiang Yuan, and Marcello Pelillo. Towards parameter-free clustering for real-world data. *Pattern Recognition*, 134:109062, 2023.

[4] Qi Li, Shuliang Wang, Xianjun Zeng, Boxiang Zhao, and Yingxu Dang. How to improve the accuracy of clustering algorithms. *Information Sciences*, 627:52–70, 2023.

[5] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014.

[6] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8934–8943, 2019.

[7] Qin Zou, Zheng Zhang, Qingquan Li, Xianbiao Qi, Qian Wang, and Song Wang. Deepcrack: Learning hierarchical convolutional features for crack detection. *IEEE Transactions on Image Processing*, 28(3):1498–1512, 2018.