A Method for Crack Detection and Quantification in Masonry Using Neural Network-Based Image Analysis

Lorenz Krefft[1], Ludwig Hoegner[2]

[1] Department of Geoinformatics, Hochschule Munchen University of Applied Sciences, Munich, Germany – lorenz.krefft@hm.edu
[2] Department of Geoinformatics, Hochschule Munchen University of Applied Sciences, Munich, Germany – ludwig.hoegner@hm.edu

Keywords: Crack Segmentation, Crack Quantification, Damage analysis, Neural Networks, Dijkstra.

Abstract

This article presents a method for the automated detection and quantification of cracks on masonry surfaces. The core of the approach is a neural network trained for semantic segmentation, which enables the identification of cracks in image data. To facilitate a physically meaningful analysis, the image data is combined with 3D geometric information. A 3D point cloud is projected onto the image plane to establish correspondences between 2D image points and 3D spatial coordinates. These 2D–3D correspondences are utilized to evaluate the detected cracks in a geometrically accurate manner. Based on the segmentation results and the projected 3D data, cracks can be classified within the point cloud and analyzed metrically. The Crack length is determined using a graph-based model, in which the crack structure is represented as a network and the longest continuous crack path is computed using Dijkstra's algorithm. The Crack width is measured in the images based on the segmentation masks and a scaling factor derived from the 2D–3D correspondences. The proposed method enables a precise and automated assessment of crack patterns in masonry structures by leveraging both image and 3D data.

1. Introduction

The early detection and analysis of damage in masonry structures is an important aspect of structural monitoring. Cracks in masonry can indicate structural problems or material aging processes and are a potential risk to structural health. However, traditional inspection methods are time-consuming, costly and highly dependent on the expertise of the inspector (Xu et al., 2019; Flotzinger et al., 2022). For this reason, alternative, automated approaches are being researched.

The use of artificial intelligence methods opens new options for a reliable and scalable crack detection. In the last few years, numerous methods for semantic segmentation have been developed that can be applied to two-dimensional images and 3D point clouds. These methods are becoming more and more important in comparison to traditional methods. By using neural networks, cracks can be detected even in the presence of noise and under different imaging conditions (Yadhunath et al., 2021).

The studies by Yiğit and Uysal (2024) and Azhari et al. (2021) focus on crack detection in 3D point clouds using neural networks. However, the development of such networks requires extensive annotated training data. While numerous publicly available datasets exist for image-based approaches, comparable datasets for point clouds are still limited. Creating custom datasets is labor-intensive, as it demands precise annotations. Furthermore, point cloud-based methods face challenges in reliably detecting fine-structured damage, such as hairline cracks (Azhari et al., 2021).

An alternative approach involves combining the strengths of image processing with 3D data. In Yang et al. (2020) and, Lawin et al. (2017) a method is described in which synthetic views are generated from point clouds. Semantic segmentation of the images is then performed, and the results are applied to the point cloud. With this approach, point clouds can be classified based on images.

In addition to point cloud-based methods, a wide range of image-based approaches for crack detection exists, ranging from classification and object detection (Wang et al., 2024) to semantic segmentation. To enable comparisons of damage across different times, Liu et al. (2022) describes a method in which images are geometrically registered, allowing time-based analysis.

Majidi et al. (2023) presented an alternative approach. In this study, a DeepLabV3+ network was trained for the semantic segmentation of cracks. The network was trained using an image resolution of 128×128 pixels. To successfully detect cracks in high-resolution images, these images are divided into smaller tiles. Within these tiles, crack segmentation is performed, and detected cracks are highlighted in red. The individual tiles are then reassembled into a single image. Multiple images of the crack were taken from different perspectives. Structure from Motion was used to create a 3D model of the scene, in which the cracks are highlighted in red.

To enable a metric description of detected cracks, Jahanshahi and Masri (2012) presents a method that photogrammetrically generates a 3D point cloud and applies classical image processing techniques for segmentation. The resulting masks are validated using a classifier, and the depth information from the 3D reconstruction is used to derive metric measurements of the cracks' lengths and widths.

A combined approach is presented in (Hu et al., 2024), where data from a LiDAR scanner is fused with RGB images. By calibrating the sensors, the 3D information can be projected onto the image plane. Semantic segmentation is then performed on the images using neural networks before the detected structures are mapped back onto the point cloud. This enables pixel-accurate analysis as well as precise spatial localization of cracks within the 3D model.

The work of Mohan and Poobal (2018) provides a comprehensive overview of image-based, automated methods for crack

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W6-2025-113-2025

analysis. According to the authors, there are already existing systems that have a high detection accuracy. However, they point out that these systems reach their limits, particularly with complex crack patterns and low contrast images. Against this background, they advocate more research into multimodal approaches. They also highlight the importance of high-resolution image data in combination with machine learning methods to improve the robustness and accuracy of crack detection.

This study presents a method for the detection and quantification of cracks in masonry surfaces. The objective of this study is to algorithmically capture and metrically describe the length and width of cracks. To achieve this, a photogrammetric point cloud of the damaged scene is first generated. The 3D points are projected onto the image plane, where semantic segmentation is performed. For this purpose, a YOLO11 network was trained using the dataset provided by Flotzinger et al. (2023). Subsequently, a methodology was developed to link cracks observed from different camera perspectives to identify them as continuous structures. For quantification, crack widths are extracted from the segmentation masks, while crack lengths are computed using graph theory and the Dijkstra algorithm. The goal of this work is to develop a robust and accurate method for automated crack detection and analysis in masonry.

2. Methodology

This paper presents a method for the detection and quantification of cracks on masonry surfaces. The cracks are identified by using semantic segmentation of image data. By fusing the image information with 3D point clouds, a metric analysis of the cracks becomes possible. Moreover, the approach enables the examination of cracks that span across multiple individual images. Finally, the detected cracks are transferred into a three-dimensional model and visualized.

The following chapters follow a systematic structure in which the developed method is introduced step by step. An exemplary dataset serves as the basis for this description. In this dataset, images of a surface with visible cracks were taken. These images were processed photogrammetrically. The resulting 3D point cloud, the camera poses, and the images of the scene constitute the starting point for the analysis.

2.1 Instance Segmentation Using Neural Networks

Cracks are detected in the images using instance segmentation. For this purpose, a YOLO11 network was trained (Jocher and Qiu, 2024). The training was based on the publicly available DACL10k dataset by Flotzinger et al. (2023), which contains 6,935 training and 975 validation images. To optimize the training parameters, a hyperparameter tuning process was carried out on 50% of the dataset using a generic mutation strategy. Over the course of 300 training cycles, the parameters listed in the table below were slightly adjusted to identify optimal settings. Each cycle consisted of 40 training epochs with an input resolution of 640×640 pixels and a batch size of 30. The AdamW optimizer was employed for this purpose. The hyperparameter tuning provided by Jocher and Qiu (2024) was used.

lr0	0.00049	hsv_v	0.25851
lrf	0.00243	degrees	0.00000
momentum	0.84597	translate	0.21017
weight_decay	0.00016	scale	0.95000
warmup_epochs	5,00000	shear	0.00000
warmup_momentum	0.91574	perspective	0.00000
box	4.05077	flipud	0.00000
cls	0.49868	fliplr	0.48232
dfl	0.68652	mosaic	1.00000
hsv_h	0.01488	mixup	0.00000
hsv_s	0.12451	copy_paste	0.00000

Table 1. Hyperparameters that are used.

Lr0 determines the initial learning rate, this means how large the first optimization steps are. Momentum is a Stochastic Gradient Descent (SGD) dynamic parameter. It helps to keep the gradient direction stable and can accelerate convergence. Warmup_epochs ensures that the learning rate increases slowly at the beginning, while warmup_momentum regulates the initial behavior of the momentum. With weight_decay, a regularization is achieved that penalizes large weight values and prevents overfitting (Jocher and Qiu, 2024).

The parameters box and cls control the weighting of the individual loss components. Box refers to the object localization, cls to the classification of the detected objects. Data augmentation techniques are used to improve model robustness. Degrees means rotations of the image. Translate, scale, shear and perspective change the geometry by shifting, scaling, shearing or perspective distortion. Flipud and fliplr flip images vertically and horizontally. Color changes are achieved using hsv_h (hue), hsv_s (saturation) and hsv_v (brightness). More complex augmentations such as mosaic combine four images, while copy_paste inserts objects from one image into another. Mixup mixes images and labels to increase the variety of training. Lrf is the final learning rate and controls the reduction of the learning rate via the training time (Jocher and Qiu, 2024).

Based on the hyperparameters listed above, a YOLO11m-seg network was trained. Similar to the hyperparameter training, a resolution of 640640 pixels was used for resolution. The batch size was set to 30. Training was performed over 300 epochs. Apart from the parameters described above, and the hyperparameters listed in Table 1, the standard settings provided by Ultralytics were applied.

The trained network was evaluated using the validation dataset from (Flotzinger et al., 2023). An Intersection over Union of 0.301 was achieved for the crack class. In addition, a precision of 0.406, a recall of 0.576 and an F1 score of 0.420 were achieved.

2.2 Reprojecting 3D Point Cloud onto 2D Images

The presented method links three-dimensional point data with image information. This requires knowledge of both the intrinsic and extrinsic camera parameters. The determination of the intrinsic parameters can nowadays be accomplished using a wide range of available software solutions. The extrinsic parameters are determined simultaneously during the photogrammetric evaluation.

The 3D points of the point cloud are projected onto the image plane. The used camera model is a pinhole camera according to Zhang (2000), which is defined as follows:

$$p = A[R|T]P_w \tag{1}$$

where p = Pixels on the image plane
A = Internal camera parameters
R, T = Rotation and translation describe the change from the world coordinate system to the camera coordinate system
P_w = 3D coordinates in the world coordinate system

A 3D point in world coordinates is transformed into the camera coordinate system using rotation and translation. This transformation is defined as:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{2}$$

where X_c, Y_c, Z_c = 3D coordinates in the world coordinate system
X_w, Y_w, Z_w = 3D coordinates in the camera coordinate system

Using this approach, the entire point cloud is transformed into the camera coordinate system. The point cloud may contain irrelevant points, for example points located behind the camera or occluded by other objects in the scene.

Points located behind the camera can be identified and excluded based on their position along the Z-axis of the camera coordinate system. Since this axis represents the viewing direction, all points with a coordinate component $Z_c < 0$ are outside the visible area and are therefore discarded. This ensures that only points in front of the camera are considered.

In addition, points that are within the camera's field of view but occluded by other objects in the scene must also be removed. For example, a wall or another obstacle may block points behind it from being visible. To eliminate such occluded points, a visibility check is performed based on the method described in Katz et al. (2007). To minimize computational effort, this analysis is not performed on the entire point cloud, but only on those points that lie within the camera's field of view.

The resulting filtered 3D points in the camera coordinate system can then be projected onto the image plane using the intrinsic camera parameters.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3}$$

where u, v = Pixels on the image plane
f_x, f_y = Focal Length
c_x, c_y = principal point

Within the projected image points, there are also points that exceed the image dimensions. These points will be removed. When points are removed, the 2D–3D correspondences are kept at all times. This is particularly important for subsequent processing steps.

2.3 Classification of the point cloud and determination of the crack width

The method described in the previous chapter forms the foundation for the subsequent analysis. The damage analysis is image-based and begins with semantic segmentation of crack segments in the images. For this purpose, the previously trained YOLO11 model is used. For each detected crack segment, a binary mask is created. The mask is used to check which of the corresponding 3D points, described in Chapter 2.2, are located within the segmentation area.

The 3D points identified within a mask are stored along with a camera ID. When this procedure is applied to all images of the photogrammetric scan, it results in a classification of the entire point cloud into the categories "crack" and "no crack". The following image 1 shows the classification of the photogrammetric point cloud. The 3D coordinates of the point cloud, which are located within the segmentation masks of cracks, are colored red.

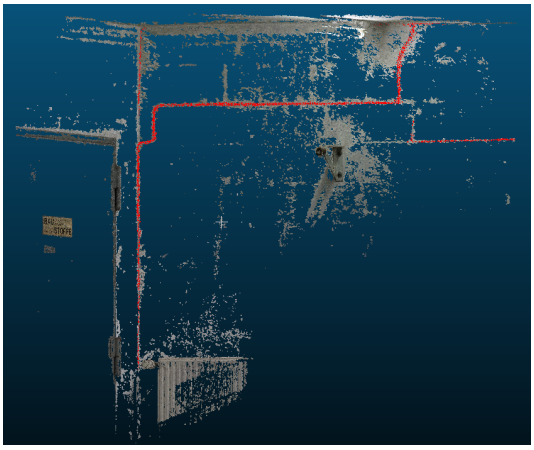

Figure 1. Classified point cloud.

In addition to classification, the width of the crack segments is analyzed using the binary masks of the segmented cracks. To minimize image artifacts and noise, a morphological erosion is first applied to the mask. Afterward, a skeletonization according to the method of Zhang and Suen (1984) is performed to extract the central axes of the cracks. The shortest distance to the boundaries of the binary mask is then calculated based on these central axes. The median of the distances, determined from the center axis to the edges of the binary masks, is defined as the crack width. This prevents local outliers or irregularities in the crack structure from being overestimated.

A scaling factor is required to describe the determined crack widths in metric units. This is calculated by the correspondence to the 3D coordinates of the point cloud. The calculated crack widths are based on the segmentation masks produced by the neural network. Since these masks tend to be wider than the actual cracks, the computed widths are multiplied by a correction factor of 0.25.

The skeletonization of the crack segments is also used to refine the corresponding 3D coordinates. To keep the amount of data low and to minimize noise in the 3D coordinates of the cracks, only those coordinates are taken which are located near the central axes of the crack segments. This means that only 3D coordinates are used for which the reprojected 2D pixels are no further than 20 pixels from the skeletonization.

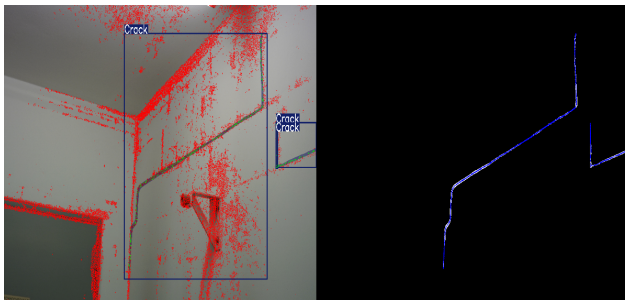

Figure 2. Reprojection of the 3D point cloud onto the image plane with semantic segmentation of cracks and skeletonization of the binary masks.

Figure 2 illustrates the previously described procedure. On the left side of the image, the reprojected 3D points are visualized in red, with only every 30th point shown for clarity. Also visible in the left section are the segmentation masks generated by the neural network. 3D points located within these masks are highlighted in green. The right side of the figure shows a binary representation of the segmentation results. Within the detected crack areas, the extracted skeletons are shown in blue, representing the medial axes of the individual cracks.

Up to this point, image data has been required for the analysis. In the following steps, this is no longer necessary. All results obtained so far are stored in a separate JSON file. In this file, an individual camera view is created for each image. Each camera view may contain multiple crack segments. For each segment, the following parameters are documented: crack width in pixels, converted crack width in meters, as well as the 2D and 3D coordinates of the crack segments. The results of the following analyses, in which connected crack instances are recognized, and the crack lengths are determined from these, are also saved within this JSON file. For this purpose, the file will be updated by adding the relevant parameters.

The results are saved in a JSON file in the following format:

```
{"Crack detection": {
  "Camera view 1": {
    "Crack number 1": {
      {"Wide_px":"[]"},
      {"Wide_m":"[]"},
      {"Coordinates_2D":"[]"},
      {"Coordinates_3D":"[]"}}
    "Crack number n": {}}
  "Camera view n": {}}}
```

Listing 1. Structure of the JSON file for crack detection.

2.4 Detection of Connected Crack Segments.

Up to this point, the crack segments in the image data have been considered in isolation. The following section analyzes if these segments belong to a continuous crack or represent independent instances. For this purpose, the previously described JSON file is processed, and the contained camera views along with their associated crack segments are analyzed. The analysis is based on the 3D coordinates of the individual crack segments.

The first crack segment of the JSON file is selected as a new crack instance. All other segments are then compared with the instances already defined. The comparison checks if the minimal distance between the 3D coordinates of a segment and the points of existing instances is below a predefined threshold value. If this is the case, the segment is assigned to the corresponding instance. If a segment cannot be assigned to any existing instance, it is treated as a new crack instance. For the dataset presented in this example, a threshold of 5 cm was defined.

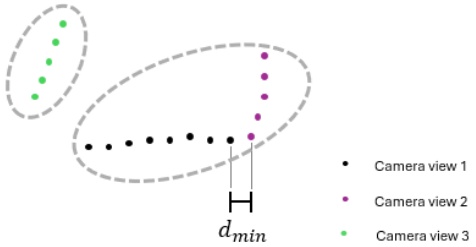

Figure 3. Analysis of connected crack segments. The points represent crack coordinates. They are colored depending on the captured camera. If a minimum distance between the individual segments is determined, they are grouped as one crack instance. This is visualized by the gray dashed ellipse.

where $\mathrm{d}_{min} = \min_{\substack{i=1,\dots,n \\ j=1,\dots,m}} \left\| \vec{a}_i - \vec{b}_j \right\|$

Figure 3 illustrates this methodology. For clarity, the approach is explained using a two-dimensional representation. In this example, the distance between the point coordinates of camera views 1 and 2 is below the defined threshold d_{min}, so the corresponding segments are grouped as one crack instance. The distances of the point coordinates of camera view 3 are not below this threshold. Therefore, these points are defined as a separate, new crack instance.

The widths of the individual crack segments have already been determined in chapter 2.3. To determine the crack width of an instance, all associated crack segments are evaluated. The maximum width within these segments is defined as the crack width of the respective instance. Each detected crack instance is saved in the JSON file. The 3D coordinates of the cracks, the crack width and the camera IDs from which the crack was detected are also saved.

2.5 Estimating Crack Length Using Graph Theory

Up to this point, cracks have already been identified as coherent objects, and their width could be determined using the segmentation masks. In the following step, the length of each individual crack is analyzed. This length estimation is performed within the three-dimensional point cloud, where crack instances are considered sequentially.

Determining the crack length presents two main challenges. First, cracks can appear highly branched, requiring the identification of the longest possible continuous path within an instance. Second, the 3D coordinates in the point cloud, although close to the actual crack centerline, exhibit a certain degree of noise. A direct reconstruction of the centerline using regression methods proves difficult due to the unstructured nature of the point cloud.

For this reason, an alternative approach is chosen. The point cloud is modelled as a graph, in which the nodes represent the 3D points of the crack instance, and the edges represent

neighboring connections. Using Dijkstra's algorithm, the path between potential start and end points of the crack structure is calculated to enable a robust estimation of the crack length (Hagberg et al., 2008).

To determine potential start and end points of a crack, its geometric boundaries within the point cloud must first be identified. This is done by selecting the points with extreme coordinate values i.e., those that exhibit the minimum or maximum values along at least one of the three spatial axes (x, y, z). These extremal points are defined as potential start and end points of a crack instance and are required for the subsequent length analysis.

The graph used for length determination consists of nodes and edges. The nodes represent the 3D coordinates of the points in a crack instance, while the edges connect each node to its ten nearest neighbors. Each edge is assigned a weight corresponding to the Euclidean distance between the connected points. This network serves as the basis for calculating the crack length. Using Dijkstra's algorithm, the shortest path between two nodes in the graph is computed. Such a path is composed of a sequence of defined edges traversing the connected nodes.To determine the actual start and end points of a crack, all combinations of the previously identified extremal points are evaluated. For each combination, the shortest path is computed. The combination resulting in the longest traveled distance within the graph is ultimately defined as the start and end point of the crack. The nodes passed represent the 3D coordinates of the main crack axis. The sum of the edges involved results in the total length of the crack.

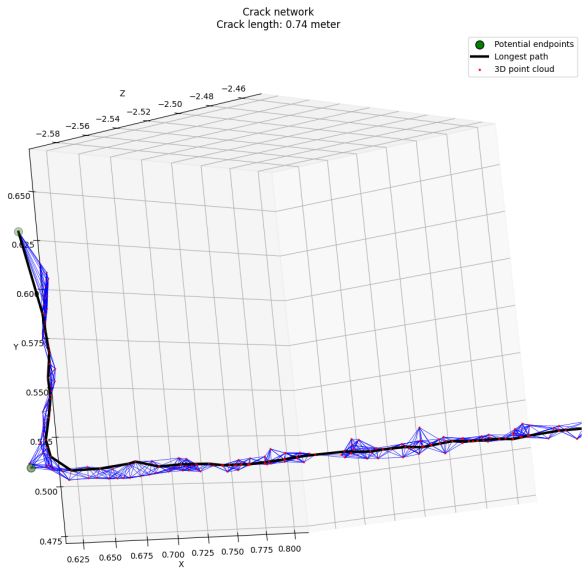

Figure 4. Crack network to calculate the crack length using the Dijkstra algorithm. The edges of the graph network are displayed in blue. The red points are the nodes of the graph and represent the segmented 3D coordinates of a crack. The determined longest crack path is visualized in black.

Figure 4 illustrates the process described above using an example crack section. The red points represent the 3D coordinates of the crack and simultaneously serve as the nodes of the underlying graph. The blue lines visualize the edges of the graph network, each connecting neighboring nodes. The green-highlighted points indicate the identified extremal points, which are considered as potential start and end points for the path calculation. The black path shows the shortest path determined between the selected extremal points and forms the basis for estimating the actual crack length.

The calculated crack path is centered within the 3D coordinates of a crack, despite the existing noise. At the same time, the model is accurate enough to follow the structural properties of the crack. Local changes in direction, such as the curve in the illustration, can be captured reliably.

3. Conclusions and Future Work

The methodology presented in this paper enables the detection and quantification of cracks. Cracks are semantically segmented in image data and combined with three-dimensional information. To illustrate the process, the method was demonstrated using a photogrammetrically generated point cloud. However, the methodology is not limited to photogrammetric data. It can also be applied to data from laser scanners, provided they are equipped with integrated camera technology. The use of a laser scanner for generating the 3D point cloud offers the advantage of a higher quality data basis.

To validate the proposed methodology, the results of the algorithmic evaluation were additionally verified through manual analysis. For this purpose, 6 different scenes with a total of 12 cracks were recorded. The results are presented in Table 2. It is evident that crack length determination is reliable. Cracks can be correctly detected even when they exhibit multiple directional changes. Discrepancies between manual and automated evaluations can be primarily attributed to insufficient crack detection in the image data.

	Algorithmically calculates	
	Crack length [m]	Crack width [mm]
Crack no.1	3,673	2,50
Crack no.2	0,742	0,80
Crack no.3	0,429	0,60
Crack no.4	1,668	1,00
Crack no.5	0,593	0,40
Crack no.6	5,530	3,70
Crack no.7	3,210	4,90
Crack no.8	1,232	1,60
Crack no.9	0,530	0,80
Crack no.10	0,240	0,50
Crack no.11	1,418	2,30
Crack no.12	4,508	0,80
	Manually calculated	
	Crack length [m]	Crack width [mm]
Crack no.1	3,699	1,10
Crack no.2	0,733	0,30
Crack no.3	0,452	0,10
Crack no.4	1,669	0,40
Crack no.5	0,599	0,40
Crack no.6	5,527	0,50
Crack no.7	3,212	1,20
Crack no.8	1,232	0,30
Crack no.9	0,515	0,20
Crack no.10	0,252	0,30
Crack no.11	1,429	0,30
Crack no.12	4,532	0,60

Table 2. Comparison of algorithmic and manual crack analysis. Mean deviations of the determined lengths, Ø 1.09 cm. Mean deviation of the determined widths, Ø 1.88 mm.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W6-2025-113-2025

Larger discrepancies were found in the measured crack widths. A possible cause is the current method for determining crack width, which relies on the segmentation masks generated by the neural network. These masks may significantly deviate from the actual crack geometry. Future work could focus on improving crack representation within the segmentation masks. A possible approach would be the integration of traditional image processing techniques, such as threshold-based methods, to further refine crack masking in image data.

However, when interpreting the comparison of crack widths, it is important to consider that crack width itself varies significantly along the length of a crack. In the presented example, the width of Crack No. 2 ranges from 0.3 mm to 1.8 mm.

The manual validation of crack lengths was conducted directly in the point cloud, while crack widths were determined using a crack width map. In future work, the proposed methodology will be tested on a test object using a laser scanner. Cracks will be professionally assessed to enable a valid statement regarding the accuracy of the proposed method.

The following illustrations show the damaged scenes with the cracks that were used for the evaluation.

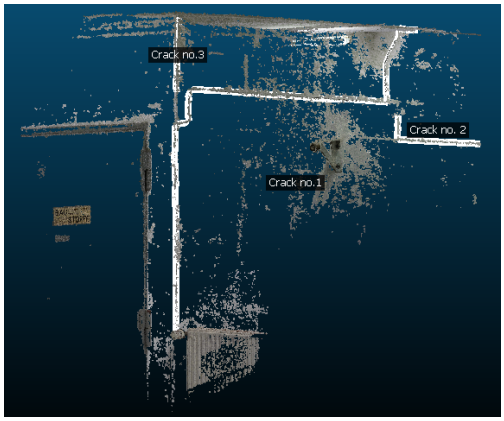

Figure 5. Manual analysis of cracks no. 1 - 3.

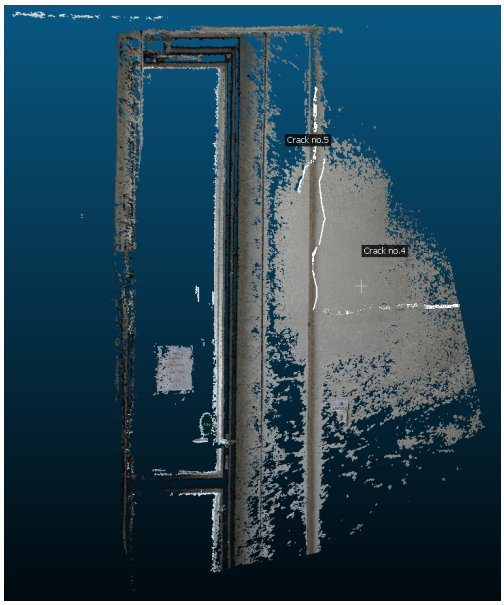

Figure 6. Manual analysis of cracks no. 4 & 5.

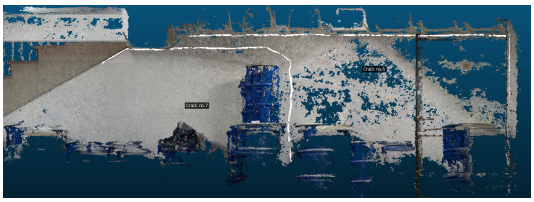

Figure 7. Manual analysis of crack no. 6 & 7.

Figure 8. Manual analysis of crack no. 8.

Figure 9. Manual analysis of cracks no. 9 - 11.

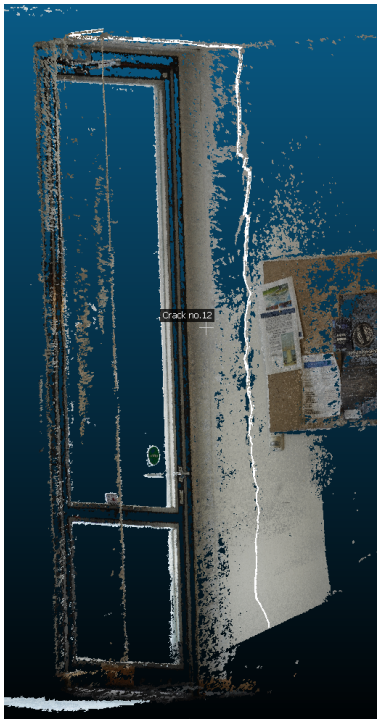

Figure 10. Manual analysis of crack no. 12.

The results presented in this paper demonstrate that cracks can be reliably quantified using the proposed methodology. However, the accuracy of the process strongly depends on the successful detection of cracks in the image data. Future work will explore alternative network architectures alongside the YOLO11 architecture used in this study to evaluate their suitability for semantic segmentation of structural damage. A comparison with a U-Net or DeeplabV3+ architecture could provide interesting insights. Additionally, the goal is to expand the range of detectable damage types. The neural network, which is used to segment the cracks in the image data, was trained based on the DACL10k dataset. This already contains 19 types of damage in the field of reinforced concrete construction. This means that, in addition to cracks, damage such as spalling, moisture damage, corrosion, exposed reinforcement bars, damage to the steel structure and damage to the glass structure will also be included in the future.

One advantage of the proposed methodology is that damage does not necessarily need to be detected in every individual image. Since structures are typically captured from multiple perspectives, it is sufficient if damage is reliably identified in at least one image. This approach is particularly crucial for damage analysis in glass structures, where defects are often visible only from specific viewpoints, requiring detection from multiple angles. Another key benefit is the tight integration of 3D data with image data. This allows the investigation of cracks that extend across multiple images. This is especially useful for fine-structured cracks where the camera must be very close to the crack itself.

This study focuses on the detection and quantification of cracks. The analysis results are stored in a structured format within a JSON file. Future research aims to integrate this information into a digital twin of the examined object. In this digital representation, detected damages will be documented and supplemented with additional information, such as surface material. This approach also enables long-term damage monitoring and time-resolved analyses.

Additionally, future research will evaluate alternative or complementary sensor technologies beyond conventional camera technology. A particular focus will be placed on the use of thermal cameras, which can reveal temperature-induced anomalies such as moisture ingress. The integration of several sensors could improve the detection and quantification of structural damage. This makes the system more robust against different environmental conditions.

References

Azhari, F., Sennersten, C. C., Milford, M., Peynot, T., 2021. PointCrack3D: Crack Detection in Unstructured Environments using a 3D-Point-Cloud-Based Deep Neural Network. *ArXiv*, abs/2111.11615.

Flotzinger, J., Rösch, P. J., Braml, T., 2023. dacl10k: Benchmark for Semantic Bridge Damage Segmentation. *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 8611-8620.

Flotzinger, J., Rösch, P. J., Oswald, N., Braml, T., 2022. Building inspection toolkit: Unified evaluation and strong baselines for bridge damage recognition. *2022 IEEE International Conference on Image Processing (ICIP)*, 1221–1225.

Hagberg, A. A., Schult, D. A., Swart, P., Hagberg, J., 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. *Proceedings of the Python in Science Conference*.

Hu, K., Chen, Z., Kang, H., Tang, Y., 2024. 3D vision technologies for a self-developed structural external crack damage recognition robot. *Automation in Construction*, 159, 105262.

Jahanshahi, M. R., Masri, S. F., 2012. Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures. *Automation in Construction*, 22, 567-576. Planning Future Cities-Selected papers from the 2010 eCAADe Conference.

Jocher, G., Qiu, J., 2024. Ultralytics yolo11.

Katz, S., Tal, A., Basri, R., 2007. Direct visibility of point sets. 26.

Lawin, F. J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F. S., Felsberg, M., 2017. Deep projective 3d semantic segmentation. *Computer Analysis of Images and Patterns: 17th International Conference, CAIP 2017, Ystad, Sweden, August 22-24, 2017, Proceedings, Part I 17*, Springer, 95–107.

Liu, Z., Li, X., Li, J., Teng, S., 2022. A New Approach to Automatically Calibrate and Detect Building Cracks. *Buildings*, 12, 1081.

Majidi, S., Omidalizarandi, M., Sharifi, M. A., 2023. INTELLIGENT 3D CRACK RECONSTRUCTION USING CLOSE RANGE PHOTOGRAMMETRY IMAGERY. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, X-4/W1-2022, 443–450.

Mohan, A., Poobal, S., 2018. Crack detection using image processing: A critical review and analysis. *Alexandria Engineering Journal*, 57(2), 787-798.

Wang, J., Wang, P., Qu, L., Pei, Z., Ueda, T., 2024. Automatic detection of building surface cracks using UAV and deep learning-combined approach. *Structural Concrete*, 25(4), 2302-2322.

Xu, H., Su, X., Xu, H., Li, H., 2019. Autonomous bridge crack detection using deep convolutional neural networks.

Yadhunath, R., Srikanth, S., Sudheer, A., Jyotsna, C., Amudha, J., 2021. Detecting Surface Cracks on Buildings Using Computer Vision: An Experimental Comparison of Digital Image Processing and Deep Learning. *Advances in Intelligent Systems and Computing*.

Yang, J., Lee, C., Ahn, P., Lee, H., Yi, E., Kim, J., 2020. Pbp-net: Point projection and back-projection network for 3d point cloud segmentation. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8469–8475.

Yiğit, A. Y., Uysal, M., 2024. Automatic crack detection and structural inspection of cultural heritage buildings using UAV photogrammetry and digital twin technology. *Journal of Building Engineering*, 94, 109952.

Zhang, T. Y., Suen, C. Y., 1984. A fast parallel algorithm for thinning digital patterns. *Commun. ACM*, 27, 236-239.

Zhang, Z., 2000. A Flexible New Technique for Camera Calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22, 1330 - 1334.