PointCrack3D: Crack Detection in Unstructured Environments using a 3D-Point-Cloud-Based Deep Neural Network*

Faris Azhari¹,², Charlotte Sennersten³, Michael Milford¹ and Thierry Peynot¹,²

Abstract—Surface cracks on buildings, natural walls and underground mine tunnels can indicate serious structural integrity issues that threaten the safety of the structure and people in the environment. Timely detection and monitoring of cracks is crucial to managing these risks, especially if the systems can be made highly automated through robots. Vision-based crack detection algorithms using deep neural networks have exhibited promise for structured surfaces such as walls or civil engineering tunnels, but little work has addressed highly unstructured environments such as rock cliffs and bare mining tunnels. To address this challenge, this paper presents PointCrack3D, a new 3D-point-cloud-based crack detection algorithm for unstructured surfaces. The method comprises three key components: an adaptive down-sampling method that maintains sufficient crack point density, a DNN that classifies each point as crack or non-crack, and a post-processing clustering method that groups crack points into crack instances. The method was validated experimentally on a new large natural rock dataset, comprising coloured LIDAR point clouds spanning more than 900 m² and 412 individual cracks. Results demonstrate a crack detection rate of 97% overall and 100% for cracks with a maximum width of more than 3 cm, significantly outperforming the state of the art. Furthermore, for cross-validation, PointCrack3D was applied to an entirely new dataset acquired in different location and not used at all in training and shown to detect 100% of its crack instances. We also characterise the relationship between detection performance, crack width and number of points per crack, providing a foundation upon which to make decisions about both practical deployments and future research directions.

I. INTRODUCTION

Cracks on structures, whether human-made (e.g. pavements, buildings, bridges, tunnels) or natural (e.g. cliffs, rock surfaces, caves), can be defined as the structure that breaks a surface continuity with a gap. They are signs of potential weakness, and can ultimately lead to catastrophic disasters such as collapses and rockfalls. For example, in an underground mining environment, surface cracks (also known as fractures) are one of the early signs of geological failures that can lead to rockfalls or even collapse of the roof [1], [2]. Over a 10-year period (2010 - 2019), in the U.S.A. alone, 49 deaths and 3,359 non-fatal accidents were reported as a result of geological failures [3]. This toll accounts for over 25% of the total death toll in underground mines. Current mitigation techniques include routine visual inspections that pose a risk to the safety of the geotechnical engineer, especially when surveying voids that are abandoned, newly blasted or post-ore extraction. Automated inspection systems using robots could eliminate this risk and increase survey frequency, resulting in earlier detection. This work focuses on the development of a key component of this system: automated crack detection. This detector should be capable of: detecting crack instances (so they can be counted), determining their position (globally and relatively to each other) and characterising them (e.g. estimating their width and length).

A large body of research has investigated the detection of cracks in structured environments, on mostly flat or smooth surface, using vision e.g. [4]. However, as noted in [5], in imagery of an unstructured surface, shadows of a protruding structure can easily be confused with an actual crack. Prior work has shown that the performance of state-of-the-art (SOTA) image-based crack detection methods is significantly compromised when applied to a shotcrete-layered unstructured surface in an underground mine site [5], even though the surfaces in this case remain relatively smooth thanks to the shotcrete. In this paper we aim to detect cracks in significantly more challenging scenes, such as highly unstructured bare rock surfaces, see Fig. 1 and Fig. 2.

To address the challenge, the automated crack detection method proposed in this paper exploits LIDAR data to capture accurate geometric information. The properties of most cracks - a thin opening that covers a relatively small area on the surface - makes for a challenging problem as point clouds in the crack region can be sparser compared to neighbouring flat contiguous surfaces. Inspired by promising early results obtained using DNN methods, this paper proposes PointCrack3D (PtCrack3D), a point-cloud-based DNN approach for crack detection on highly unstructured surfaces.

*This work was supported by Australian Government Research Training Program (RTP), Queensland University of Technology (QUT) through the QUT Centre for Robotics and Mining3. 
¹QUT Centre for Robotics, Brisbane, Australia. faris.azhari@connect.qut.edu.au
²Mining3, Pinjarra Hills, Australia.
³CSIRO Mineral Resources, Pullenvale, Australia.

Fig. 1. Scanning an unstructured surface with cracks using a 3D LIDAR scanner with the coloured point cloud and labelled crack (red) overlaid.
PtCrack3D comprises three key components: an adaptive down-sampling method that maintains sufficient crack point density, a DNN that classifies each point as crack or non-crack, and a post-processing clustering method that groups crack points into crack instances.

The proposed method was trained and validated experimentally on a new dataset of a large natural rock cliff, on which it is shown to detect 97% of the crack instances overall, and 100% of the cracks at least 3 cm wide, thereby strongly outperforming a state-of-the-art point-cloud based defects detection method. Furthermore, the paper proposes a cross-validation of the method, where PtCrack3D applied ‘out of the box’ is shown to detect 100% of the crack instances of an entirely new dataset acquired in different location and not used at all in the training.

The rest of the paper is structured as follows. Sec. II discusses the relevant literature. Sec. III outlines the crack detection method and implementation followed by Sec. IV which evaluates the performance of the approach. Finally, conclusions and future work are discussed in Sec. VI.

II. RELATED WORK

The automation of crack detection has been a topic of wide interest that follows along with the advancement and affordability of perception sensors such as cameras and LIDARs. In addition, with the advancement of Deep Neural Network (DNN) methods, utilising DNN architectures has become the SOTA in detecting cracks [4], [6], [7], [8]. However, existing research has focused on detecting cracks on structured surface i.e. mostly flat or with a known geometry such as on pavements [4], [7], [8], concrete [6], [7], [4] or tunnels [9] using monocular vision. These vision-based methods have shown promising results, but have been seen to struggle when applied to more challenging environments where the surface is unstructured [5]. 3D sensors have also been utilised for crack and defects detection, however, they have been applied on similar structural domains such as pavements [10], [11], concrete [12], [13], [14], timber [15] and aeroplane exterior [16].

To the best of the authors’ knowledge no prior work achieved LIDAR-based crack detection method in an unstructured environment using deep learning. Hence, in this section, relevant literature discussed concerns: (a) point cloud-based defects detection systems; (b) DNN architecture for point cloud.

A. Point cloud-based defects detection methods

Point cloud-based crack detection has not been explored as much as its image-based counterpart. Hence, this section explores defects detection, a more general application domain. In [12], [14], the point clouds of concrete surfaces are projected to a 2D plane producing range images and are used for detection, for example, applying DNN method on the images [17]. Other methods include grid search [10] and plane fitting [15] which have been applied on pavement and timber point clouds, respectively. Defects on aeroplane exteriors have been detected using region growing on the surface normal estimation [16]. In [18], Model Point Feature Histogram (MPFH) uses local surface descriptors paired with a machine learning method to detect defects on a variety of point cloud surfaces (e.g. welding, artificial teeth, ceramics) where one of the defects are surface cracks. MPFH was reported to achieve an average accuracy of 94% in the best configuration. In these applications, the surfaces are flat or at least a priori known and could be easily modelled, unlike the highly unstructured surfaces in this work.

B. Deep neural network architecture for point cloud

PointNet [19] was the first DNN method that performs convolution directly on the raw point cloud input without the need of any point cloud representation such as voxelisation or projection. Further works based on PointNet was done such as adding local region computation [20], [21]. Based on local region computations, [22], [23] added a module that computes the correlation between the extracted local regions and achieved better performance compared to methods without it. Additionally, implementations such as [24], [25], [26] were proposed for fast large-scale point cloud segmentation. However, these methods were never tested on large imbalanced datasets in an unstructured environment where the object of interest is small compared to the scene and represented by points in low density.

Point cloud-based DNN methods have been successfully applied for semantic segmentation on real-world data, for example, segmenting ground, buildings and vegetation of aerial point cloud [27], indoor and outdoor object segmentation in [28], [29], respectively. However, point cloud DNN has not been applied to segmentation problems with high class imbalance.

III. POINTCrack3D

This paper proposes PointCrack3D, a point-cloud-based crack detection algorithm for unstructured surfaces using a DNN architecture. Given a point cloud of an unstructured surface, the objective is first to classify each point into either crack or non-crack and then to cluster crack points into their respective crack instances to perform crack instance detection. The system consists of three components: (a) Data preparation; (b) DNN architecture; (c) Point clustering.

A. Pre-Processing

Given a point cloud of a scanned surface, the raw data is first discretised into voxels. The point clouds contained in each voxel are then sampled down to meet the required DNN input resolution followed by normalisation of the point coordinates and its corresponding feature set.

The data preparation process is governed by the parameter \([d, n, s]\), where \(d\) is the dimension of the voxel, \(n\) is the number of points inside a voxel and \(s\) is the stride at which the voxel moves in space, where \(s = d\) means no overlapping between neighbouring voxels. Each voxel that exceeds \(n\) number of points is then down-sampled using an adapted voxel grid-based method as shown in Algorithm 1. Respectively to the nature of cracks points on a surface point cloud,
this down-sampling method preserves the sparse points that represent a crack compared to applying random sampling method which leads to the diminishing of sparse crack points. This discretisation step (i.e., voxelisation followed by down-sampling) preserves the geometrical features of cracks on a large surface compared to using a whole down-sampled point cloud directly as an input.

Finally the voxelised points and their respective features are normalised within the range of [0,1] with respect to the entire training dataset which finally gives the input as:

\[ V_{i=1...n} = P_j \{X_j, F_j\}_{j=1...n} \]  

where \( v_n \) is the number of voxels, \( X_j \) and \( F_j \) are the normalised coordinate, \( \{x_j, y_j, z_j\} \) and the normalised features of the \( j \)-th points, and \( n \) is the number of points in a voxel. This gives an input size of \( n \times m \) where \( m \) depends on the dimension of features included, \( m = 3 + \text{dim}(F) \).

**Algorithm 1: Modified voxel grid sampling**

**Input:** Point cloud = \( pc \), No. desired points = \( n \)  
**Output:** Down-sampled \( pc = pc_{out} \)  
grid = floor(\( \sqrt[3]{m} \));  
v = voxelise(\( pc, \text{grid} \));  
while size(\( v \)) < \( n \) do  
grid ++;  
v = voxelise(\( pc, \text{grid} \));  
end  
if size(\( v \)) > \( n \) then  
v_\( n \) = choose [size(\( v \)) - \( n \)] randomly from \( v \);  
foreach \( i \in \text{index}(v_\( n \)) \) do  
\( v_\( n \) = \text{choose a random voxel neighbour of} \ [v_\( n \)] \neq v_\( n \];  
v_\( n \)(\( i \)) = [v_\( n \)(\( i \)) \neq v_\( n \]);  
end  
end  
foreach \( i \in \text{index}(v) \) do  
c = mean(\( v_{pc}(i) \));  
\( pc_{out}(i) = \min(\text{dist}(v_{pc}(i), c)) \);  
end

\[ h(P_j|z_j) = \frac{1}{1+\exp(-z_j)} \]

where \( z_j \) is the output corresponding to \( P_j \) from the final dense layer of the architecture. \( h(P_j|z_j) \) has a value [0,1]. To generalise, the notation \( H_i \) i.e., \( H_i \) for the confidence level of each point in the \( i \)-th voxel and \( H_S \) for dataset \( S \), will be used throughout this paper to denote the output from the DNN architecture given the input dataset \( S \).

**C. Point Clustering**

The output from the DNN only tells us how confident the prediction is on each point in a particular voxel but does not make any inference on whether the points are an instance of a single crack. Grouping points into clusters belonging to the same instance of crack enables the extraction of measurements of the cluster of points i.e. crack such as length, width, or volume.

The proposed post-processing follows these steps: (1) Reconstruct surface point clouds from voxels. (2) For each surface point cloud, take points \( p \) with \( H_p \geq H_i \) (i.e. points with confidence level greater or equal to \( H_i \)), where \( H_i \) is a pre-defined threshold. Points with \( H_p < H_i \) are labelled as non-crack. (3) The points in \( p \) are grouped into clusters, where two points are considered to be from the same cluster if their distance is less than a set threshold \( \Delta_i \). (4) For each cluster on a surface, reject clusters that have fewer points than a set threshold \( \Delta_n \) and assign the points as non-cracks. For the remaining clusters, assign a similar value to each point of a cluster, with distinct values across clusters.

**IV. IMPLEMENTATION**

This section describes how PtCrack3D was validated which includes the nature of the dataset, hyperparameters, hardware and the metrics used to evaluate the performance of the proposed crack detection system.

**A. Dataset**

To date, there is no publicly available dataset collected for the purpose of evaluating crack detection on an unstructured surface, hence, a new data collection was collected. The
Kangaroo Point cliffs (about 18 m high) located in Brisbane, Australia was chosen as the surfaces are unstructured and contain many cracks. A FARO® Focus3D x330 HDR LIDAR scanner was used to scan the cliffs section-by-section (between 15 to 30 m each). The scanning setting of the LIDAR scanner was set at a constant value of a resolution of 20,480 vertical points per one complete resolution and a scan quality of 6/8. This decision was made due to the scanner’s capability (i.e. would not allow maxing out both quality and resolution) based on visually inspecting and deciding the balance needed for the point cloud to be able to capture the geometry of cracks. In total there are 21 scans that include the RGB values and the intensity of the reflected beam for each point. Each point was then labelled by hand into two categories: crack and non-crack, hence, a point \( p_j \) contains \( P_j \{ x_j, y_j, z_j \}, E_j \{ r_j, g_j, b_j, i_j \} \{ g_i \} \) where \( \{ x, y, z \} \) are the position relative to the sensor, \( \{ r, g, b, i \} \) are the RGB information and intensity respectively, while \( g_i \) is the added labelled ground truth.

In total, there are 412 cracks across KP dataset. Two-thirds of the cracks are randomly assigned for training while the rest are for testing. For the training set, surface points that are within 15 cm from the edges of a crack are included as negative. The restriction on the number of surface points included increases the crack points representation to 10%. The training set is then split randomly into training and validation with a ratio of 2:1. The remaining surface points are included in the testing set. In the testing set, crack points represent 0.08% of the total points.

The point clouds are then voxelised, \( \text{voxel}_d,n,s \) using \( d = 0.5 \text{ m} \) and \( n = 2048 \). These numbers are chosen based on experimentation where it gives favourable results while balancing the appropriate density of voxels. As for \( s \), different values were tested, \( s = \{ d, 0.35 \times d, 0.30 \times d, 0.25 \times d \} \). For the testing set, \( s \) is set equal to \( d \). Voxels with less than \( n \) points are discarded from each set. The system was also tested on varying input feature combination. During training, each input point in each voxel is perturbed by 0.001 \( \text{ x } \text{ m} \) limited to the range \([-0.005,0.005]\) where \( r \) is a value randomly sampled from a standard normal distribution.

### B. Hyperparameters

The hyperparameters used for the PointConv architecture are as shown in Table I. Additionally, the bias of the last layer i.e. Dense(1), of the DNN architecture (Fig. 2) is initialised to \( \log(N_{\text{pos}}/N_{\text{neg}}) \) where \( N_{\text{pos}} \) and \( N_{\text{neg}} \) is the number of crack points and number of non-crack points in the training dataset respectively. As for the focal loss parameters, all possible combinations of \( \gamma \) and \( \alpha \) values were tested.

| Hyperparameters | Value |
|-----------------|-------|
| Optimiser       | Adam \((\beta_{1/2} = (0.9, 0.999), \epsilon = 10^{-4})\) |
| Epochs          | 101   |
| Learning rate, \( lr \) | 0.01 |
| Learning rate decay | \( lr = 0.5 \times lr \) after every 10 epochs |
| Loss function   | focal loss \( = (\gamma, \alpha) \) |
| Focal Loss parameters | \((0.2, 0.4, 0.5, 10.25, 50, 75, 90)\) |
| Batch normalisation | true |
| Batch size      | 5     |

where true positives \((TP)\) are the number of actual crack points classified as crack, false negatives \((FN)\) are the number of actual crack points classified as non-cracks and false positives \((FP)\) are the number of surface points classified as cracks. These metrics give the performance of the detector at the per-point level, however, it does not capture the performance of the detector at per-crack level (crack instance detection). Hence, we define the rate of crack instances detected, \( \text{cr}_\text{det} \) as:

\[
\text{cr}_\text{det} = \frac{1}{N_{\text{cr}}} \sum_{n=1}^{N_{\text{cr}}} \left\{ \begin{array}{ll}
1, & \text{if any } \text{cr}_\text{pred} \cap \text{cr}_\text{real} \\
0, & \text{otherwise}
\end{array} \right. 
\]

(4)

where \( N_{\text{cr}} \) is the total number of cracks in the dataset, \( \text{cr}_\text{pred} \) is an instance of detected crack and \( \text{cr}_\text{real} \) is an instance of a real crack. In Eq. 4 the condition \( \text{cr}_\text{pred} \cap \text{cr}_\text{real} \) (i.e. a detected crack instance intersects a real crack instance) is only true if the \( \{ \text{number of intersected points} \} \geq \{ \alpha \} \) number of points in \( \text{cr}_\text{pred} \), where \( \alpha \) is a number between \([0,1]\). It is also true if one or more \( \text{cr}_\text{pred} \) intersects one \( \text{cr}_\text{real} \), hence this metric is paired with crack continuity:

\[
\text{cr}_\text{cont} = \frac{1}{N_{\text{cr}}} \sum_{n=1}^{N_{\text{cr}}} \frac{1}{N_{n,\text{cr}_\text{pred} \cap \text{cr}_\text{real}}} \sum_{n=1}^{N_{\text{cr}}} \sum_{n=1}^{N_{\text{cr}}} \left\{ \begin{array}{ll}
1, & \text{if any } \text{cr}_\text{pred} \cap \text{cr}_\text{real} \\
0, & \text{otherwise}
\end{array} \right. 
\]

(5)

where \( N_{\text{cr}} \) is the total number of real cracks in the dataset and \( N_{n,\text{cr}_\text{pred} \cap \text{cr}_\text{real}} \) is the number of predicted crack instance that satisfies the condition \( \text{cr}_\text{pred} \cap \text{cr}_\text{real} \) for the \( n \)th crack. This metric measures the fragmentation of the detection on a crack. Finally crack precision, \( \text{cr}_\text{pre} \) is defined as:

\[
\text{cr}_\text{pre} = \frac{N_{\text{cr}_\text{pred} \cap \text{cr}_\text{real}}}{N_{\text{cr}_\text{pred}}} 
\]

(6)

where \( N_{\text{cr}_\text{pred} \cap \text{cr}_\text{real}} \) is the number of predicted crack instance that satisfies the condition \( \text{cr}_\text{pred} \cap \text{cr}_\text{real} \) and \( N_{\text{cr}_\text{pred}} \) is the total number of predicted crack instances in the dataset. This metric gives the ratio of crack clusters detected to the total clusters detected. Each metrics generates a value between \([0,1]\) where a value closer to one indicates better performance. Note that these metrics can only be evaluated after the post-processing step as the output from this step are clusters of points where each cluster is considered as an instance of detected cracks.

### D. Training Execution

PtCrack3D was trained and tested on an Intel i7-8700K, 64GB RAM with NVIDIA GeForce GTX 1080 Ti graphics card. The training set used was augmented by translating the
point cloud 10 times along randomly picked axes by a value randomly generated from within [0, 0.50] m. The results shown in this section are based on the validation set. The goal of the experimentation was to determine the best DNN configuration and parameters for the crack detection system.

The first test was to discover the best $\gamma$ and $\alpha$ values for the focal loss function. For this test, the position $(x, y, z)$ of the point was used as the only feature input. While a number of combinations were used (as in Table I), only tests showing promising results were included in Table II. The configuration with the highest F1 score was chosen. To further justify the use of focal loss and the chosen parameters, the focal loss result was compared to the result using binary cross-entropy as shown in Table II – last row. The F1 score and recall value of the focal loss was higher than of the binary cross-entropy, however, binary cross-entropy performed better in terms of precision which is common when there is a class imbalance towards negative samples. By applying focal loss, during training, less weight is given to easy sample points i.e. non-crack and vice versa. For the application of crack detection where failure to detect cracks could lead to disastrous consequences, a higher recall value is more desirable, whilst compromising the precision.

We evaluated the effect of using different combination of point features, see Table III. The results show that adding colour information ($rgb$) on top of position-only $(xyz)$ increases the performance in both precision and recall by about 10% and 20% respectively. This is because the structure of cracks often makes them appear darker than their surrounding surface. Adding the LIDAR intensity $i$ to $xyz$ increases performance by almost 5%, however, adding $i$ on top of $rgb$ does not seem to alter the performance as it is likely that $rgb$ already captures the information useful in discriminating cracks.

Finally we tested the impact of varying stride lengths using the original training set i.e. without the translation augmentation, see Table IV. Increasing the overlapping increases the performance in all aspects. This is due to the network having more chance of learning a certain feature multiple times in different positions in a voxel. However, having lower stride length increases the number of input voxels, hence increasing the time required to complete the training of 101 epochs. As in Table IV – 3 last rows, an increase of 5% overlap caused the training duration to almost double.

V. EXPERIMENTAL RESULTS

The detection performance of PtCrack3D was evaluated using the model trained with voxel stride of 0.125 m,
and compared to SOTA defects detection, MPFH [18] as mentioned in Sec. II. For a fair comparison, MPFH was fine-tuned to our dataset. For our MPFH implementation, a neural network was used after the feature extraction step. The rgb values of each point were also included as a feature on top of the extracted ones, adding 3 additional features for the neural net. This addition of RGB values increased the overall performance of MPFH with an absolute increase of around 7% for precision and recall during training.

### A. Pointwise Classification

Fig. 4 shows PtCrack3D per-point classification performance (in solid lines). At threshold, $\Delta_H = 0.5$, PtCrack3D obtained a recall score of 76%, specificity of 92% and a precision of 0.7%. The low precision score is due to the large class imbalance where small percentage of surface points falsely detected as crack points will exponentially decrease the precision score.

In comparison, MPFH, shown in dotted-lines in Fig. 4 at $\Delta_H = 0.5$, achieved a recall score of 8%, which is 68% lower than PtCrack3D in absolute. The precision is consistent at 0.5% across all threshold values while ours scored 13% maximum when the threshold is $\Delta_H = 0.74$.

### B. Crack Detection

This section focuses on the detection of crack instances, after the post-processing step of PtCrack3D. Table IV shows the results for different confidence thresholds and post-processing configurations. For our method, we found that $\Delta_H = 0.59$ provides the most relevant balance between precision and recall, given the application. In this configuration the method detected 97% of the cracks in the test dataset, even though only half of the crack points were classified as belonging to a crack. For MPFH, the results are shown for $\Delta_H = 0.005$ and $[\Delta_r, \Delta_n] = [0.04, 15]$ where the performance was best (both precision and recall are at their maximum). MPFH was only able to detect 47% of the cracks instances. PtCrack3D clearly outperforms MPFH, even with the addition of colour, across all metrics. Increasing the threshold $\Delta_H$ to 0.65 for our method, leads to a similar recall to that obtained with MPFH, but with a much higher precision and still a significantly higher crack detection rate of 78%.

Further evaluation of our method (with $\Delta_H = 0.59$) was performed by analysing the detection outcome against the width and the number of points of each crack instance, see Fig. 5. The width of a crack was measured by taking the shortest distances between the points on the opposite edges of the crack at a constant interval across its length. This was done manually from the labelled point cloud. Fig. 5 shows that the proposed method successfully detected 100% of the cracks with a maximum width of 0.03 m or above. In contrast, MPFH detected 91% of the 11 cracks with a maximum width above 0.06 m and only 61% of the 57 cracks with a maximum width above 0.03 m.

PtCrack3D was able to detect 100% of crack with more than 500 points, which is similar to MPFH. As for cracks with 500 points and below, PtCrack3D managed to detect 97% of them while MPFH only detected 40%. Having more points for a small crack allows for more relevant information to be extracted compared to small cracks with fewer points.

At $\Delta_H = 0.65$ and 0.005 for PtCrack3D and MPFH, where the per-point recall scores are almost similar, PtCrack3D was able to detect more cracks compared to MPFH. This is because our method is more consistent in the ability to detect crack points across each crack (i.e. being able to detect about the same percentage of points across all cracks) while MPFH detects more points on easier cracks.

| ΔH | PtCrack3D | MPFH |
|----|-----------|------|
| $[\Delta_r, \Delta_n] = [0.02, 0.20]$ | 0.59 | 0.005 |
| precision | 0.013 | 0.015 |
| recall | 0.73 | 0.26 |
| specificity | 0.95 | 0.99 |
| $c_d$ | 0.99 | 0.99 |
| $c_m$ | 0.84 | 0.93 |
| $c_p$ | 0.0075 | 0.027 |

### TABLE IV

**Point-wise detection performance with varying voxel stride.**

| Voxel Stride | Precision | Recall | F1 Score | Duration (d,h,m) |
|--------------|-----------|--------|----------|------------------|
| 0.300 [90%] | 0.45      | 0.47   | 0.46     | 0.0229           |
| 0.250 [60%] | 0.50      | 0.50   | 0.50     | 0.1537           |
| 0.175 [65%] | 0.60      | 0.60   | 0.57     | 1.1038           |
| 0.150 [70%] | 0.63      | 0.63   | 0.60     | 2.0745           |
| 0.125 [75%] | 0.67      | 0.67   | 0.63     | 4.0231           |

Fig. 4. Performance curves of PtCrack3D on Kangaroo Point (KP) and Coolum Beach (CB) dataset and MPFH applied on KP.

### TABLE V

**Performance of crack detection methods. $\Delta_H =$ confidence threshold, $[\Delta_r, \Delta_n] = [\text{distance threshold, point threshold}].$
(i.e. cracks with larger width or points) and failed to detect any points on harder cases.

Qualitative analysis of the results shows that our system was able to detect cracks with sufficient points for characterisation. As shown in Fig. 3, where the coloured points i.e. blue, red and cyan are TP, FN and FP crack points classified by the method. The FN are scattered in groups of small numbers where the blue clusters are sufficient to extract crack characteristics such as length. However, PtCrack3D also has the tendency to detect sharp-concaved edges as a crack as can be seen in numbers on the right-hand side of the full surface in the left-most image of Fig. 3 with a close-up view of a sample on the left-most image.

C. Cross-validation

A second dataset was collected to test the trained model on a different scene. Scans of unstructured rock surfaces of Coolum Beach (see Fig. 1), located 120 km north of Kangaroo Point, were captured using the same LIDAR scanner and configuration. The rock formations are between 1.5−3 m in height and 3−5 m in width which allows for high density point cloud across the surfaces. In total, 5 scans were captured covering a total surface area of approximately more than 100 m² with 35 cracks representing about 7% of the total points with width ranging from 0.5 cm to 35 cm.

1) Implementation: PtCrack3D was applied on the entire Coolum dataset without any re-training, with the same implementation and parameters as in Sec. IV except for the post-processing step. The $[\Delta_r, \Delta_n]$ parameters of the point clustering were set to [0.05, 300] to account for the larger dimensions of the cracks and higher point density due to the sensor being closer to the scanned surface.

2) Results and Discussions: PtCrack3D managed to detect all of the cracks in the dataset in terms of crack-wise performance. Note that the smallest crack is about 0.5 cm and 0.7 cm in width at the smallest and largest opening, respectively. The high resolution of the smallest crack (i.e. length and points about 12 cm and 500 points, respectively) enables more geometrical information available for detection. For the detected cracks, PtCrack3D achieved a performance of $cr_{con} = 0.94$ and $cr_{pre} = 0.71$ which is on par with the Kangaroo Point dataset.

Fig. 4 shows a visualisation of the point cloud output on a surface sample from the Coolum dataset which represents the detection behaviour in general. The FPs (in cyan) are similar to the behaviour on Kangaroo Point dataset i.e. mostly around the edges of cracks and in corners with high changes of surface normals. Achieving to detect all crack instances, PtCrack3D underestimates the width of a crack with a large width as highlighted in the yellow box of Fig. 6. This is because the opening width is more than 35 cm at the widest point, whereas the largest opening on the dataset used to train PtCrack3D is only about 10 cm.

Fig. 6 shows the point-wise performance curves (in dashed-dotted lines) for the trained PtCrack3D method implementation on the Coolum dataset. A small absolute drop in recall and specificity (i.e. at threshold of 0.5) of about 14% can be observed compared to the Kangaroo Point test performance. The precision increased by about 20%. However, this increase in performance reflects the higher crack to non-crack point ratio of the Coolum dataset. Normalising the Coolum precision score by equalising its FP rate to the Kangaroo Point performance, shows an absolute drop of about 18% compared to the expected precision scores given by the differences in crack points ratio. The drop in performance across all measured point-wise metrics is expected as the model are trained using a different dataset. As mentioned earlier, it is more important to be able to detect cracks as an object, whereas the point-wise detection enables fine details measurements to be extracted from detected cracks.

VI. CONCLUSIONS

In this paper, we proposed PointCrack3D, a point-cloud-based crack detection method capable of automatically de-
detecting crack instances from 3D LIDAR data in highly unstructured surfaces such as rock cliffs or natural/mine tunnels. This is the first crack detection method utilising a point-based deep learning architecture for detection on unstructured surfaces. PtCrack3D includes a method which down-samples a point cloud while maintaining the point density of cracks and a post-processing step to segment cracks via clustering of 3D points corresponding to the detected crack instances.

PtCrack3D was first experimentally validated on a dataset of LIDAR point clouds capturing a large natural rock cliff. The method was shown to detect 97% of the crack instances in the test set overall, and 100% of cracks with a maximum width of more than 3 cm, where the SOTA MPFH defects detection method could only detect 47% (and 61%, resp.) of those cracks. PtCrack3D was then applied ‘out of the box’ to a second dataset acquired in a distinct location and managed to detect all cracks, indicating its detection capability on an out-of-sample case. The method was evaluated in different parameters configuration that could be fine-tuned to fit detection requirements. As an example, setting a high confidence threshold value may be suitable for applications where small cracks are less important to be identified whilst focusing on larger cracks and keeping false alarms low. It could also be concluded that the detection rate depends both on the width of a crack and the number of points that makes up the crack. Automatically detecting cracks in highly unstructured surfaces using PtCrack3D is a critical step towards the deployment of robots for autonomous crack inspection in an unstructured environment.

REFERENCES

[1] G. M. Molinda, C. Mark, and D. R. Dolinar, “Assessing coal mine roof stability through roof fall analysis,” Proc. of the new technology for coal mine roof support, vol. 9453, no. 151, pp. 53–72, 2000.
[2] T. Szwedzicki, “Rock mass behaviour prior to failure,” Int. J. of Rock Mechanics and Mining Sciences, vol. 40, no. 4, pp. 573–584, 2003.
[3] MSHA, “Mine Injury and Worktime Reports,” 2020. [Online]. Available: https://arweb.msha.gov/ACCINJ/accinj.htm
[4] F.-C. Chen and M. R. Jahan-shahi, “ARF-Crack: Rotation invariant deep fully convolutional network for pixel-level crack detection,” Machine Vision and Applications, vol. 31, no. 6, p. 47, 2020.
[5] F. Azhari, C. Senmersten, and T. Peynot, “Evaluation of Vision-based Surface Crack Detection Methods for Underground Mine Tunnel Images,” in Australas. Conf. on Robotics and Automation, 2019.
[6] C. V. Dung and L. D. Anh, “Autonomous concrete crack detection using deep fully convolutional neural network,” Automation in Construction, vol. 99, pp. 52–58, 2018.
[7] Q. Zhu, M. D. Phung, and Q. Ha, “Crack Detection Using Enhanced Hierarchical Convolutional Neural Networks,” in Australas. Conf. on Robotics and Automation, 2019.
[8] Q. Zou, Z. Zhang, Q. Li, X. Qi, Q. Wang, and S. Wang, “DeepCrack: Learning Hierarchical Convolutional Features for Crack Detection,” IEEE Trans. Image Processing, vol. 28, no. 3, Mar. 2019.
[9] R. Montero, E. Menendez, J. G. Victores, and C. Balaguer, “Intelligent robotic system for autonomous crack detection and characterization in concrete tunnels,” in ICARSC, Apr. 2017, pp. 316–321.
[10] M. Zhong, L. Sui, Z. Wang, and D. Hu, “Pavement Crack Detection from Mobile Laser Scanning Point Clouds Using a Time Grid,” Sensors, vol. 20, no. 15, July 2020.
[11] A. Zhang, K. C. P. Wang, Y. Fei, Y. Liu, C. Chen, G. Yang, J. Q. Li, E. Yang, and S. Qiu, “Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces with a Recurrent Neural Network,” Computer-Aided Civil and Infrastructure Eng., vol. 34, no. 3, pp. 213–229, Mar. 2019.
[12] X. Xu and H. Yang, “Intelligent crack extraction and analysis for tunnel structures with terrestrial laser scanning measurement,” Advances in Mech. Eng., vol. 11, no. 9, Sept. 2019.
[13] M. M. Sarkar, T. A. Ali, A. Abdelatifah, S. Yehia, and A. Elaksher, “A cost effective method for crack detection and measurement on concrete surface,” ISPRS, vol. XLII-2/W8, pp. 237–241, 2017.
[14] S. Cho, S. Park, G. Cha, and T. Oh, “Development of Image Processing for Crack Detection on Concrete Structures through Terrestrial Laser Scanning Associated with the Octree Structure,” Appl. Sci., vol. 8, no. 12, Nov. 2018.
[15] M. Cabaleiro, R. Lindenbergh, W. Gard, P. Arias, and J. van de Kuijlen, “Algorithm for automatic detection and analysis of cracks in timber beams from LiDAR data,” Const. Build. Mater., vol. 130, Jan. 2017.
[16] I. Jovančević, H.-H. Pham, J.-J. Orteu, R. Gilblas, J. Harvert, X. Maurice, and L. Bréthes, “3D Point Cloud Analysis for Detection and Characterization of Defects on Airplane Exterior Surface,” J. of Nondestruct. Eval., vol. 36, no. 74, 2017.
[17] Z. Allen, W. K. C. P. Yue, L. Yang, T. Suyu, C. Cheng, L. J. Q., and L. Baosian, “Deep Learning-Based Fully Automated Pavement Crack Detection on 3D Asphalt Surfaces with an Improved CrackNet,” J. of Computing in Civil Eng., vol. 32, no. 5, Sept. 2018.
[18] C. Madrigal, J. Branch, A. Restrepo, and D. Mery, “A Method for Automatic Surface Inspection Using a Model-Based 3D Descriptor,” Sensors, vol. 17, no. 10, Oct. 2017.
[19] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” in CVPR, 2017, pp. 77–85.
[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space,” 2017.
[21] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, “Pointwise Convolutional Neural Networks,” in CVPR, June 2018.
[22] W. Wu, Y. Li, X. Li, and L. Feng, “PointConv: Deep Convolutional Networks on 3D Point Clouds,” in CVPR, 2019, pp. 9613–9622.
[23] Y. Li, R. Bu, M. Sun, and B. Chen, “PointCNN : Convolution On X-Transformed Points,” arXiv:1801.07791, 2018.
[24] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds,” in CVPR, 2020.
[25] Z. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, “Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation,” in CVPR, 2021.
[26] K. Liu, Z. Gao, F. Lin, and B. M. Chen, “FG-Net: Fast Large-Scale LiDAR Point CloudsUnderstanding Network Leveraging Correlated-Feature Mining and Geometric-Aware Modelling,” arXiv:2012.09439, 2020.
[27] M. Soilán, R. Lindenbergh, B. Riveiro, and A. Sánchez-Rodríguez, “PointNet for the automatic classification of aerial point clouds,” ISPRS, vol. IV-2/W5, pp. 445–452, May 2019.
[28] J. Li, L. Fu, P. Wang, and C. Sun, “Indoor point cloud recognition with deep convolutional networks,” in Int. Conf. on Opt. Instrum. and Technol. SPIE, 2020, pp. 65–74.
[29] L. Ma, Y. Li, J. Li, W. Tan, Y. Yu, and M. A. Chapman, “Multi-Scale Point-Wise Convolutional Neural Networks for 3D Object Segmentation From LiDAR Point Clouds in Large-Scale Environments,” IEEE Trans. Intell. Transp. Syst., 2020.
[30] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” in ICCV, Oct. 2017, pp. 2999–3007.