mathematics

MDPI

Article

Research on Real-Time Detection Algorithm for Pavement Cracks Based on SparseInst-CDSM

Shao-Jie Wang [1,2], Ji-Kai Zhang [1,2,*] and Xiao-Qi Lu [3]

[1] School of Information Engineering, Inner Mongolia University of Science and Technology, Baotou 014010, China; wsj101@foxmail.com
[2] Key Laboratory of Pattern Recognition and Intelligent Image Processing in Inner Mongolia Autonomous Region, Baotou 014010, China
[3] School of Information Engineering, Inner Mongolia University of Technology, Hohhot 101051, China; wsjzll1213@outlook.com
* Correspondence: jkzhang0314@imust.edu.cn; Tel.: +86-180-4722-3826

Abstract: This paper proposes a road crack detection algorithm based on an improved SparseInst network, called the SparseInst-CDSM algorithm, aimed at solving the problems of low recognition accuracy and poor real-time detection of existing algorithms. The algorithm introduces the CBAM module, DCNv2 convolution, SPM strip pooling module, MPM mixed pooling module, etc., effectively improving the integrity and accuracy of crack recognition. At the same time, the central axis skeleton of the crack is extracted using the central axis method, and the length and maximum width of the crack are calculated. In the experimental comparison under the self-built crack dataset, SparseInst-CDSM has an accuracy of 93.66%, a precision of 67.35%, a recall of 66.72%, and an IoU of 84.74%, all higher than mainstream segmentation models such as Mask-RCNN and SOLO that were compared, reflecting the superiority of the algorithm proposed in this paper. The comparison results of actual measurements show that the algorithm error is within 10%, indicating that it has high effectiveness and practicality.

Keywords: intelligent transportation; road cracks; image segmentation; SparseInst algorithm; convolutional attention module

MSC: 68T07

Citation: Wang, S.-J.; Zhang, J.-K.; Lu, X.-Q. Research on Real-Time Detection Algorithm for Pavement Cracks Based on SparseInst-CDSM. *Mathematics* **2023**, *11*, 3277. https://doi.org/10.3390/math11153277

Academic Editors: Vitaly Kober and Tae Sun Choi

Received: 6 June 2023
Revised: 17 July 2023
Accepted: 24 July 2023
Published: 26 July 2023

1. Introduction

Highways are key infrastructure, serving as hubs for transportation and playing an irreplaceable role in people's daily lives. However, as investment in highways and other infrastructure in China increases year by year, many highways in service continue to deteriorate and become damaged, requiring regular monitoring and assessment of their condition. The primary method used in traditional bridge fracture detection is human measurement, with low efficiency, high missed detection rate, long time consumption, and high cost. Additionally, factors such as crack width and length take a while to calculate and process. Thus, automatic and effective fracture detection is crucial for determining the structural health of a bridge.

Early methods of detecting and maintaining the condition of the road surface generally relied on manual inspection, which was not only labor- and time-intensive, but also had a low detection accuracy and some dangers [1–3]. Scholars around the world have conducted a series of extensive and in-depth research using the latest scientific and technological developments to accurately and effectively extract crack information from images [4–6]. In 2014, Wang et al. [7] proposed a road surface crack extraction method based on valley bottom boundary; it generates findings for crack detection by applying a number of image processing methods. A crack connection technique for road surfaces was put forth by Liang

et al. [8] in 2015, and it is based on Prim's minimum spanning tree. The algorithm fills the cracks to create the crack structure. These conventional fracture-detecting techniques have clear drawbacks. Each technique is intended for a particular database or circumstance, but if the dataset or scenario changes, the crack detector will fail.

The threshold segmentation algorithm is one of the most basic methods in crack image segmentation, with the characteristics of less computation, simple operation, and robust performance. This algorithm can segment an image into black and white colors by extracting its grayscale value information. However, when dealing with images with weak contrast, threshold segmentation usually requires contrast enhancement first. In 1992, Kirschke et al. [9] proposed a road surface crack image segmentation algorithm based on the histogram, but it is only suitable for clear crack recognition. Subsequently, Oh et al. [10] proposed an iterative threshold segmentation algorithm, but it requires the manual setting of thresholds. Therefore, the threshold segmentation algorithm is suitable for road surface crack images with consistent background texture, uniform illumination, and high contrast.

Some traditional algorithms can be used to implement road crack detection, such as the improved K-Means algorithm proposed by Fang et al. [11]. Senthan et al. [12] proposed to use fuzzy Hough transform to detect cracks in road images, taking into account the fact that cracks are composed of nearly straight segments embedded in surfaces with considerable texture. The minimum path selection algorithm proposed by Rabin et al. [13] realizes automatic crack detection of two-dimensional road images. However, these algorithms require the manual setting and adjustment of parameters and have a strong dependence on human operation. In summary, edge-detection algorithms mainly judge whether it is a crack edge based on local gray and gradient information, which is only suitable for crack images with strong edge information, and tend to misjudge the background with strong edge information as cracks. When the noise is high, the effect of edge detection is poor.

Deep-learning techniques have been extensively applied to the identification and segmentation of road surface cracks in recent years [14–16]. By merging deep-learning techniques with road surface crack detection technology [17–19], this technology has significantly increased the efficiency and accuracy of road surface crack detection. However, due to the characteristics of high similarity between road surface cracks and background and small and irregular shapes of cracks in reality, accurate identification has always been a challenging problem. Improving the accuracy and timeliness of image crack extraction has become the focus of current research. Early CNN-based crack detection approaches essentially consist of two tasks: finding the detection target's bounding box and determining its category. These algorithms only identify the smallest bounding rectangle of the target crack in the image. Currently, window-based and region-based object detection methods are the two most often used approaches for finding objects. Window-based detection techniques use a fixed-size window to scan the image and a classifier to categorize each sliding window section. Cha et al. [20] used a window-based neural network to detect road surface cracks. The outcomes demonstrated that this method could gather crack information more precisely than conventional edge detection methods, although its detection accuracy was influenced by window width and length, which were challenging to measure. An increase in detection accuracy is hampered when the window size is too large and contains an excessive amount of irrelevant information. On the other hand, if the sliding window is too small, there will not be enough crack information in the sliding window region to establish whether or not there are any. This will reduce the detection accuracy.

Methods for detecting objects based on regions create candidate regions that initially employ region proposal approaches, establish regions of interest, and then carry out feature extraction. Due to its superior detection effect, Faster Region CNN (Faster R-CNN) has been utilized numerous times for fracture identification on road surfaces [21,22]. However, Maeda et al. [23] claim that they are only able to identify road fractures and cannot learn the maximum width, length, or area of a geometric feature. Attard et al. [24,25] discovered road damage using the most recent object identification techniques, Inception V2 and MobileNet. Research Mask R-CNN was also used to detect road surface cracks with

satisfactory results. However, the above methods have limited detection accuracy and cannot achieve pixel-level detection due to irregular crack shapes.

The area detection method can only locate cracks and cannot obtain the geometric dimensions of the cracks. Therefore, in order to achieve this goal, crack segmentation is necessary. Semantic segmentation, which categorizes each pixel to identify whether it belongs to a crack or the background, is one of the potential techniques being studied to increase the accuracy of road surface fracture detection. Zhang, L. et al. [26] used an improved CNN method that obtained better results compared to other traditional methods. Zhang et al. [27] adopted another efficient structure called CrackNet which has a strong anti-interference ability and can maintain stable detection results, having strong adaptability and robustness. However, these two methods cannot segment small cracks well because the traditional CNN has limitations in fine image segmentation; thus, a fully convolutional network [28,29] gradually began to be applied to road surface crack detection. In contrast to CNN, fully convolutional networks (FCN) may accept inputs of any size and utilize deconvolution to upsample the most recent feature map and restore it to the original input image's size, enabling prediction for each individual pixel.

More and more encoder–decoder model frameworks are being applied to CNN as people strive for improved detection accuracy. The decoder is a network used to gradually restore feature information, while the encoder is a classification network used to extract input features. Bang et al. [30] used a new encoder–decoder network with more layers and deeper network structure for the pixel-level identification of urban road surface cracks, and it is capable of identifying flaws in black box photos. For the quantification and detection of cracks, Ji et al. [31] utilized an integrated approach based on DeepLabV3+. Two-step convolutional networks were employed by Chun et al. [32] and J. Liu et al. [33] to first recognize and locate cracks before segmenting them. The most common CNN among the numerous encoder–decoder networks is named U-Net, according to Ronneberger et al. [34]. Its outstanding performance on medical photos drew the attention of numerous scholars for further study. Researchers in the field of civil engineering started using U-Net to find structural road surface cracks since the size and form of medical cells and cracks in the road surface differ. Although the U-Net network performs well in the field of crack detection, considering that future crack detection will be fully automated in real time, its further application will still be hindered by data volume increasing significantly, high computational cost, long training process, etc. [35,36]. The introduction of the SOLO model brings new ideas and methods to the field of instance segmentation [37]. The SOLO model adopts a new segmentation strategy that can more accurately segment targets and obtain high-quality segmentation results. Additionally, the SOLO model adopts a center-point-based target segmentation scheme, which can quickly and efficiently predict and segment instances in images. However, the segmentation effect of the SOLO model on small targets is not good because the SOLO model is an instance location-based segmentation algorithm, so there may be certain limitations for objects of different sizes. Especially for smaller objects, the SOLO model may not be able to capture enough location information, resulting in poor segmentation effect. The SOLO model was subsequently upgraded to the SOLOv2 model [38], introducing some new technologies to improve the model's segmentation performance and efficiency, such as using a distributed head network, mask feature pyramid, etc. However, SOLOv2 still has some limitations.

SparseInst is a real-time instance segmentation framework based on sparse instance activation maps [39]. Compared with traditional image-based or deep-learning-based methods, SparseInst has the following advantages: it adopts a sparse feature-based design that can significantly reduce computation and storage space without losing information, improving algorithm efficiency and robustness; relative to image-based methods, it has faster speed and higher accuracy in crack detection tasks. However, for small-sized cracks, SparseInst's segmentation effect is still not good, and under certain conditions, such as low resolution, uneven illumination, and large noise interference, the detection effect will be greatly affected.

Therefore, in view of the problems of low detection accuracy and easy background interference in related work, this paper improves the original SparseInst network in Chapter 2 by adding CBAM module [40], DCNv2 convolution [41] and introducing SPM stripe pooling structure and MPM hybrid pooling structure [42] to adaptively highlight object information areas, improve detection accuracy, and achieve the real-time accurate detection of cracks. In Chapter 3, the SparseInst-CDSM algorithm is used to extract clear crack images, extract the central axis skeleton of the crack through the central axis method, calculate its pixel length and width, and then use the formula to convert the pixel size into actual physical size according to current standards to judge whether cracks need maintenance, reducing labor costs and greatly improving work efficiency. Chapter 4 analyzes the experimental environment and experimental results, verifying the feasibility of the algorithm proposed in this paper.

2. Methodology

The algorithm in this paper is based on the SparseInst network and improves the framework to achieve crack morphology segmentation and extraction. As shown in Figure 1, the model mainly includes three components: the backbone, the encoder, and the IAM-based decoder. Given an input image, the backbone extracts multi-scale image features (i.e., C3, C4, and C5).

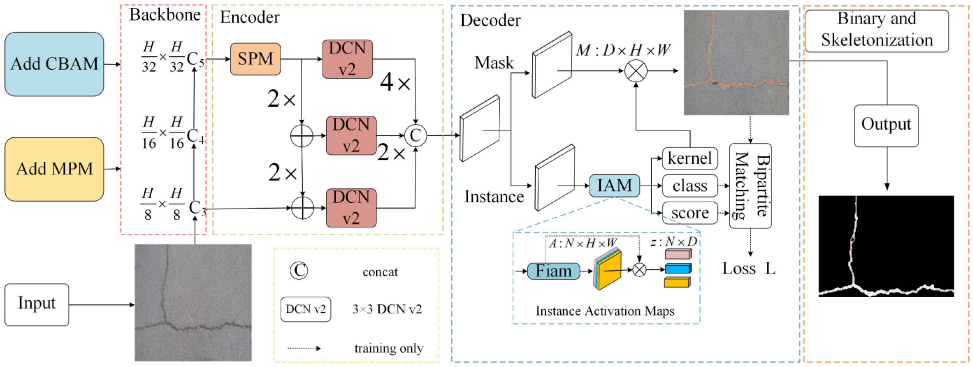


Figure 1. Improved SparseInst instance segmentation model.

The original SparseInst encoder uses a pyramid pooling module (PPM), and this paper replaces PPM with SPM to avoid establishing unnecessary connections between distant positions. In Figure 1, '4×' or '2×' indicates upsampling by a factor of 4 or 2. The IAM-based decoder consists of two branches, namely the instance branch and the mask branch. In the instance branch, the IAM module predicts instance activation maps (as shown in the right column) to obtain instance features {zi}N for identification and mask kernels. The mask branch aims to provide mask features M and multiply them with the predicted kernels to generate segmented masks.

2.1. Attention Mechanism Module

In order to enhance the feature extraction capability of the SparseInst backbone network, this paper adds the CBAM module, as shown in Figure 2, without destroying the structure of the feature extraction network.

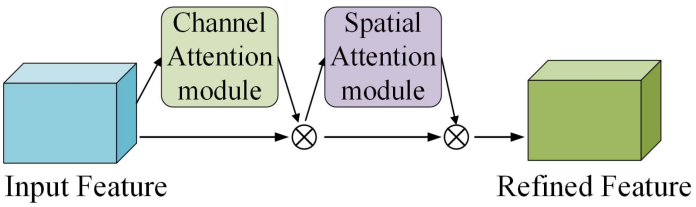
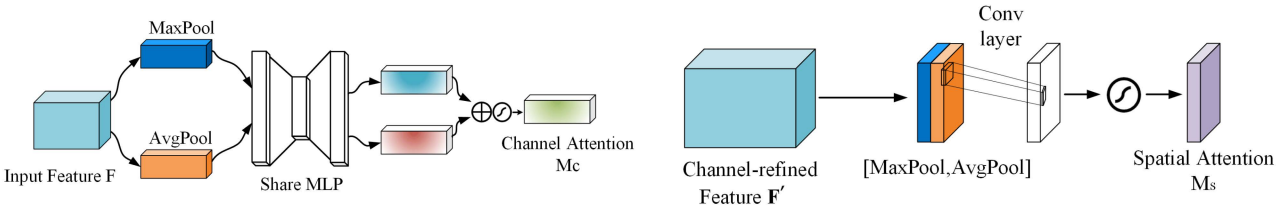


Figure 2. CBAM module structure.

This module first performs the maximum pooling process of the encoding part in the channel attention module, as shown in Figure 3a. The size of the input feature map is $H \times W \times C$, where H and W are the height and width of the feature map, respectively, and C is the number of channels of the feature map. Here, this paper first uses two pooling methods, MaxPool and AvgPool, to obtain two $1 \times 1 \times C$ feature maps, denoted as M and A, respectively. In order to obtain a weight coefficient between 0 and 1, the sigmoid function is used to combine the two feature maps, M and A, into two completely connected layers. This weight coefficient will be used to adjust the weight of the input feature map. The final output feature map is obtained by multiplying the weight coefficient with the input feature map. For the channel attention module, use the following formula:

$$\begin{aligned} M_s(F) &= \sigma(\mathrm{MLP}(\mathrm{AvgPool}(F)) + \mathrm{MLP}(\mathrm{MaxPool}(F))) \\ &= \sigma(W_1(W_0(F^c_{\mathrm{avg}})) + W_1(W_0(F^c_{\mathrm{max}}))) \end{aligned} \tag{1}$$



(**a**) (**b**)

Figure 3. (**a**) Channel attention structure; (**b**) spatial attention structure.

In this case, the shared MLP module in the channel attention module is represented by the MLP (Multilayer Perceptron). This module first compresses the number of channels before expanding them back to their original number. The Relu activation function yields the outcome of two activations.

After the channel attention module, the spatial attention module is introduced, and it should concentrate on which area of the space has more significant features, as shown in Figure 3b. Its input size is $H \times W \times C$. One channel dimension is subjected to max pooling and average pooling to produce two $H \times W \times 1$ feature maps, which are then combined in the channel. Next, the two feature maps are concatenated in the channel dimension, and $H \times W \times 2$ is the current feature map. Then, after passing through a convolutional layer, it is restored to one channel with a convolution kernel of 7×7. The final feature map, while maintaining HW constant, is $H \times W \times 1$. The sigmoid function is multiplied by the input feature map to produce the final feature map. The following equation describes the spatial attention module:

$$\begin{aligned} M_s(F) &= \sigma(f^{7\times7}([\mathrm{AvgPool}(F); \mathrm{MaxPool}(F)]) \\ &= \sigma(f^{7\times7}(F^s_{\mathrm{avg}}; F^s_{\mathrm{max}})) \end{aligned} \tag{2}$$

2.2. Deformable Convolution Module

When traditional CNN modules are used for visual recognition, there are some defects in fixed geometric structures that are difficult to avoid. Convolutional units, for instance, can only sample the input feature map at specific locations, whereas pooling layers diminish spatial resolution at a specific ratio. An ROI (Region of Interest) pooling layer also divides a ROI into fixed spatial units, and is devoid of an inherent mechanism for dealing with geometric alterations. These defects limit the efficiency and accuracy of traditional CNN modules in handling geometric transformations.

Therefore, adding deformable convolution can solve these problems to some extent because it can make different positions have different receptive field sizes and shapes to better adapt to the diversity of objects. In this way, the accuracy and robustness of convolutional neural networks can be improved so as to better handle various practical problems. This paper adds an offset to the traditional convolution operation of SparseInst,

as shown in Figure 4. It is this offset that makes the convolution deform into an irregular convolution. This offset can be a decimal and needs to be calculated using the method of bilinear interpolation.

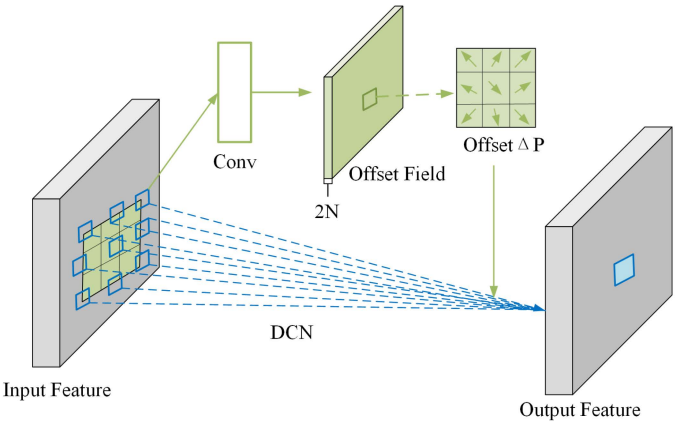


Figure 4. A 3 × 3 deformable convolution.

Deformable convolution DCN v1 will introduce some irrelevant areas and interfere with the extraction of features, reducing the performance of the algorithm; therefore, this paper adopts DCN v2. In DCN v2, not only is the offset of each sampling point added, but also a weight coefficient is added to distinguish whether the introduced area is an area of interest. If the area of this sampling point is not an area of interest, the weight is assumed to be 0. The formula is as follows:

$$y(p_0) = \sum_{p_n \in R} w(p_n) \cdot x(p_0 + p_n + \Delta p_n) \cdot \Delta m_k \tag{3}$$

2.3. Improved Context Encoder

In order to achieve faster inference speed, the original SparseInst used single-layer prediction. Considering the limitations of the single-layer features of various scale objects, the original SparseInst reconstructed the feature pyramid network and proposed an instance context encoder, as shown in Figure 5. In order to increase the receptive field and fuse features from P3 to P5, the instance context encoder employs the pyramid pooling module PPM [43] after C5. This improves the output single-stage features' multi-scale representation even further.

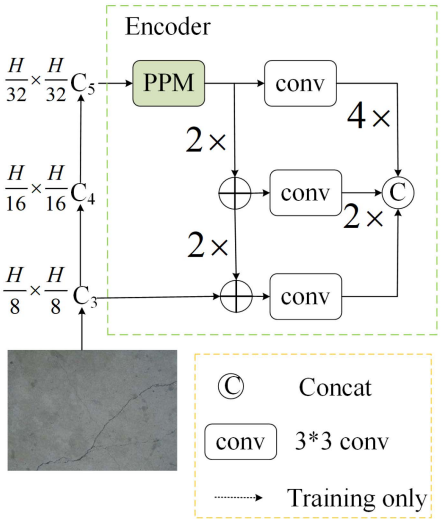


Figure 5. Encoder structure diagram.

However, the ability of PPM to utilize contextual information is limited because only square kernel shapes are applied. In addition, PPM is only modularized above the backbone network, so it cannot be flexibly or directly applied to the network building blocks of feature learning, and PPM heavily relies on standard spatial pooling operations, which can actually lose important information features in specific scenarios. In response to the problems with the PPM feature pyramid, this paper uses the strip pooling module (SPM) instead of the PPM module, as shown in Figure 1. To gather remote context from several spatial dimensions, SPM employs horizontal and vertical strip pooling procedures. Establishing long-range dependencies between discretely distributed regions and encoding regions in a strip shape using extended kernel shapes is simple when horizontal and vertical strip pooling layers are present.

At the same time, because the kernel shape of this algorithm is narrow in other dimensions, it emphasizes documenting regional specifics. These features distinguish the suggested SPM from conventional spatial pooling, which uses square kernels. Let the input a two-dimensional tensor, and in strip pooling, the pooling window is $(H, 1)$ or $(1, W)$. The biggest difference between strip pooling and average pooling is reflected here. Strip pooling averages all feature values in rows or columns. The formula is as follows:

$$y_i^h = \frac{1}{W} \sum_{0 \le j \le w} x_{i,j}, y_i^v = \frac{1}{H} \sum_{0 \le i \le H} x_{i,j} \tag{4}$$

Compared with the PPM module, the SPM module considers a narrow range instead of the entire feature map, avoiding unnecessary connections between distant positions, as shown in Figure 6 below. Let $x \in R^{C \times H \times W}$ be the input tensor, where C represents the number of channels; where x is first fed into two parallel channels, each of which contains a horizontal or vertical bar pooling layer; and where, after being modulated by a one-dimensional convolutional layer with a kernel size of 3, the current position and its surrounding features are determined. Define $y_{c,j}^h \in R^{C \times H}$ and $y_{c,j}^v \in R^{C \times W}$; then, $y \in R^{C \times H \times W}$ can be expressed as:

$$y_{c,i,j} = y_{c,i}^h + y_{c,j}^v \tag{5}$$

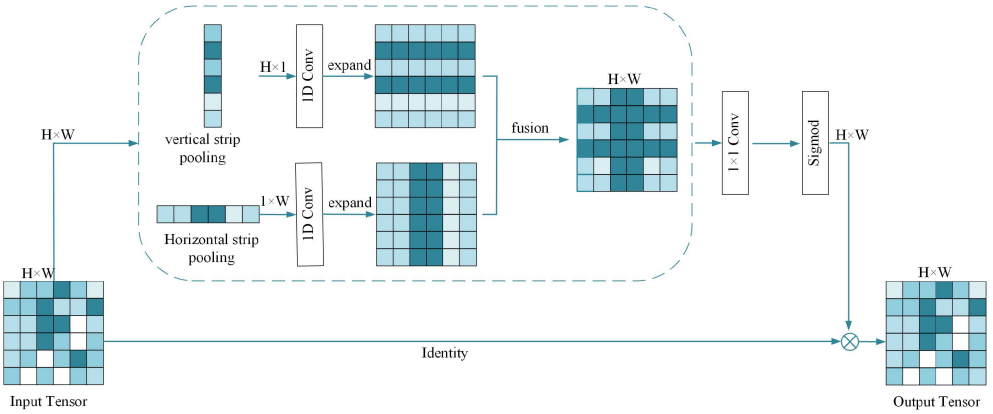


Figure 6. SPM Structure.

The final output is:

$$z = \text{Scale}(x, \sigma(f(y))) \tag{6}$$

Multiplication of elements is represented by Scale(.,.), σ represents the sigmoid function, and f represents 1×1 convolution. In order to improve the performance and efficiency of the model, the inner product of feature vectors is combined and calculated to make the model more lightweight. In addition, SPM is used to encode the horizontal and vertical information of the image and balance the weights of different parts to optimize the features.

This method greatly improves the global context information collection ability of the model, thus improving its performance.

2.4. Add Mix Pooling Module

In order to increase the distinctiveness of feature representations, the MPM module focuses on collecting multiple types of context information through various pooling methods. It is composed of two sub-modules: standard pooling and stripe pooling, as shown in Figure 7. Stripe pooling makes it possible to connect regions that are discretely distributed throughout the scene and encode areas with band-like structures, as shown in Figure 7a. However, for cases where semantic regions are closely distributed, spatial pooling is also required to capture local context information. With this in mind, as shown in Figure 7b, a lightweight pyramid pool sub-module is used to collect short-distance dependencies. It starts with two spatial pooling layers and then moves on to a convolutional layer for extracting multi-scale features and a 2D convolutional layer for retaining the original spatial information. The size of the merged feature maps after each merge is 20×20 and 12×12, respectively. All three sub-paths are then merged by summation. These two sub-modules can simultaneously capture short- and long-range dependencies between different positions and are essential for scene parsing networks.

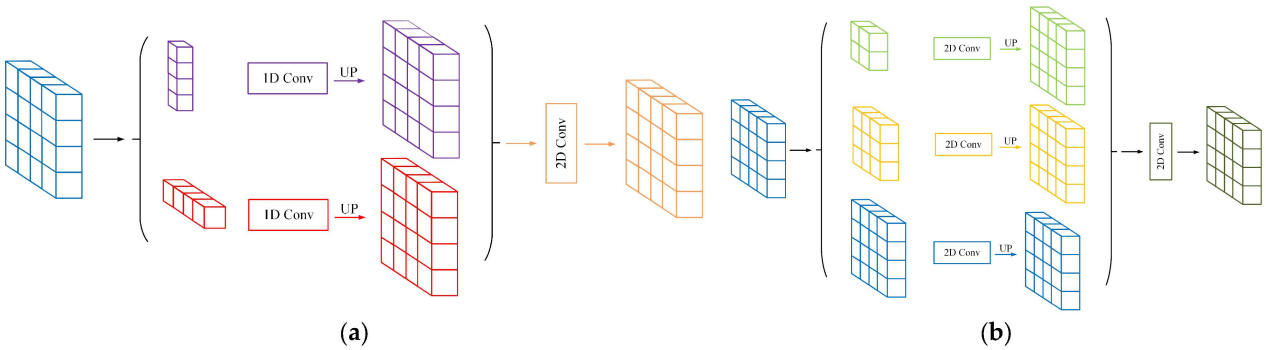


Figure 7. MPM structure of (**a**) long-distance dependency aggregation sub-module and (**b**) short-distance dependency aggregation sub-module.

Since MPM is modularly designed, it can be directly built on the backbone network, as shown in Figure 1. Since the output of the backbone network is 2048 channels, a 1×1 convolutional layer is first connected to the backbone network to reduce the output channels from 2048 to 1024; then, two MPMs are added. Each MPM uses 256 channels (i.e., a 1/4 reduction rate) for all convolutional layers with kernel sizes of 3×3 or 3. To forecast the segmentation map, a convolutional layer is finally implemented.

2.5. Preventing the Problem of Overfitting

Overfitting is a common problem in machine learning and statistical modeling, where the model overfits the training data, reducing its generalization ability. The model performs well on the training dataset but cannot generalize well on new datasets. The model focuses too much on the details and noise of the training data, ignoring the true characteristics of the data distribution, resulting in poor performance on new unseen data. Overfitted models are also more sensitive to outliers, noise, or errors in the input data. This means that even in the presence of slight interference or errors, the model may produce unreliable predictions. To address the problem of overfitting, this paper uses the following methods to avoid overfitting:

(1) Data Augmentation: By randomly transforming and augmenting the training data, the diversity of the training data can be increased. This can effectively reduce overfitting and improve the model's generalization to new images. The data augmentation operations used in this paper include random cropping, rotation, scaling, and flipping.

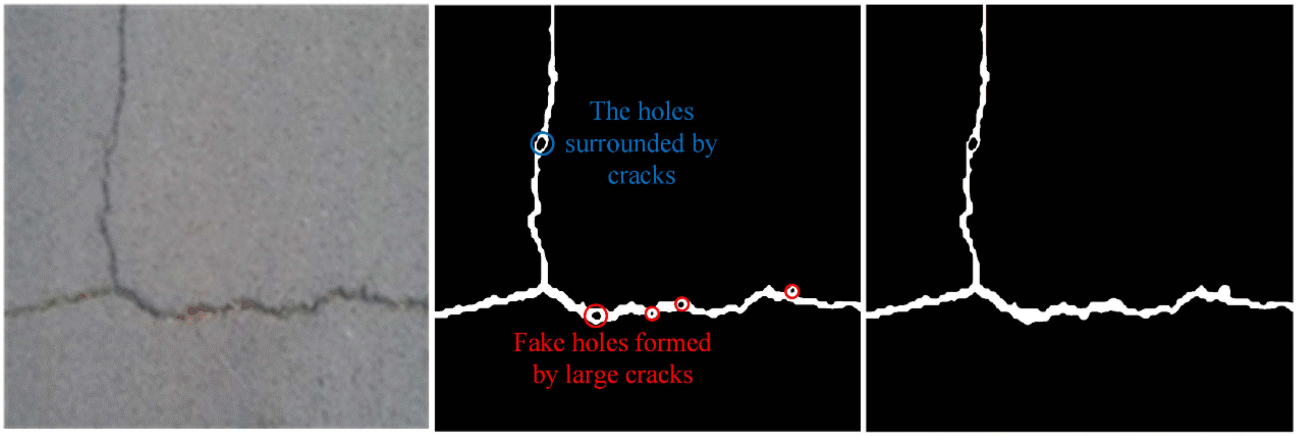
(2) Regularization: Regularization is a method of limiting the complexity of the model by introducing a regularization term into the loss function. Common regularization methods include L1 regularization and L2 regularization. In this paper, regularization is used to penalize large weight values in the model, thereby avoiding overfitting.
(3) Early Stopping: Early stopping is a simple and effective method to prevent overfitting. It monitors the performance metrics on the validation set and stops training before the model starts to overfit. Generally, when the performance on the validation set no longer improves, it can be considered that the model has reached its best generalization ability.

3. Crack Skeleton Extraction and Crack Size Calculation

3.1. Crack Morphology Skeletonization

In order to obtain crack segments from the crack edge mapping, it is necessary to merge the crack edges and represent them in a simplified manner; i.e., the area limited by the crack edges should be skeletonized. The resulting crack skeleton not only reflects the expansion of the crack but also greatly facilitates the positioning of the crack segments. Given that the crack skeleton is contained inside its two edges, the crack region is reconstructed using a morphological closure technique employing crack edge confidence mapping. In order to get rid of the crack skeleton, the fracture area is then narrowed [44–46].

As shown in Figure 8a, the space between the two edges is filled by morphological closure, which recreates a whole crack. Given that the crack's severity is not known, it is difficult to choose the appropriate shape and size of the structural element for morphological closure. A tiny structural component cannot effectively cover extensive crack regions, as evidenced by the red circular area in Figure 8b, resulting in false holes. Incorrect crack areas will result in extracted skeletons that do not match the expansion of the crack. In order to obtain the correct crack area, a small structural element is first used in order to patch the fracture, and then each hole is located using linked component analysis. The typical grayscale of the picture is situated between the gray levels of the road surface and the crack and can be chosen as a threshold to make a distinction between the two. In light of this, road surface images may be affected by uneven illumination, and the local average gray level of the area around the hole is used to distinguish between real holes and false holes. If a hole's average gray level exceeds the regional average value, it is a real hole; otherwise, it is a false hole that should be filled. As shown in Figure 8c, if the holes produced by small structural elements' morphological closure between crack pixels are filled in, the crack area is correct and can be used to extract skeletons.



(**a**) (**b**) (**c**)

Figure 8. Reconstruction of crack area (**a**) Original crack image; (**b**) False holes in large cracks; (**c**) The result after processing.

Skeleton extraction is a technique in the field of image processing and computer vision that aims to extract the skeleton information of objects from images. It is commonly used for tasks such as object analysis, recognition, and classification. This article uses the medial axis method to extract the medial axis of the crack skeleton, which is divided into the following steps:

(1) Convert the crack image into a binary image; that is, set the pixel values in the crack area of the image to white and other areas to black.
(2) Use the Canny algorithm [47] to perform edge detection on the binary image.
(3) The skeletonization algorithm proposed by Ma et al. [48] is used to skeletonize the binary image obtained by edge detection and extract the midline of the crack area.
(4) Connect the pixels on the medial axis to obtain the skeleton diagram of the crack.
(5) Post-process the crack skeleton diagram to remove redundant lines and fill in broken line segments.

These operations can eliminate noise and unimportant details in the image and retain key shape information. In order to obtain a one-pixel-wide crack skeleton, border pixels in the crack area must be continuously removed such that they no longer interfere with the crack's continuity. This procedure will retain redundant skeletons brought about by noise or minor edge changes. Figure 9 depicts the removal of the skeleton result of Figure 8c, where spurious junctions and superfluous skeletons are produced.

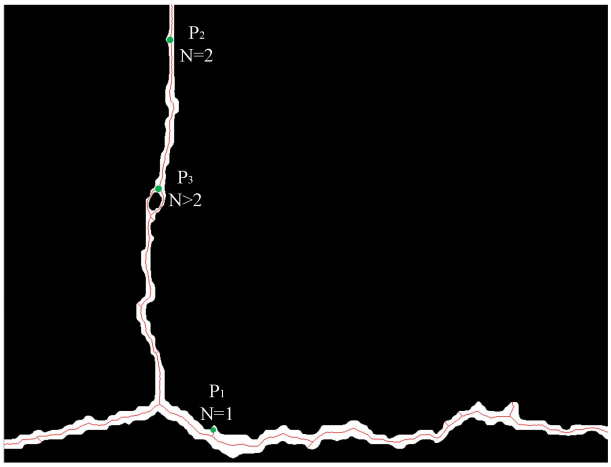


Figure 9. Redundant skeletons and false intersections.

They do not accurately depict the crack's fundamental topology. Therefore, after skeletonizing the crack, it is necessary to eliminate redundant skeletons and further refine the skeletonization results. Trim redundant skeletons through the subsequent actions:

(1) Because the duration of a curved crack should be significantly larger than its breadth, the maximum crack width specified in the precise crack quantification method of AASHTO PP67-10 is employed as the default cutoff for trimming [49].
(2) Track the eight neighbors of each skeletal pixel in a clockwise fashion. Let N stand for how many times a pixel's color switches from white to black. According to Figure 10, if N = 2, it is a typical skeletal pixel (P2); if N > 2, it is identified as an intersection (P3); if N = 1, the current pixel is an endpoint (P1).
(3) Begin at any endpoint and work your way along the skeleton until you reach another endpoint or intersection; then, part of the skeleton is documented. The skeleton must be pruned if its length is less than the standard pruning threshold since it is redundantly short. After pruning, a result will be obtained, as shown in Figure 10.

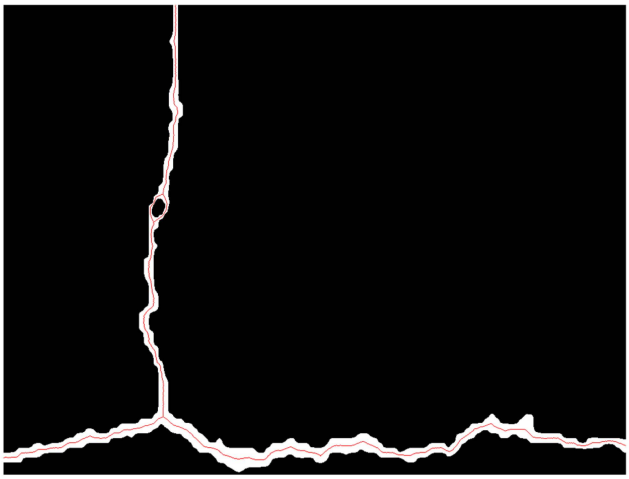


Figure 10. Result of pruning redundant skeletons.

3.2. Calculation of Crack Pixel Length

For cracks with branches, the following two methods are generally used to calculate their length:

(1) Using the total length calculation method, add the length of the main crack and its branch cracks to obtain the total length of the crack.
(2) Using the main crack length calculation method, only calculate the length of the main crack and do not calculate the length of the branch cracks. This method is suitable for situations where the branch cracks are short and dense.

The pixel length of the crack is calculated by obtaining a smooth and complete skeleton diagram. Since the crack has a bending process, a straight line mode is used instead of a bending mode. The idea is to measure each segment's length in the skeleton diagram, add them all together, and then subtract that number to obtain the crack's length. The diagram of the crack length calculation is shown in Figure 11.

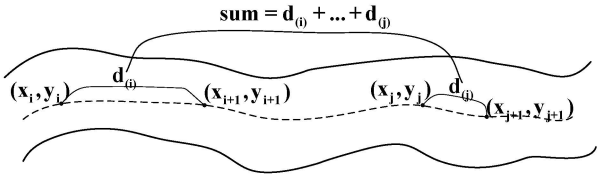


Figure 11. Crack length calculation method.

The calculation process is as follows:

(1) Obtain the n sets of target point coordinates between the starting point and the end point (x_i, y_i), $i = 1, 2, \ldots, n$ through iterative branch skeleton.
(2) Use the formula below to determine the straight-line separation between two places:

$$d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, i = 1, 2, \ldots, n \tag{7}$$

(3) Include the line of sight distances of each part:

$$sun = \sum_{i=1}^{n-1} d_i \tag{8}$$

(4) Keep repeating the above steps until the distance between the two points is finally calculated.

3.3. Calculation of Maximum Crack Pixel Width

Find a point in the crack skeleton and use the tangent direction of that point as the direction of the pixel to draw the normal one. Locate a point on the central axis where it intersects with the normal, and then determine the distance between that point and the edge point. Twice this distance is the pixel width of the crack. Calculate the width of each crack in the skeleton diagram and compare it with the maximum value. The crack width at its widest is the outcome. The calculation method for the maximum width of the crack is shown in Figure 12:

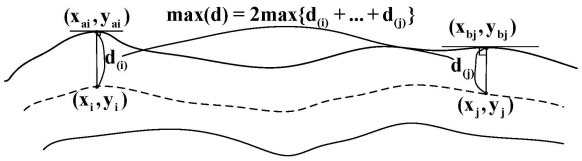


Figure 12. Maximum crack width calculation.

The entire process of calculating the width of the crack is as follows:

(1) Obtain the n sets of target point coordinates between the starting point and the end-point $(x_{ai}, y_{bi}), i = 1, 2, \ldots, n; a, b = 1, 2, \ldots, n$ through an iterative branch skeleton.

(2) The orientation of the skeleton can be obtained from the coordinates of each point on the skeleton; that is, its normality is discernible. The appropriate points on the central axis can be identified by using the procedure described above to obtain their coordinates. The target point is $(x_i, y_i), i = 1, 2, \ldots, n$.

(3) The crack's breadth is now equal to double the separation between the two points:

$$d_i = \sqrt{(x_{ai} - x_i)^2 + (y_{ai} - y_i)^2}, i = 1, 2, \ldots, n; a, b = 1, 2, \ldots, n \tag{9}$$

(4) The largest crack at each point is compared:

$$\max(d) = 2\max\left\{d_{(i)} + \ldots + d_{(j)}\right\} \tag{10}$$

(5) Repeat the procedure a few more times to determine the last point's crack's breadth.

3.4. Calculation of Crack Physical Size

Using the formula to connect the object size in the image with the real size so as to achieve accurate measurement, the specific calculation method is as follows:

$$P = (D \times \mathrm{O})/(f \times S) \tag{11}$$

where P represents the physical size, D represents the shooting distance, O represents the pixel size of the crack, f represents the focal length, and S represents the number of pixels per centimeter of the photosensitive element. The pixel size of the crack can be calculated by pixel calibration; the shooting distance is the distance between the camera and the object, which can be obtained by methods such as triangulation; and the focal length is an internal parameter of the camera and can be obtained from the technical parameters of the camera. The number of pixels per centimeter corresponding to the photosensitive element is the size of the camera's image sensor, which can also be obtained from the technical parameters of the camera. Through this formula, it is possible to calculate the physical size corresponding to any pixel in the image, thereby achieving an accurate measurement of object size and distance in the image.

4. Experimental Results and Analysis

4.1. Experimental Environment

The experimental environment of this article is shown in Table 1. The parameter settings are shown in Table 2.

Table 1. Experimental environment configuration.

Experimental Environment	Experimental Configuration
Operating system	Ubuntu 20.04.4
CPU	Intel core i5-10400
GPU	GTX 3060 12 GB
RAM	16 GB
Experimental tools	Pycharm + python 3.8.12
Deep-learning framework	Pytorch + detectron 2

Table 2. Parameter settings.

Parameter Name	Parameter Value
NUM_CLASSES	1
Weight_decay	0.05
Learning_rate	0.00005
Iter	270,000

4.2. Experimental Dataset Collection

To facilitate the conversion of crack sizes in the image to actual crack sizes, the ratio of crack length in the image to actual length is set to 1:4, the camera lens is kept perpendicular to the road surface, and the distance between the camera and the road surface remains unchanged when acquiring images. The image acquisition device is shown in Figure 13. The camera is mounted on a bracket, and the operator only needs to control the switch to use the continuous acquisition mode during acquisition. In order to obtain crack images under different lighting conditions, this paper collected road crack images from 6 a.m. to 9 a.m., 10 a.m. to 12 p.m., and 5 p.m. to 7 p.m. on a sunny day, and from 7 a.m. to 9 a.m., 10 a.m. to 12 p.m., and 3 p.m. to 5 p.m. on a cloudy day. The collected road cracks include asphalt road surfaces and concrete road surfaces, and the cracks on asphalt and concrete road surfaces are somewhat different. The cracks on asphalt road surfaces are mainly fatigue cracks and alligator cracks, which are linear or reticulated in distribution. The cracks on concrete road surfaces can be shrinkage cracks, cold joints, thermal cracks, structural cracks, etc., with different shapes and distribution methods. The original crack images were divided into 200 $\times$ 200 images. A total of 1500 segmented images with cracks and 300 segmented images without cracks were selected as training samples. To ensure the completeness of the samples, all lengths of cracks should be included in the training samples. To evaluate a training iteration, 300 images were randomly selected from the training samples as test samples. It is worth noting that in order to ensure the independence between training samples and test samples, these 300 images did not participate in this iteration.

4.3. Set Evaluation Indicators

The algorithm model in this paper is evaluated by four indicators: accuracy, precision, recall, and intersection over union (IoU). Table 3 presents the particular calculating formulas. Accuracy can be used to determine how accurate a model is, that is, the ratio of the number of pixels correctly identified by the model to the total number of pixels. Precision represents the proportion of samples that are truly positive among those that the model selected as positive. IoU intersection over union is the ratio of the intersection and union of true values and predicted values. TP stands for the proportion of positive pixels that were correctly identified, TN for positive pixels that were correctly identified, FP for negative pixels that

were mistakenly identified as positive, and FN for positive pixels that were mistakenly identified as negative.

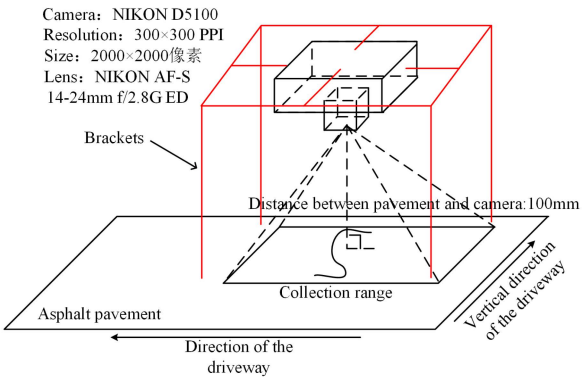


Figure 13. Data acquisition device.

Table 3. Calculation formulas for evaluation indicators.

Evaluation Indicators	Calculation Formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP + FP}$
Recall	$\frac{TP}{TP + FN}$
IoU	$\frac{TP}{TP + FP + FN}$

4.4. Instance Segmentation Comparison Experiment

In order to verify the feasibility of the improvement of the SparseInst model by the algorithm in this paper, a comparison experiment was conducted. By comparing the algorithm in this paper with the current mainstream segmentation models, as shown in Figure 14, it was found that, under the condition of sufficient light and clear crack morphology, several other model methods can extract the crack morphology. However, the algorithm in this paper can extract the cracks clearly and completely, while the cracks extracted by other methods are all somewhat blurred or broken. The above results show that under ideal conditions, the accuracy of crack extraction by the algorithm in this paper is better than that of the original SparseInst network and other segmentation network models compared. This also shows that the improvement of the SparseInst model by the algorithm in this paper is feasible and can improve the accuracy and robustness of crack recognition.

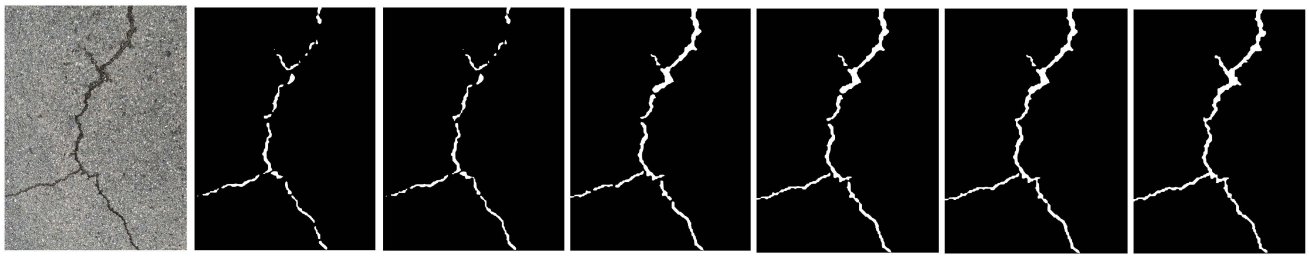


Figure 14. Recognition effect of six methods in ideal environment.

In order to confirm the algorithm's viability even more, this paper conducted experiments under dark–light conditions, as shown in Figure 15, and under complex crack conditions, as shown in Figure 16.

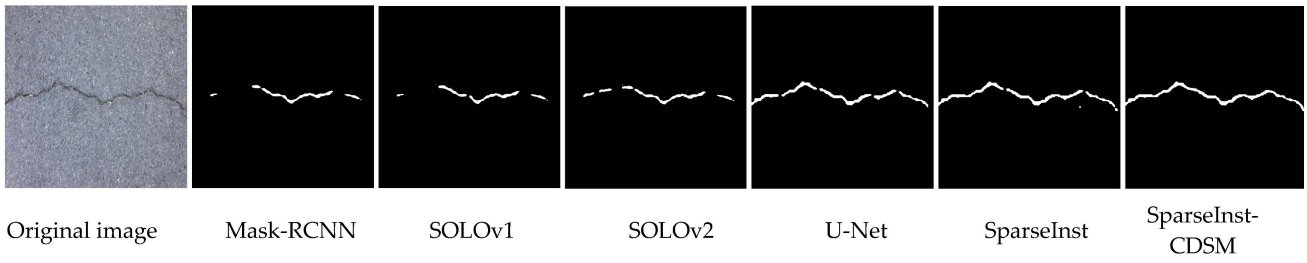


Figure 15. Recognition effect of six methods in dark–light environment.

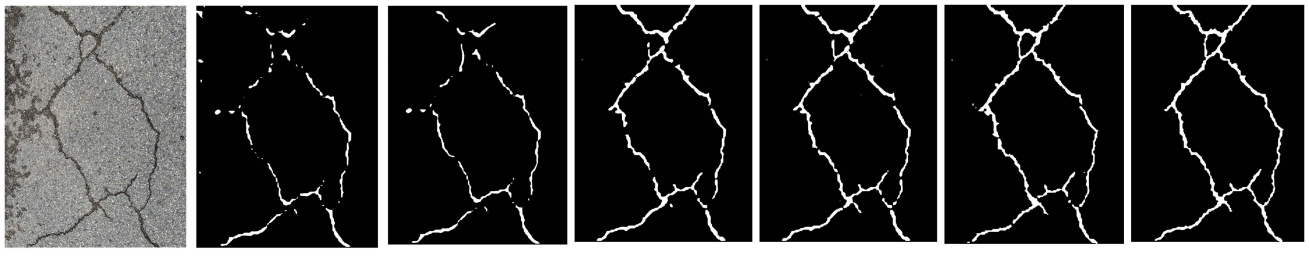


Figure 16. Effect of six methods in recognizing complex cracks.

Through the above experiments, it can be concluded that under conditions of insufficient light or complex crack morphology, other segmentation models often fail to extract cracks completely, while the algorithm in this paper can extract crack segmentation results of a certain quality under these conditions. This is because the algorithm in this paper can better use the texture information in the image to improve the recognition ability of cracks. Therefore, compared with the original SparseInst network model, the algorithm in this paper performs better in dealing with complex situations.

The specific experimental comparison results are shown in Table 4. Under the same dataset, the algorithm in this paper is compared with eight mainstream segmentation network models. According to the evaluation indicators defined in the previous section, the values of each indicator are calculated. The accuracy of the algorithm in this paper is 94.58%, the precision is 82.77%, the recall is 83.26%, and the IoU is 87.68%, which are higher than both the original SparseInst network and other network models compared, fully reflecting the superiority of the algorithm in this paper.

Table 4. Crack detection accuracy.

Method	Accuracy	Precision	Recall	IoU
Mask-RCNN	83.21%	65.86%	62.75%	81.69%
DeepLab	81.33%	61.67%	71.45%	79.34%
SegNet	84.47%	65.24%	72.34%	82.57%
PSNet	83.25%	70.12%	73.33%	80.21%
SOLOv1	85.35%	71.42%	76.76%	83.68%
SOLOv2	86.21%	73.39%	77.28%	82.57%
U-Net	88.31%	80.77%	81.67%	84.33%
SparseInst	89.45%	74.76%	80.39%	82.97%
SparseInst-CDSM	94.58%	82.77%	83.26%	87.68%

Figure 17 shows the comparison of the precision–recall curve test results of various segmentation algorithms under the same dataset. The results show that under the same recall rate, the precision of the algorithm proposed in this paper is higher. At the same time, under the same precision conditions, the recall rate of the algorithm in this paper is higher. Therefore, it can be concluded that the algorithm in this paper is better than other algorithms in terms of crack segmentation effects.

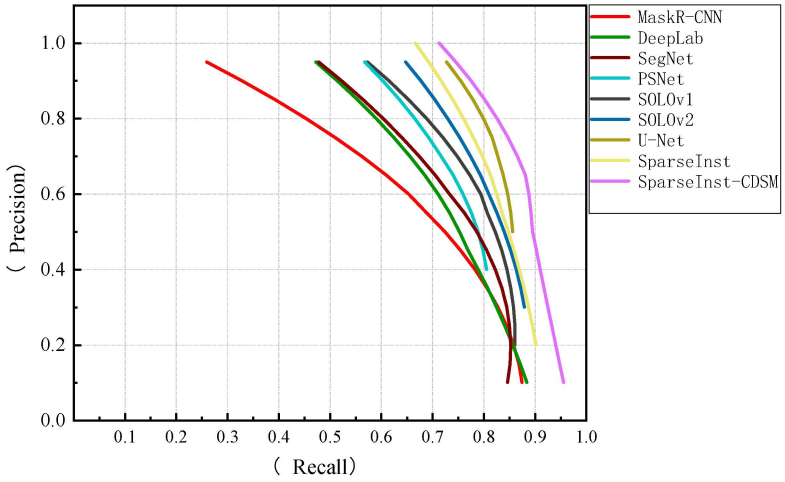


Figure 17. Precision–recall curve test chart.

The algorithm suggested in this research needs to be further tested to confirm its efficacy and a more thorough evaluation of the model's performance, the public dataset CRACK500, and a self-built dataset are used to compare with some mainstream instance segmentation networks on the evaluation index AP (Average Precision). AP considers the precision (Precision) at different recall (Recall) levels, thus more comprehensively evaluating the performance of the model. Different IoU thresholds are selected, including 50%, 75%, and the average precision AP, under all IoU thresholds as reference standards. The experimental results are shown in Table 5.

Table 5. Instance partition network comparison.

Datasets	Methods	AP%	AP50%	AP75%	FPS
CRACK500	Mask-RCNN	61.63	87.45	77.31	27.7
CRACK500	DeepLab	63.77	86.33	76.53	21.2
CRACK500	SegNet	62.35	84.34	80.32	28.8
CRACK500	PSNet	63.65	85.76	81.33	23.3
CRACK500	SOLOv1	65.37	87.75	79.67	32.4
CRACK500	SOLOv2	62.78	88.21	80.52	37.9
CRACK500	U-Net	66.72	89.32	82.45	29.7
CRACK500	SparseInst	65.47	90.77	81.77	52.5
CRACK500	SparseInst-CDSM	69.89	92.86	84.62	56.2
Self-built datasets	Mask-RCNN	62.57	88.73	79.73	29.8
Self-built datasets	DeepLab	61.76	86.82	78.46	22.4
Self-built datasets	SegNet	64.37	83.67	81.42	27.3
Self-built datasets	PSNet	63.79	88.92	81.97	25.7
Self-built datasets	SOLOv1	66.67	87.81	82.33	35.5
Self-built datasets	SOLOv2	67.78	89.44	83.52	41.6
Self-built datasets	U-Net	68.74	90.47	82.45	33.1
Self-built datasets	SparseInst	67.68	93.73	83.26	57.6
Self-built datasets	SparseInst-CDSM	71.72	95.86	85.57	61.9

After being verified in the CRACK500 dataset and self-built dataset, the algorithm in this paper has shown excellent performance in object detection tasks. In the CRACK500 dataset, the AP value of the algorithm in this paper reached 69.89%, the AP50 value was 92.86%, and the AP75 value was 84.62%; in the self-built dataset, the AP value of the algorithm in this paper was 71.72%, the AP50 was 95.86%, and the AP75 was 85.57%. Compared with the comparative segmentation model, the algorithm in this paper showed higher AP, AP50, and AP75 indicators in both datasets, proving that it has higher accuracy and can effectively detect crack defects in images. Since the goal of SparseInst-CDSM is real-time instance segmentation, this paper mainly compares the differences between

SparseInst-CDSM and the latest real-time instance segmentation methods in terms of accuracy and inference speed. Table 5 shows that SparseInst-CDSM achieves 56.2 FPS on the CRACK500 dataset and 61.9 FPS on the self-built dataset, outperforming most real-time segmentation models with better performance and inference speed.

4.5. Ablation Experiment

Through ablation experiments, the improved modules of the model can be clearly verified to prove the role of each improved module. Here, we chose the Resnet101 backbone network with the best performance in the above comparison experiment as the backbone. On this basis, the effectiveness of adding the attention mechanism, DCN convolution, and introducing the SPM strip pooling module and the MPM mixed pooling module was verified, as shown in Table 6.

Table 6. Ablation experiment results.

Model	CBAM	DCN v2	SPM	MPM	AP%	AP_{50}%	AP_{75}%
SparseInst101	×	×	×	×	67.72	93.81	78.35
Optimization model 1	√	×	×	×	67.23	94.67	79.53
Optimization model 2	×	√	×	×	68.44	92.98	79.13
Optimization model 3	√	√	×	×	68.71	93.73	78.85
Optimization model 4	×	×	√	×	67.16	92.46	78.56
Optimization model 5	√	√	√	×	69.44	94.33	80.11
Optimization model 6	×	×	×	√	68.97	93.22	79.12
SparseInst-CDSM	√	√	√	√	71.21	95.86	85.57

Through ablation experiments, the model's performance in identifying targets was shown to be significantly improved by the addition of attention processes. The attention mechanism allows the model to focus more accurately on areas of interest, thereby improving its accuracy and robustness. At the same time, the effectiveness of DCN convolution in segmentation tasks was verified. This is because DCN convolution has stronger feature expression and better object edge recognition abilities. The SPM strip pooling module can decompose the feature map into strips of different scales and perform pooling operations for each strip, thereby improving the multi-scale expression ability of the feature map. The MPM mixed pooling module combines the advantages of different pooling methods to improve the diversity and robustness of the feature map. These improved modules can better solve various challenges in segmentation tasks, giving the model a wider application prospect in practical applications.

4.6. Comparison of Calculated and Measured Values of Cracks

The width of the cracks is measured manually using a crack width tester and a crack detector. The length of the cracks is calculated by scanning the cracks with a laser scanner and then using software to calculate the length of the cracks. Figure 18 shows the comparison results.

Two hundred sets of calculated and measured values were randomly selected. The horizontal axis is defined as the measured value, and the vertical axis is defined as the calculated value. When the point falls on $y = x$, it proves that the calculated value and the measured value are the same, and the effect is the best. As can be seen from the figure, most of the points are basically concentrated near the $y = x$-axis. After calculation, the average error does not exceed 10%, and the maximum error does not exceed 20%. However, it can also be seen from the figure that some points have large errors and are far away from the $y = x$-axis. One possible reason is that there is a certain error in the manual measurement process. Due to the different shapes and sizes of cracks and their different positions, manual measurement may have some subjectivity and errors. In addition, during image processing, some crack information may be missed or improperly handled, which can also cause errors.

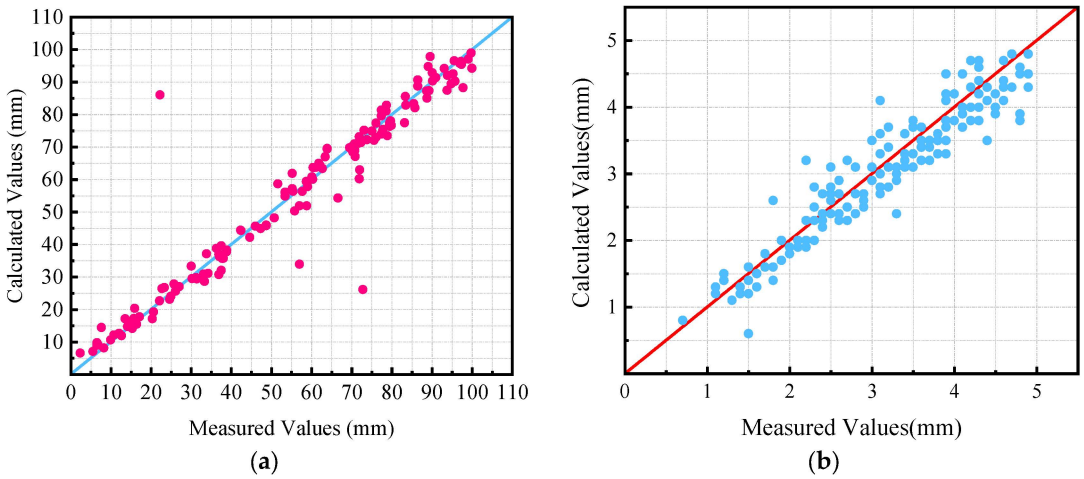


Figure 18. Scatter plot of measured and calculated values of crack length and width (**a**) Crack length (**b**) Maximum crack width.

5. Conclusions

For the detection of road cracks, this paper proposes the SparseInst-CDSM algorithm with the addition of the CBAM attention mechanism, the introduction of the SPM stripe pooling structure and the MPM hybrid pooling structure, as well as the addition of DCNv2 convolution to design and implement an algorithm for measuring road crack-related features. Experimental results show that under the same dataset, the evaluation indicators of SparseInst-CDSM are 94.58% accuracy, 82.77% precision, 83.26% recall rate, and 87.68% intersection over union, all higher than other segmentation networks. Compared with the original SparseInst evaluation indicators, they have increased by 5.13%, 8.01%, 2.87%, and 4.71%, respectively, and the feature distinction is obvious. In addition, by comparing the calculated crack length and width values with the actual measured values, the error is basically kept within 10%. In summary, the road crack area segmentation algorithm based on the SparseInst network proposed in this paper provides an effective solution to the problem of road crack detection. At the same time, this algorithm also has a wide range of application prospects and can provide references and inspiration for feature segmentation problems in other fields. Further research and optimization of this algorithm are expected to bring more extensive and in-depth applications to the field of road maintenance and repair.

Author Contributions: S.-J.W.—edited and wrote the manuscript and collected and analyzed the data; J.-K.Z.—validation and formal analysis and supervision of project administration and funding acquisition; X.-Q.L.—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China project, grant number 51904161.

Data Availability Statement: Data are contained within the article. The data presented in this study can be requested from the authors.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gavilán, M.; Balcones, D.; Marcos, O.; Llorca, D.F.; Sotelo, M.A.; Parra, I.; Ocaña, M.; Aliseda, P.; Yarza, P.; Amírola, A. Adaptive Road Crack Detection System by Pavement Classification. *Sensors* **2011**, *11*, 9628–9657. [CrossRef] [PubMed]
2. Sun, X.; Huang, J.; Liu, W. Decision model in the laser scanning system for pavement crack detection. *Opt. Eng.* **2011**, *50*, 127207. [CrossRef]
3. Yao, M.; Zhao, Z.; Yao, X.; Xu, B. Fusing complementary images for pavement cracking measurements. *Meas. Sci. Technol.* **2015**, *26*, 025005. [CrossRef]
4. Hu, G.X.; Hu, B.L.; Yang, Z.; Huang, L.; Li, P. Pavement Crack Detection Method Based on Deep Learning Models. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 1–13. [CrossRef]

5. Abdellatif, M.; Peel, H.; Cohn, A.G.; Fuentes, R. Pavement Crack Detection from Hyperspectral Images Using A Novel Asphalt Crack Index. *Remote Sens.* **2020**, *12*, 3084. [CrossRef]
6. Ren, J.; Zhao, G.; Ma, Y.; Zhao, D.; Liu, T.; Yan, J. Automatic Pavement Crack Detection Fusing Attention Mechanism. *Electronics* **2022**, *11*, 3622. [CrossRef]
7. Wang, W.; Wu, L. Pavement crack extraction based on fractional integral valley bottom boundary detection. *J. South China Univ. Technol. (Nat. Sci. Ed.)* **2014**, *42*, 117–122.
8. Liang, R.; Zhigang, X.; Xiangmo, Z.; Jingmei, Z. Pavement crack connection algorithm based on prim minimum spanning tree. *Comput. Eng.* **2015**, *41*, 31–36.
9. Kirschke, K.R.; Velinsky, S.A. Histogram-based approach for automated pavement-crack sensing. *J. Transp. Eng.* **1992**, *118*, 700–710. [CrossRef]
10. Oh, H.; Garrick, N.W.; Achenie, L.E. Segmentation algorithm using iterative clipping for processing noisy pavement images. In *Imaging Technologies: Techniques and Applications in Civil Engineering. Second International Conference Engineering Foundation and Imaging Technologies Committee of the Technical Council on Computer Practices*; American Society of Civil Engineers: Reston, VA, USA, 1998.
11. Fang, C.; Zhe, L.; Li, Y. Images crack detection technology based on improved K-means algorithm. *J. Multimed.* **2014**, *9*, 822.
12. Mathavan, S.; Vaheesan, K.; Kumar, A.; Chandrakumar, C.; Kamal, K.; Rahman, M.; Stonecliffe-Jones, M. Detection of pavement cracks using tiled fuzzy Hough transform. *J. Electron. Imaging* **2017**, *26*, 053008. [CrossRef]
13. Amhaz, R.; Chambon, S.; Idier, J.; Baltazart, V. Automatic Crack Detection on Two-Dimensional Pavement Images: An Algorithm Based on Minimal Path Selection. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 2718–2729. [CrossRef]
14. Zhang, A.; Li, Q.; Wang, K.C.P.; Qiu, S. Matched Filtering Algorithm for Pavement Cracking Detection. *Transp. Res. Rec. J. Transp. Res. Board* **2013**, *2367*, 30–42. [CrossRef]
15. Hongxun, S.; Weixing, W.; Fengping, W.; Linchun, W.; Zhiwei, W. Pavement crack detection by ridge detection on fractional calculus and dual-thresholds. *Int. J. Multimed. Ubiquitous Eng.* **2015**, *10*, 19–30.
16. Oliveira, H.; Correia, P.L. Automatic road crack detection and characterization. *IEEE Trans. Intell. Transp. Syst.* **2012**, *14*, 155–168. [CrossRef]
17. Ma, D.; Fang, H.; Wang, N.; Xue, B.; Dong, J.; Wang, F. A real-time crack detection algorithm for pavement based on CNN with multiple feature layers. *Road Mater. Pavement Des.* **2022**, *23*, 2115–2131. [CrossRef]
18. Feng, X.; Xiao, L.; Li, W.; Pei, L.; Sun, Z.; Ma, Z.; Shen, H.; Ju, H. Pavement Crack Detection and Segmentation Method Based on Improved Deep Learning Fusion Model. *Math. Probl. Eng.* **2020**, *2020*, 8515213. [CrossRef]
19. Wu, Y.; Yang, W.; Pan, J.; Chen, P. Asphalt pavement crack detection based on multi-scale full convolutional network. *J. Intell. Fuzzy Syst.* **2021**, *40*, 1495–1508. [CrossRef]
20. Cha, Y.-J.; Choi, W.; Büyüköztürk, O. Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks. *Comput. Civ. Infrastruct. Eng.* **2017**, *32*, 361–378. [CrossRef]
21. Cha, Y.-J.; Choi, W.; Suh, G.; Mahmoudkhani, S.; Büyüköztürk, O. Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types. *Comput. Civ. Infrastruct. Eng.* **2018**, *33*, 731–747. [CrossRef]
22. Majidifard, H.; Jin, P.; Adu-Gyamfi, Y.; Buttlar, W.G. Pavement Image Datasets: A New Benchmark Dataset to Classify and Densify Pavement Distresses. *Transp. Res. Rec. J. Transp. Res. Board* **2020**, *2674*, 328–339. [CrossRef]
23. Maeda, H.; Sekimoto, Y.; Seto, T.; Kashiyama, T.; Omata, H. Road Damage Detection and Classification Using Deep Neural Networks with Smartphone Images. *Comput.-Aided Civ. Infrastruct. Eng.* **2018**, *33*, 1127–1141. [CrossRef]
24. Attard, L.; Debono, C.J.; Valentino, G.; Di Castro, M.; Masi, A.; Scibile, L. Automatic crack detection using mask R-CNN. In Proceedings of the 2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA), Dubrovnik, Croatia, 23–25 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 152–157.
25. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
26. Zhang, L.; Yang, F.; Zhang, Y.D.; Zhu, Y.J. Road crack detection using deep convolutional neural network. In Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 3708–3712.
27. Zhang, A.; Wang, K.C.P.; Li, B.; Yang, E.; Dai, X.; Peng, Y.; Fei, Y.; Liu, Y.; Li, J.Q.; Chen, C. Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network. *Comput.-Aided Civ. Infrastruct. Eng.* **2017**, *32*, 805–819. [CrossRef]
28. Dung, C.V.; Anh, L.D. Autonomous concrete crack detection using deep fully convolutional neural network. *Autom. Constr.* **2019**, *99*, 52–58. [CrossRef]
29. Yang, X.; Li, H.; Yu, Y.; Luo, X.; Huang, T.; Yang, X. Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network. *Comput.-Aided Civil Infrastruct. Eng.* **2018**, *33*, 1090–1109. [CrossRef]
30. Bang, S.; Park, S.; Kim, H.; Kim, H. Encoder–decoder network for pixel-level road crack detection in black-box images. *Comput.-Aided Civ. Infrastruct. Eng.* **2019**, *34*, 713–727. [CrossRef]
31. Ji, A.; Xue, X.; Wang, Y.; Luo, X.; Xue, W. An integrated approach to automatic pixel-level crack detection and quantification of asphalt pavement. *Autom. Constr.* **2020**, *114*, 103176. [CrossRef]

32. Chun, P.J.; Izumi, S.; Yamane, T. Automatic detection method of cracks from concrete surface imagery using two-step light gradient boosting machine. *Comput.-Aided Civ. Infrastruct. Eng.* **2020**, *36*, 61–72. [CrossRef]
33. Liu, J.; Yang, X.; Lau, S.; Wang, X.; Luo, S.; Lee, V.C.; Ding, L. Automated pavement crack detection and segmentation based on two-step convolutional neural network. *Comput.-Aided Civ. Infrastruct. Eng.* **2020**, *35*, 1291–1305. [CrossRef]
34. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; Part III 18. pp. 234–241.
35. Jang, K.; An, Y.K.; Kim, B.; Cho, S. Automated crack evaluation of a high-rise bridge pier using a ring-type climbing robot. *Comput.-Aided Civ. Infrastruct. Eng.* **2021**, *36*, 14–29. [CrossRef]
36. Jiang, S.; Zhang, J. Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system. *Comput. Civ. Infrastruct. Eng.* **2020**, *35*, 549–564. [CrossRef]
37. Wang, X.; Kong, T.; Shen, C.; Jiang, Y.; Li, L. Solo: Segmenting objects by locations. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; Part XVIII 16. pp. 649–665.
38. Wang, X.; Zhang, R.; Kong, T.; Li, L.; Shen, C. Solov2: Dynamic and fast instance segmentation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17721–17732.
39. Cheng, T.; Wang, X.; Chen, S.; Zhang, W.; Zhang, Q.; Huang, C.; Zhang, Z.; Liu, W. Sparse instance activation for real-time instance segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 4433–4442.
40. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
41. Wang, R.; Shivanna, R.; Cheng, D.; Jain, S.; Lin, D.; Hong, L.; Chi, E. DCN V2: Improved Deep Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In Proceedings of the Web Conference, Ljubljana, Slovenia, 19–23 April 2021; pp. 1785–1797. [CrossRef]
42. Hou, Q.; Zhang, L.; Cheng, M.M.; Feng, J. Strip pooling: Rethinking spatial pooling for scene parsing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 4003–4012.
43. Zhao, H.; Shi, J.; Qi, X.; Wang, X.; Jia, J. Pyramid scene parsing network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–27 July 2017; pp. 2881–2890.
44. Weng, X.; Huang, Y.; Wang, W. Segment-based pavement crack quantification. *Autom. Constr.* **2019**, *105*, 102819. [CrossRef]
45. Zhou, Q.; Ding, S.; Qing, G.; Hu, J. UAV vision detection method for crane surface cracks based on Faster R-CNN and image segmentation. *J. Civ. Struct. Health Monit.* **2022**, *12*, 845–855. [CrossRef]
46. Wieser, E.; Seidl, M.; Zeppelzauer, M. A study on skeletonization of complex petroglyph shapes. *Multimed. Tools Appl.* **2016**, *76*, 8285–8303. [CrossRef]
47. Lynn, N.D.; Sourav, A.I.; Santoso, A.J. Implementation of Real-Time Edge Detection Using Canny and Sobel Algorithms. In Proceedings of the IOP Conference Series: Materials Science and Engineering, Chemnitz, Germany, 24 March 2021; IOP Publishing: Tokyo, Japan, 2021; Volume 1096, p. 012079.
48. Ma, J.; Ren, X.; Tsviatkou, V.Y.; Kanapelka, V.K. A novel fully parallel skeletonization algorithm. *Pattern Anal. Appl.* **2022**, *25*, 169–188. [CrossRef]
49. Qiu, S.; Wang, W.; Wang, S.; Wang, K.C. Methodology for Accurate AASHTO PP67-10–Based Cracking Quantification Using 1-mm 3D Pavement Images. *J. Comput. Civ. Eng.* **2017**, *31*, 04016056. [CrossRef]