Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Deep Learning Based Automatic Crack Detection and Segmentation for Unmanned Aerial Vehicle Inspections

Kangcheng Liu[1], Xiaodong Han[2], and Ben M. Chen[1]

***Abstract*— Automatic crack detection and segmentation plays a significant role in the whole system of unmanned aerial vehicle inspections. In this paper, we have implemented a deep learning framework for crack detection based on classical network architectures including Alexnet, VGG and Resnet. Moreover, inspired by the feature pyramid network architecture, a hierarchical convolutional neural network (CNN) deep learning framework which is efficient in crack segmentation is also proposed, and the performance of it is compared with other state of the art network architecture. We have summarized the existing crack detection and segmentation datasets and established the largest existing benchmark-dataset on the internet for crack detection and segmentation, which is open-sourced for the research community. Our feature pyramid crack segmentation network is tested on the benchmark-dataset and gives satisfactory segmentation results. A framework for automatic unmanned aerial vehicle inspections is also proposed for the crack inspection tasks of various concrete structures.**

I. Introduction

Concrete structures play an important role in ground transportation networks. However, traditionally, concrete structure crack inspections are conducted manually by human operators with heavy and expensive mechanical equipment. It is logistically challenging, labor-intensive, costly, and dangerous, especially when inspecting the substructure and superstructure in harsh environments that are hard and dangerous to be accessed by human operators. Therefore, it is very meaningful and significant for us to develop a fully autonomous intelligent unmanned aerial system for inspecting large-scale concrete structures and detecting the defects such as cracks. Most importantly, a significant module for Unmanned Aerial Vehicle (UAV) intelligent inspection system is to develop computer vision algorithms for processing images captured and detecting cracks and structural damages.

Concrete structure crack inspection is a vehement research topic. In recent years, there are also many works done on inspection and exploration based inspection [1] [2] [3]. Traditional methods such as AdaBoost and SVM [4] [5] have been adopted to do crack detection and segmentation. While in the past few years, CNN-architectures such as Alexnet [6], VGG [7], Resnet [8] [9] and Googlenet [10] have shown great performance in object detection and classification, therefore they naturally have potentials to be adopted to do crack detection in various concrete inspection task [11] [12]. In this paper, we have implemented different kinds of networks and their accuracy and speed are compared.

This study proposes a framework about automatic crack detection and segmentation for UAV inspections. The contribution of this study is fourfold.

1) An overall unmanned aerial system architecture for UAV inspections is proposed.
2) We have established and tested neural network frameworks based on classical networks including Alexnet, VGG and Resnet for crack detection.
3) We have proposed and validate a hierarchical convolutional neural network (CNN) framework called *CrackNet* which is efficient in crack segmentation on our self-established dataset. We also compare our network with other state of the art segmentation network structures.
4) The benchmark dataset is merged and open-sourced for the research community.[1] To our knowledge, this is the biggest dataset on the internet which can be used for both crack detection and segmentation so far.

The rest of the paper is organized as follows: Section II introduces the overall system architecture of our unmanned aerial systems. The deep learning architectures for crack detection are described in Section III. Section IV elaborately illustrates the feature pyramid network architectures for crack segmentation. Section V discusses the benchmark dataset setup and the crack segmentation experimental results. Section VI concludes the paper and proposes the future work.

II. The System Architecture

A. The Avionic System

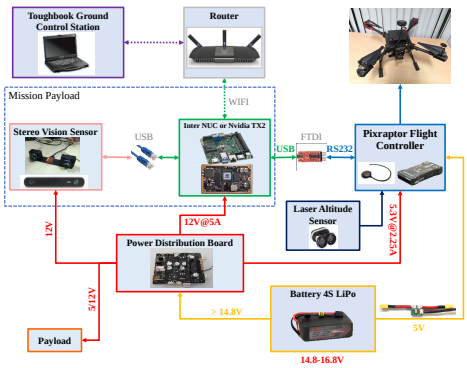

Fig. 1. The avionic system

[1]https://github.com/KangchengLiu/Crack-Detection-and-Segmentation-Dataset-for-UAV-Inspection

[1]K. Liu and B. M. Chen are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong. (email: kcliu@mae.cuhk.edu.hk, bmchen@cuhk.edu.hk)

[2]X. Han is with the College of Computer and Control Engineering, Minjiang University, Fuzhou, China. (email: hxdgod@mju.edu.cn)

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

The overall hardware architecture of our quadrotor system is shown in Figure 1. The 450 sized UAV frame has been used for our UAV platform. A typical control method in our group is a cascade control loop tackling with inner loop (the attitude loop) and the outer loop (the position loop) respectively [13]. Among those control approaches the composite nonlinear feedback (CNF) [14] method for attitude stabilization and robust perfect tracking (RPT) [15] method for trajectory tracking shows the best performance. We have used PX4 based flight controller to implement the flight control law. And we utilized Inter NUC (or Nvidia Jetson TX2 as an alternative) as the on-board mission computer for vision processing, target detection, etc. Communications of UAVs with other UAVs and GCS (Ground Control Station) are done using WIFI.

B. The Overall System for UAV Inspections

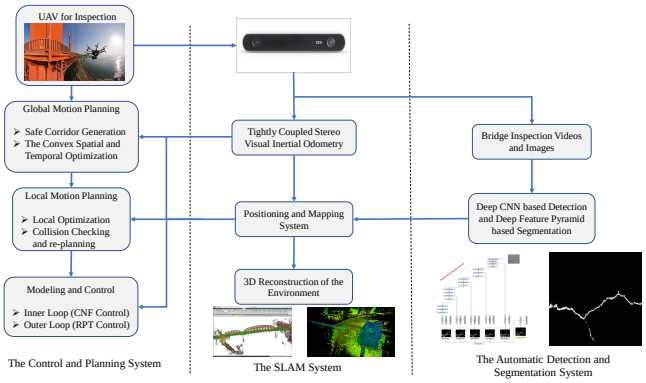

Fig. 2. The overall system structure

As shown in Figure 2, the UAV inspection system consists of 3 subsystems:

1) The Control and Planning System
2) The Simultaneous Localization and Mapping (SLAM) System
3) The Automatic Detection and Segmentation System

The subsystems should be integrated to make an autonomous unmanned system, and the system integration of all the technical components is the most challenging task.

III. Deep Learning Based Crack Detection

Considering the CNN architecture as a classifier, the crack detection problem can be formulated as a binary classification problem to classify between the crack and non-crack image samples.

A. Dataset Preparation

A representative and typical dataset is very important for the crack detection task. In the previous research [16] [17] [18] [19] [20], every method is validated on their self-established small-scale dataset to evaluate its performance, which makes the comparisons between different methods difficult. We have summarized different crack datasets and constructed a benchmark dataset for crack detection and segmentation. To our knowledge, this is the biggest dataset which can be utilized for both crack detection and segmentation and it will be beneficial for further researches.

B. Data Preprocessing and Labeling

The pre-processing of training images is done in order to make full use of the information in the image. The original big images are cropped into the 100×100 sub-image samples and the sub-images are labelled as the crack or non-crack. We have used Photoshop and *labelme* [21] to obtain the pixel level label of the image. If the percentage of the crack pixel in the cropped image is greater than 9%, we define the image as the crack image.

C. The Network Structure and Training

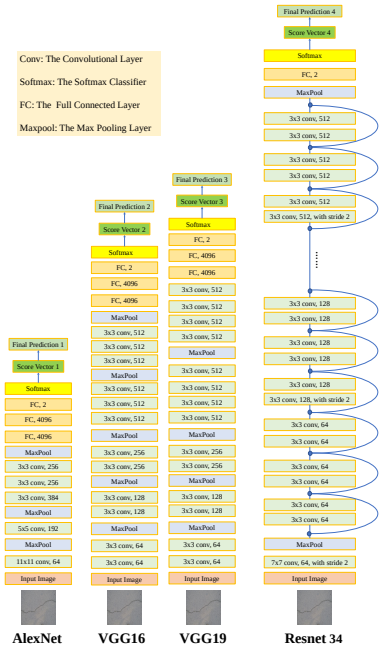

Fig. 3. The different network architectures to implement crack detection

This paper used different architectures (VGG-net, Alexnet, Resnet) to do image classification and compare the results. The network architectures are shown in Figure 3. In order to accelerate the training process and improve the accuracy on our relatively small dataset (compared with *ImageNet*), we utilize the transfer learning approach and use the pre-trained weight on ImageNet provided by *Pytorch* to initialize the weights of different network architectures.

The training runs 500 iterations with a batch size 32 over 15,000 images. The data is divided into 3 parts, 70% for training, 15% for testing and 15% for validation. We set the learning rate as 1e-3 for the first 250 iterations and 3e-4 for the next 250 iterations. The cross-entropy loss was used for the network training. And the loss function can be formulated as follow:

$$L = \sum_{x^{In}} y(x^{In}) \cdot log(p(x^{In})) + (1 - y(x^{In})) \cdot log(1 - p(x^{In}))] \quad (1)$$

where $p(x^{In})$ represents the predicted possibility of a input image to be a crack image. While y is the label of the input image. For crack image, y=1. For non-crack image, y=0.

IV. Feature Pyramid Network Based Crack Segmentation

The conventional CNNs architectures are more effective for image classification or recognition tasks, but not suitable for more complex image understanding tasks, such as semantic segmentation. The crack segmentation problem can be formulated as a pixel level classification problem and the '0' and '1' refer to "non-crack" and "crack" respectively. Inspired by the hierarchical structure of Full Convolutional Network (FCN) [22] and Segnet [23], we have proposed a feature pyramid based CNN network structure for pixel-wise semantic segmentation.

A. Motivation

The main idea lies in that the high-level and low-level feature maps can be fused to obtain the final prediction feature map. Note that when doing upsampling on a convolutional feature map, the abstraction levels of the different convolution layers are different. The features learned by the shallow layer are local features. As the convolutional network becomes deeper, the receptive field continues to increase, the feature learned becomes more abstract.

As shown in Table II and Figure 4, the optimal prediction occurs in the 2nd and 3rd convolutional feature map. It demonstrates that the shallow feature maps give satisfactory prediction of crack profile but is very sensitive to local noise in image, and are often not robust when directly used for prediction. While the abstract features learned from deeper network layers are very robust to noise in image, but they are prone to suffer from gradient disappearance or boundary blurring, which makes the feature map too fuzzy to retaining the prediction boundaries for the crack. Therefore, in order to get accurate boundary segmentation, the tradeoff method is to upsample the feature maps of different levels to the same size and then fuse the outputs of different layers of the ConvNet to get a better final prediction feature map.

B. Network Architecture

As shown in the Figure 4, the first 13 convolutional layers are inspired by the VGG-16 network and they can be divided into 5 convolutional blocks. Batch normalization is used to speed up the convergence of the network training. Every convolutional layer is consist of convolution, batch normalization and Rectified Linear Unit (ReLU). The fully connected layers are abandoned because pixel level prediction is required. And the last pooling layer is also not used because the output of it is too blurry to generate a satisfactory prediction feature map. The max-pooling of 2×2 filter is adopted after each convolutional blocks of the VGG-16 backbone. It makes the amount of parameters in the network smaller while not causing the translational variance over the small spatial shift. Then we do 1×1 convolution with one kernel to the output feature maps of the 5 convolutional blocks to make them become 1-channel. Next the *bilinear Interpolation* is utilized to do deconvolution to the prediction feature map to make it become the same size as the input image. Finally, the linear fusion method can be used to concatenate outputs of different convolutional blocks. And 1×1 convolution with 1 output depth is applied to the 5-channel concatenation result to make it become 1-channel fused prediction feature map. At last, the softmax classifier is used to do logistic regression for both side output prediction feature map and fused prediction feature map to get classification at pixel level. And the outputs of softmax classifier are 1-channel possibility maps indicating the possibility of each pixel to be a crack pixel. Then a fixed threshold can be assigned to get the prediction label based on the outputs of the softmax layer.

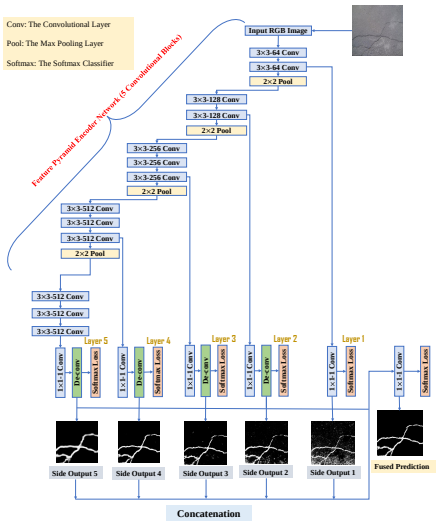

Fig. 4. The overall segmentation network structure

Finally, the guided filtering (GF) [24] post-processing is also utilized to do refinement to the prediction results.

C. Loss Function Formulation and Optimization

One of the major problems in crack segmentation is that more than 90% of the pixels are non-crack, which will easily cause the training problems because of the imbalanced model updating [25]. The original cross-entropy loss function can not be adopted because even if all the pixels are classified as non-crack, the prediction accuracy is still acceptable and correct prediction is given to most of the pixels. However, it makes no sense to our crack segmentation task. Therefore, rather than simply use the cross-entropy loss function, we adopted the class-balanced loss function to weight between the crack and non-crack pixel differently, which is inspired by [23] [25]. The details will be discussed as follows:

Denote the training set which includes N images as $V = (S^n, T^n), n = 1, 2, ..., N$, in which the set $S^n = \{s_j^{(n)}, j = 1, 2, ..., J\}$ represents the original input image, the set $T^n = \{t_j^{(n)}, j = 1, 2, ..., J, t_j^{(n)} \in \{0, 1\}\}$ represents the ground truth crack label feature map which is corresponding to S^n. J denotes the total amount of pixels in each image.

For each labeled pixel, the target is to train our network to learn a model that produces final prediction which is consistent with the ground truth. Denote H as the number of convolutional blocks ($H = 5$ in our case), the feature map generated by the side output of each convolutional block can be written as $F^{(h)} = F_j^{(h)}, j = 1, .., J$, where $h = 1, 2, .., H$. And the fused feature map can be written as F^{fused}=$F_j^{fused}, j = 1, 2, .., J$. Then the parameters of the convolutional layers of VGG net can be defined as W_1, and the parameters of the sided output convolutional layers and final fuse convolutional layer can be defined as $W_2 = \{w_1, w_2, ..., w_H, w_{fuse}\}$, in which H is the number of the side output, which is equal to the number of convolutional blocks in our case. The total parameter set of W_1, W_2 can be denoted as W. For the crack segmentation task, we should attach less significance to the loss function of non-crack pixels which is in the majority, while attach more significance to the loss function crack pixels which are in the minority. We adopted median frequency balancing in the weight determination. If the total number of crack and non-crack pixels in the training set are p and q respectively. The class frequencies of crack and non-crack are $\frac{p}{p+q}$ and $\frac{q}{p+q}$, while the median for the 2 classes is 0.5. Then the median divided by the class frequency gives the weight of 2 classes. In our case, the weights of the loss function for the crack pixels and non-crack pixels are $\alpha_1 = \frac{p+q}{2p}$ and $\alpha_2 = \frac{p+q}{2q}$ respectively. Then for each side-output layer, the improved loss function can be formulated as:

$$\begin{aligned} L_{side}^h(F_j; W_1; W_2) = -\alpha_2 \sum_{j \in S^-} log(1 - P(F_j; W_1; W_2)) \\ -\alpha_1 \sum_{j \in S^+} log(P(F_j; W_1; W_2)) \end{aligned} \tag{2}$$

where $h = 1, 2..., H$ respectively, S^+ and S^- are the total number of crack pixels and non-crack pixels respectively for an input image. And *P* denotes the predicted possibility of each pixel to be a crack one. Next the improved loss function for the fused output can be also written as:

$$\begin{aligned} L_{fuse}(F_j^{fused}; W_1; W_2) = -\alpha_2 \sum_{j \in S_-} log(1 - P(F_j; W_1; W_2)) \\ -\alpha_1 \sum_{j \in S_+} log(P(F_j; W_1; W_2)) \end{aligned} \tag{3}$$

And then the total loss function is written as:

$$\begin{aligned} L_{total}(W_1; W_2) = \sum_{j=1}^{J} (\sum_{h=1}^{H} \lambda_h L_{side}^h(F_j; W_1; W_2) \\ + L_{fuse}(F_j^{fused}; W_1; W_2)) \end{aligned} \tag{4}$$

The parameter λ_h can be tuned to assign different weights to the loss at different levels of the feature map. And the weights are finally set as $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$ = $\{0.5, 1.0, 0.8, 0.5, 0.3\}$, which gives the best segmentation results. The detailed experiment results and comparisons are shown in Section V. Finally, the computed total loss function can be utilzed for back propagation in the optimization procedure which is similar to Stochastic Gradient Descent (SGD). To improve the convergence speed, Adam algorithm was adopted to achieve SGD with momentum which accumulates the past gradient to accelerate the gradient downhill.

D. Model Parameters and Data Argumentation

The network proposed above is implemented on the open-source deep learning framework *Pytorch*. The parameters of the whole convolutional network are initialized with normal distribution. And the *Bilinear Interpolation* is utilized to do upsampling in the deconvolution process. The learning rate is set to 9e-5 and the batch size is set to 2. The momentum and weight decay are set to 0.8 and 6e-4 respectively in the gradient descent optimization. The network is trained with 2.4e5 iterations in total and the learning rate will be decrease by 20% after every 3e4 iterations. All the training are done on a Nvidia GeForce GTX RTX2080Ti GPU.

Data argumentation also plays a significant role in the training of the networks to enhance the utilization of the training dataset. The images are rotated to 6 different angles from 0 degree to 360 degree with a 60 degree interval. And we also flipped the images horizontally therefore the whole dataset is enhanced 12 times. In this way, we can significantly increase the diversity of data available for training models, without actually collecting new data.

V. Experimental Results

A. Comparison of Results of crack detection

TABLE I
THE COMPARISON OF DETECTION RESULTS BETWEEN METHODS

Network Architecture	Accuracy/%	Validation Time/ms
Alexnet [6]	94.6	139.8
VGG 16 [7]	95.5	135.3
VGG 19 [7]	95.7	137.4
Resnet [8]	96.9	123.7

Crack detection test was done on the device with Intel Xeon E5-2620 v4 CPU and Nvidia RTX 2080Ti GPU. As shown in Table I, the Resnet has the best accuracy and interference speed among different kinds of networks. Although the CNN architectures can not give perfect prediction, the result for crack detection is important because it can be utilized for coarse localization of the defected region. And the definition for crack sub-images based on the percentage of the crack pixel is not always accurate. More accurate prediction will require human to label the crack or non-crack sub-image ground truth which is time-consuming and labour-consuming. Therefore segmentation of crack at pixel level is necessary for a more complete crack inspection system.

B. The Benchmark Dataset Setup

In our project, efforts have been made to merged more than 10 different crack segmentation datasets available on the Internet. And we also collected some crack images from taking photos of concrete structures and labelled them

utilizing a effective graphical image annotation tool: *labelme* [21]. The improvement can be made based on the benchmark dataset and there is no doubt that this will be beneficial and crucial to the whole research community of crack detection and segmentation. In addition, more data will be added and merged into the dataset with the research project going on. Following are some of the datasets which we collected images from:

1) GAPs Dataset: The German Asphalt Pavement Distress (*GAPs*) dataset [17] provides a standardized large-scale and high-quality dataset. A total of 1969 gray-scale images are included in the GAPs dataset.

2) CRACK500 Dataset: The *CRACK500* [16] is a dataset containing more than 500 $2,000 \times 1,500$ images by using phones to collect data. A pixel level labelled binary map is assigned to each image.

3) CFD Dataset: The *CFD* [19] is a crack dataset containing 118 480 $\times$ 320 images. The images are captured by iPhone and the profiles of the cracks are manually annotated.

4) AELLT Dataset: The *AELLT* [20] is a crack patterns dataset containing 58 crack images of 256 $\times$ 256 pixels. The contours of the crack are annotated therefore it can be utilized for crack segmentation.

5) Cracktree260 Dataset: The *cracktree260* [18] is a dataset containing 260 images of size 800 $\times$ 600 with various cracks. The annotation of this dataset gives pixel-wise label, which is very convenient to be used directly for training and testing.

After adding our own data, the crack segmentation dataset consists of approximately 11300 images. To the best of our knowledge, this is the biggest dataset for crack segmentation on the internet so far. All the images are resized to 256 $\times$ 256 for training purpose. The final training set consists of 9000 images and the testing set consists of 1500 images.

C. The Evaluation Metrics

The metrics to evaluate the segmentation performance are listed as follows:

- The Global Pixel Accuracy (A), which is define as the percentage of the correctly predicted pixel.
- The Mean Intersection Over Union (MIOU), which is a significant metric in semantic segmentation.
- The best F-measure on a dataset based on a fixed threshold (DS)
- The aggregated F-measure score on a dataset for the best threshold in image (IS)
- The precision at the best F-measure DS (BP)
- The recall at the best F-measure DS (BR)

The detailed definition of them are shown as follows: The global pixel accuracy (A) and mean intersection over union (MIOU) are well defined in [22]. Denote the number of true positive, true negative, false positive, false negative samples as N_{TP}, N_{TN}, N_{FP} and N_{FN} respectively, the precision (P) can be represented as:

$$\mathrm{P} = \frac{N_{TP}}{N_{TP} + N_{FP}}$$

And the recall (R) can be represented as:

$$\mathrm{R} = \frac{N_{TP}}{N_{TP} + N_{FN}}$$

Then the F-measure can be represented as:

$$F - \mathrm{measure} = \frac{2PR}{P + R}$$

And the best F-measure on a dataset for a fixed threshold m (DS) can be written as:

$$\mathrm{DS} = max\{\frac{2P^m \times R^m}{P^m + R^m}, : m = 0.01, 0.02, 0.03, ..., 0.99\}$$

The precision and recall at the best F-measure are denoted as BP (Best Precision) and BR (Best Recall) respectively. While the aggregated F-measure score on a dataset for the best scale in image (IS) can be given as:

$$\mathrm{IS} = \frac{1}{N_{im}} \sum_{i=1}^{N_{im}} max\{\frac{2P_i^m \times R_i^m}{P_i^m + R_i^m} : m = 0.01, 0.02, 0.03, ..., 0.99\}$$

where m denotes the threshold, i denotes the index of the image and N_{im} denotes the total amount of images. P_m and R_m are the precision and recall at the threshold m. P_i^m and R_i^m denote the precision and recall of image i.

D. The Comparison of Segmentation Results Between Different Side Outputs

TABLE II
THE COMPARISON OF SEGMENTATION RESULTS BETWEEN DIFFERENT SIDE OUTPUTS

Output	A	MIOU	BP	BR	DS	IS
Side 1	94.9	82.8	81.1	79.8	80.4	81.1
Side 2	95.6	85.3	85.3	81.7	83.5	84.2
Side 3	95.2	84.7	83.2	82.8	83.0	83.8
Side 4	94.7	82.2	78.3	81.0	79.6	80.9
Side 5	91.8	76.5	66.3	77.7	71.5	72.3
CrackNet	96.3	86.8	84.8	84.6	84.7	85.8
CrackNet (With GF)	**96.8**	**87.7**	**86.6**	**84.8**	**85.7**	**86.5**

Firstly, we did visualization of different convolutional layers and compared the segmentation results. The final segmentation prediction of different side output layers, fused and GF refined results are shown in Figure 5. The side outputs of the shallower layers gives fine prediction of the crack contour, but are sensitive to noises. While the outputs of the deeper layers can successfully filter the local noises in the image, but the contours for the cracks are blurred. As shown in Table II, the best segmentation result of the sides output feature maps is given by the side output of 2nd and 3rd layer. And the fused result with guided filter post-processing gives the best segmentation performance if trained with the best loss function weights at different level, which will be discussed below in Subsection E.

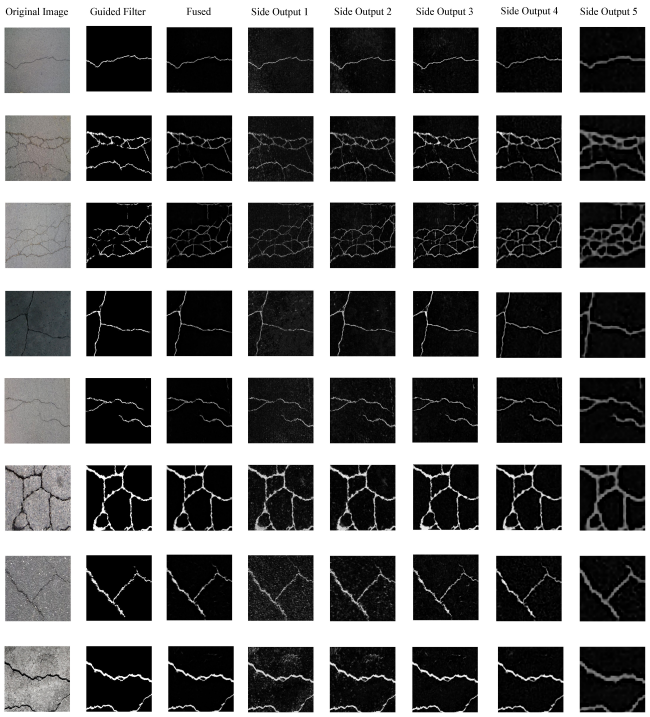

Fig. 5. Segmentation results of the side outputs, fused results and guided filter results

E. The Comparison of Loss Function Weights at Different Level

The hyper-parameter setting in the loss function to weight losses between different side outputs is very important to give a better semantic segmentation result. However, it is impossible to try all combinations of parameters so we choose some typical parameters settings to evaluate the influence of them on final prediction results. In the first 2 cases, we give larger weights to the shallower layers, while in the 3rd and 4th cases, larger weights are given to the deeper layers. In the 6th and 7th cases, largest weights are given to the 2nd layer and 3rd layer, which gives the best feature map prediction among different layers. While in the 5th case, the weights are equal among different layers. The parameter setting of different cases are:

- Case 1: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{4.0, 2.0, 1.0, 0.5, 0.25\}$
- Case 2: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{9.0, 3.0, 1.0, \frac{1}{3}, \frac{1}{9}\}$
- Case 3: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{0.25, 0.5, 1.0, 2.0, 4.0\}$:
- Case 4: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{\frac{1}{9}, \frac{1}{3}, 1.0, 3.0, 9.0\}$:
- Case 5: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{1.0, 1.0, 1.0, 1.0, 1.0\}$:
- Case 6: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{0.3, 0.7, 1.0, 0.7, 0.3\}$:
- Case 7: $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\} = \{0.5, 1.0, 0.8, 0.5, 0.3\}$:

As shown in Table III, from the results, we can see that giving larger weights to the deeper layers or shallower layers did not contribute to improve the segmentation performance. While giving larger weight to the 2nd and 3rd layer gives the best performance. This is corresponding to the fact that the best feature map prediction is given by the 2nd and 3rd layer. And more significance should be given to the loss function of them to give a better segmentation performance.

TABLE III

THE COMPARISON OF ASSIGNING DIFFERENT WEIGHTS TO THE LOSS AT DIFFERENT LEVEL OF FEATURE MAP

Case	A	MIOU	BP	BR	DS	IS
Case 1	94.4	83.1	82.1	80.8	81.4	82.9
Case 2	94.5	82.8	81.9	81.2	81.5	82.4
Case 3	94.6	83.3	82.2	81.5	81.8	83.3
Case 4	94.7	83.8	82.7	81.2	81.9	82.9
Case 5	95.8	86.5	84.3	83.8	84.0	84.5
Case 6	96.2	86.9	85.6	84.3	84.9	85.7
Case 7 ***(best)***	**96.8**	**87.7**	**86.6**	**84.8**	**85.7**	**86.5**

F. The Comparison with other Methods

In order to verify the effectiveness of our method, the results of our *CrackNet* is compared with 4 other very popular state-of-art CNN architectures in semantic segmentation.

1) HED [26]: The *HED* is a multi-level feature learning network which shows great performance in edge detection.

2) Segnet: The *Segnet*[23] is a deep convolutional encoder-decoder architecture for Image Segmentation.

3) FCN: The *FCN*[22] is a fully connected network which also shows its performance in semantic segmentation.

4) U-Net: The *U-Net*[27] is a network which also has the encoder-decoder architecture, and it is initially utilized for biomedical image segmentation.

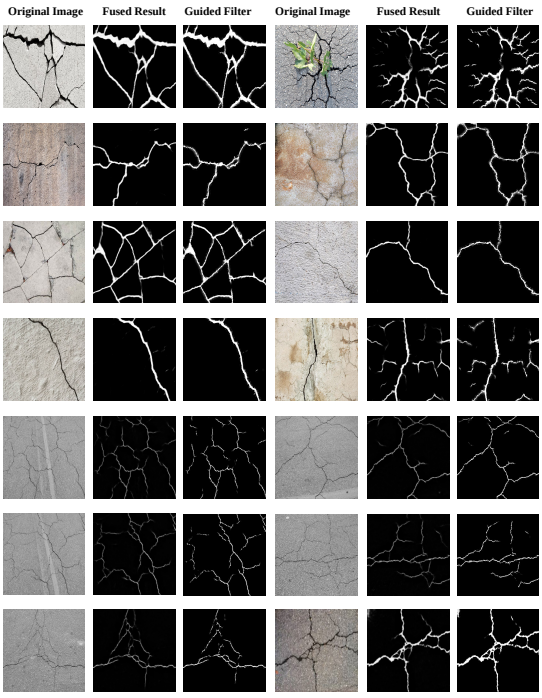

Fig. 6. Segmentation results in some challenging circumstances with various cracks and noises

Our *CrackNet* is specially designed for crack segmentation with pre-processing and post-processing methods as well as fine tuned weight given in Section IV. From the results in Table IV, it is demonstrated that *CrackNet* with guided filtering post-processing and the best loss function weight

TABLE IV

THE COMPARISION OF SEGMENTATION RESULTS BETWEEN METHODS

Methods	Interference Times/(ms)	Threshold (m)	A	MIOU	BP	BR	DS	IS
HED [26]	67	0.47	92.8	75.9	74.5	79.3	76.8	77.5
Segnet [23]	192	0.55	96.0	85.5	83.3	84.8	84.0	84.6
FCN-8s [22]	105	0.57	93.1	75.3	74.2	75.5	74.8	75.9
U-Net [27]	102	0.56	96.1	85.9	84.0	84.6	84.3	85.1
CrackNet (Ours)	132	0.48	**96.8**	**87.7**	**86.6**	**84.8**	**85.7**	**86.5**

gives better performance than other methods in crack segmentation task. As shown in Figure 6, the detection results in some challenging circumstances with various cracks and noises also demonstrate the effectiveness and robustness of our method.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we have proposed a formal and systematic solution to the automatic crack detection and segmentation for UAV inspections. More importantly, a benchmark dataset is also established and open-sourced for the community. In the future, we plan to concentrate on the integration of the sub-components and subsystems of the unmanned aerial system. More typical images of the concrete cracks will be merged into the dataset with our research project going on. The concrete defects are not restricted to crack and specific inspection techniques should be developed for other structural damages. With the advancement of CNN architecture for semantic segmentation, we also plan to develop more advanced CNN architectures for better performance in the crack detection and segmentation task.

REFERENCES

[1] D. Zhu, T. Li, D. Ho, T. Zhou, and M. Q. Meng, "A novel ocr-rcnn for elevator button recognition," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3626–3631.

[2] D. Zhu, T. Li, D. Ho, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*. IEEE, 2018.

[3] D. Zhu, Y. Du, Y. Lin, H. Li, C. Wang, X. Xu, and M. Q.-H. Meng, "Hawkeye: Open source framework for field surveillance," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6083–6090.

[4] P. Prasanna, K. J. Dana, N. Gucunski, B. B. Basily, H. M. La, R. S. Lim, and H. Parvardeh, "Automated crack detection on concrete bridges," *IEEE Transactions on Automation Science and Engineering (TASE)*, vol. 13, no. 2, pp. 591–599, 2014.

[5] C. V. Dung *et al.*, "Autonomous concrete crack detection using deep fully convolutional neural network," *Automation in Construction*, vol. 99, pp. 52–58, 2019.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012, pp. 1097–1105.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *The International Conference on Learning Representations (ICLR)*, 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[9] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1492–1500.

[10] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[11] F.-C. Chen and M. R. Jahanshahi, "Nb-cnn: deep learning-based crack detection using convolutional neural network and naïve bayes data fusion," *IEEE Transactions on Industrial Electronics (TIE)*, vol. 65, no. 5, pp. 4392–4400, 2017.

[12] L. Yang, B. Li, W. Li, Z. Liu, G. Yang, and J. Xiao, "Deep concrete inspection using unmanned aerial vehicle towards cssc database," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 24–8.

[13] G. Cai, B. M. Chen, and T. H. Lee, *Unmanned Rotorcraft Systems*. Springer Science & Business Media, 2011.

[14] G. Cai, B. M. Chen, K. Peng, M. Dong, and T. H. Lee, "Modeling and control of the yaw channel of a uav helicopter," *IEEE Transactions on Industrial Electronics (TIE)*, vol. 55, no. 9, pp. 3426–3434, 2008.

[15] B. M. Chen, *Robust and H_∞ Control*. Springer, 2000.

[16] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, "Road crack detection using deep convolutional neural network," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3708–3712.

[17] M. Eisenbach, R. Stricker, D. Seichter, K. Amende, K. Debes, M. Sesselmann, D. Ebersbach, U. Stoeckert, and H.-M. Gross, "How to get pavement distress detection ready for deep learning? a systematic approach," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 2039–2047.

[18] Q. Zou, Y. Cao, Q. Li, Q. Mao, and S. Wang, "Cracktree: Automatic crack detection from pavement images," *Pattern Recognition Letters*, vol. 33, no. 3, pp. 227–238, 2012.

[19] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, "Automatic road crack detection using random structured forests," *IEEE Transactions on Intelligent Transportation Systems (TITS)*, vol. 17, no. 12, pp. 3434–3445, 2016.

[20] R. Amhaz, S. Chambon, J. Idier, and V. Baltazart, "Automatic crack detection on two-dimensional pavement images: An algorithm based on minimal path selection," *IEEE Transactions on Intelligent Transportation Systems (TITS)*, vol. 17, no. 10, pp. 2718–2729, 2016.

[21] K. Wada, "labelme: Image Polygonal Annotation with Python," https://github.com/wkentaro/labelme, 2016.

[22] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[23] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 12, pp. 2481–2495, 2017.

[24] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, vol. 35, no. 6, pp. 1397–1409, 2012.

[25] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2650–2658.

[26] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2015, pp. 1395–1403.

[27] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-assisted Intervention (MICCAI)*. Springer, 2015, pp. 234–241.