Autonomous unmanned aerial vehicles and deep learning-based damage detection

By Dong Ho Kang

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

In partial fulfilment of the requirements of the degree of

Doctor of Philosophy

Civil Engineering

University of Manitoba

Winnipeg, Manitoba Canada

Abstract

Infrastructure failure causes the loss of human lives and high socio-financial costs. Due to the continuous aging of infrastructure, a proper structural health monitoring (SHM) system is required to ensure the safety of structures and reduce repair costs through the early detection of structural damage. Existing visual inspection methods are not reliable due to the low frequency of inspection, subjective evaluation of structural damage, and vulnerability of inspectors' safety, along with high costs. Traditional damage detection methods have similar limitations, since they require a large number of sensors to monitor large-scale infrastructure and involve high levels of uncertainty due to environmental noises and sensor malfunctions.

Computer vison techniques have been implemented to overcome the limitations mentioned above, relying on image processing algorithms to extract damage-sensitive features. However, it is very difficult to extract a robust damage-sensitive feature. To resolve this limitation, I developed two deep learning-based damage detection methods using computer vision. The first method is a hybrid pixel-level crack segmentation and quantification method for complex cracks on rough scenes. The developed hybrid method provides robust damage detection for images, which addresses the uncertainties of traditional approaches. The second method is a real-time semantic transformer representation network (STRNet) for crack segmentation. The proposed STRNet can process 49 images per second with a mean intersection over union score of 92.6, which represents state-of-the-art performance in this area when it comes to accuracy.

Using advanced deep learning methods and computer vision for damage detection still requires a great number of cameras to monitor large-scale infrastructure, which can be expensive. Consequently, another achievement of this thesis is that I developed an autonomous flight method using unmanned aerial vehicles (UAVs) for SHM purposes. Some critical parts of the bridge system, which should be monitored, are located beneath the bridge deck where global positioning system (GPS) signals are very weak or not available. Therefore, a three-dimensional pseudo map was developed using an inexpensive ultrasonic beacon system to replace the GPS signals for the autonomous flight of the UAVs.

Co-authorship

This thesis follows the grouped manuscript format of the University of Manitoba Faculty of Graduate Studies. The majority of this thesis consists of the following published journal articles:

- Kang, D. and Cha, Y.J., 2018. Autonomous UAVs for structural health monitoring using deep learning and an ultrasonic beacon system with geo-tagging. Computer-Aided Civil and Infrastructure Engineering, 33 (10), 885-902, [Chapter 3].

  The candidate contributed to conceiving the proposed methods, implemented the coding in Python and C++ software, and performed multiple experiments. The candidate also wrote the draft of the paper and responded to reviewer comments.

- Kang, D., Benipal, S.S., Gopal, D.L. and Cha, Y.J., 2020. Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning. Automation in Construction, 118, 103291, [Chapter 4].

  The candidate contributed to improving the proposed methods. The candidate implemented the coding in Python and MATLAB, after which implementation was tested using the prepared data. The candidate also wrote the draft of the paper and responded to reviewer comments.

- Kang, D. and Cha, Y.J., 2020. Efficient attention-based deep encoder and decoder for automatic crack segmentation, Structure health monitoring, Sage, (accepted) [Chapter 5].

  The candidate contributed to conceiving the main idea of the paper and collected the dataset used for experimentation. The candidate implemented coding in Python, after which implementation was tested using the prepared dataset. The candidate also wrote the draft paper.

Acknowledgements

I would like to express my sincere gratitude to my entire family. I will do my best to support my family in the future. I also want to express my gratitude to Dr. Cha, Dr. Ho and Dr. Kavgic, who provided me with advice and reviewed my research, especially Dr. Cha, who supported me as my supervisor during the past five years of my PhD program.

I appreciate my current and former colleagues, Mr. Ali, BeckMan, Benipal, Choi, Epp, Ghorbani, Mahmoudkahani, Wang and Miss Suh, especially, Mr. Ali, who supported a lot of my requests to prepare the experiment.

I appreciate all of my Korean friends who have supported me during my international studies. Woosik Moon has added to my home life as a good roommate. Sungwook Choi and Kiyong Park are always supporting me even when I live in different time zones. I also appreciate the help from Jiyeon Park in collecting the crack image dataset of South Korean structures.

Finally, I express thanks for support for this research from an NSERC Discovery Grant (RGPIN-2016-05923), NSERC Engage Grant (EGP/515025-2017), NSERC Engage Grant (533690 - 2018), Research Manitoba 2018 New Investigator Operating Grant (3481) and CFI JELF Grant (37394).

Table of Contents

List of Tables

List of Figures

Chapter 1 Introduction

This chapter discusses the background on structural damage identification using computer vision, deep learning and UAVs, specifies the scope of this thesis, and describes the object of research, contribution of the studies, and overall outline of the thesis.

1.1 Background

The American Society of Civil Engineers (ASCE) reports that school facilities and dams in North America require continuous monitoring to prevent any sudden failures since many of the structures are nearing the end of their usual lifespan. However, there is a lack of information about the actual status of the structures (ASCE, 2021). The failure of infrastructure is disastrous due to the possibility of fatal consequences for human lives as well as high financial losses. For example, in Italy in 2018, the Morandi Bridge collapsed and caused the death of 43 people. Replacing old and deteriorated infrastructure comes at a significant expense in terms of time and financial cost. Therefore, early detection of structural damage through proper structural health monitoring (SHM) systems is needed to prevent the fatal collapse of structures and reduce overall repair and rehabilitation costs.

There are many different approaches for detecting structural damage, but they are usually classified as either physics model-driven or data-driven approaches. Physics model-driven approaches assess structural status based on multi-physics models and measured sensor data. Data-driven approaches use sensor networks deployed on structures of interest, and machine learning or deep learning algorithms are applied to identify damage in the sensor network data. These two approaches are vulnerable to environmental noises and sensor malfunctions, which can mask small cracks in critical components of structural systems (Feng et al., 2015). It is difficult to validate the causes of abnormal signals, whether they are due to actual damage or uncertainties, including noises (Cha et al., 2016). Engineers must conduct on-site visits to identify the causes of abnormal signals, which have very high frequencies. Therefore, these approaches are not reliable overall.

To overcome the limitations of the two approaches mentioned above, visual inspection conducted by trained engineers is still the main approach for monitoring and detecting structural damage (Abdel-Qader et al., 2003). However, there are two major problems. First, the accuracy of this method largely depends on the skills and experience of engineers. Highly-skilled engineers are limited in number and cannot inspect numerous structures. Additionally, personnel bias affects the interpretation of structural damage status. Second, some aspects of infrastructure are difficult to access, such as beneath a bridge deck, which calls for the use of special vehicles to access various critical components (e.g., expansion joints, bearings, connections). Therefore, bridge should be closed for inspection, which leads to low frequency inspections. Most countries

worldwide perform biannual inspections, which are not sufficient to prevent the sudden failure of bridge systems (Darby et al., 2019; Owen, 2007).

Computer vision-based structural damage detection methods have been suggested to overcome the limitations of visual inspections conducted by trained engineers (Abdel-Qader et al., 2003). Early stage computer vision-based damage detection methods were developed using simple image processing techniques. Specifically, computer vison data (e.g., RGB images) are processed using various image processing algorithms to extract damage-sensitive features. The extracted damage sensitive-features are then classified by trained machine learning algorithms. For example, in order to detect loosened bolts, the horizontal and vertical lengths of bolt heads can be calculated using image processing. Those dimensions can then be extracted as a damage-sensitive feature to train a linear support vector machine algorithm (Cha et al., 2016; Ramana et al., 2019). As damage-sensitive features, hue saturation and intensity that have been modified from a color space value can be used to train a decision tree algorithm and detect steel corrosion (Son et al., 2014). The characteristics of concrete crack shapes have also been used as damage-sensitive features that were then classified by a support vector machine and neural network (Jahanshahi et al., 2013).

However, these early stages of computer vison-based damage detection methods showed poor performances under the influence of uncontrolled environmental factors, such as lighting, shadows, and blurry images (Cha et al., 2017). In particular, it is difficult to select or extract damage-sensitive features that are robust to changing-light conditions. Traditional computer vision and machine learning-based damage detection methods require an engineer's careful intervention to choose or extract proper damage-sensitive features, since they mostly fail to identify features that are robust to changing-light conditions, and only work for a single type of damage (Cha et al., 2017).

Recently, Cha et al. (2017) suggested a deep convolutional neural network (CNN) (LeCun, 1998) method for structural concrete crack detection that yielded 97% accuracy under various lighting intensities and conditions. Compared to the traditional machine learning approaches, a deep CNN is able to extract a robust damage-sensitive feature through training processes. Specifically, a deep CNN can be trained with a great number of images that have been subjected to various lighting conditions, such as spot lightening, shadows, and blurriness. As a result, a well-trained CNN yields a very robust performance compared to existing algorithm-based image

processing and machine learning methods. Through comparative studies, a well-trained CNN showed superior performances compared to the Canny and Sobel edge detection algorithms, which are traditional image processing methods (Cha et al., 2017). Interest in the CNN approach within SHM research has increased due to its accuracy and robustness.

1.2 Problem definition

The CNN-based damage detection method (Cha et al., 2017) is an innovative method to overcome the limitations of the traditional computer vision and machine learning-based methods. It uses a sliding window technique to localize detected damage within an input image. The sliding window technique uses a sliding window with a fixed size, but there are various shapes and sizes of damage, such as cracks, corrosion and loosened bolts, and the sizes are always different within the input RGB images, dependent on the distance between the camera and the object, as well as the properties of the camera, such as the type of lens and image sensors. Another limitation is that the CNN-based damage detection method was only applied on a pure monotonous concrete surface, and not on complex scenes. Usually, structures and facilities are located in scenes with complex backgrounds. Damage detection in complex scenes is a very challenging problem; it is not easy to extract damage-sensitive features, since there are many damage-like (i.e., crack-like) features, such as electric wires and expansion joints. Therefore, a new way of damage detection using CNN is required to accurately quantify detected damage in complex scenes. Accurate quantification of detected damage is essential to evaluate the status of a structural system in terms of safety.

Another critical problem of computer vision-based damage detection is that although the approaches can detect structural damage, they require a large number of image sensors (i.e., cameras) to monitor large-scale civil infrastructure, which is expensive and computationally intensive. Recently, UAVs have been applied in the field of SHM to reduce time and costs (Metni and Hamel, 2007; Eschmann et al., 2012; Hallermann et al., 2013; Sankarasrinivasan et al., 2015). UAVs allow data to be obtained easily and remotely for SHM (Metni and Hamel, 2007; Hallermann et al., 2013). Nevertheless, well-trained pilots are required to operate the UAVs, which is also expensive and limited as far as the time it takes to cover numerous bridges and building systems. Another limitation is that all manual flights of UAVs rely on global positioning system (GPS) technology to localize detected damage. However, some critical parts of bridge systems that

are vulnerable to structural damage and can cause sudden collapses are located beneath the bridge deck. GPS signals become weak and unreliable beneath bridge decks for the autonomous flight of UAVs. Therefore, a more efficient, inexpensive and reliable method for UAV flight is required to autonomously navigate GPS-denied areas for SHM purposes.

1.3 Research objectives

As discussed above, there are limitations to using traditional damage detection approaches for an actual large-scale bridge system. There are also limitations to using computer vision-based damage detection due to the flight mode of UAVs. Specifically, the traditional manual flight of UAVs is limited when it comes to inspecting numerous bridge systems, since it requires on-site visits and careful, skillful manipulation, which are cumbersome. The CNN-based damage detection method showed robust performance under various lighting conditions, but its capacity for localization and quantification needs to be improved significantly.

In this thesis, an autonomous damage detection system has been developed by integrating advanced CNN methods and the autonomous flight of UAVs. To realize this goal, the three objectives below have been defined.

1. Develop an autonomous UAV flight method for the GPS-denied regions of bridge systems.
2. Develop a hybrid computer vision-based crack segmentation method that can detect concrete cracks at the pixel level in complex scenes by integrating image processing and region-based CNN methods.
3. Develop a real-time crack segmentation method using an encoder and decoder-based end-to-end CNN method to reduce the inspection costs of autonomous UAVs.

As the result, the autonomous damage inspection UAV is developed in this research.

1.4 Scope of work

The thesis is composed of six chapters. This chapter (i.e., Chapter 1) provides background information on the research topics related to computer vision-based damage detection and UAV application in the field of SHM. Chapter 1 also defines the problems that this thesis will investigate, and describes the goals, objectives and contributions of the thesis. Chapter 2 provides an extensive

literature review and offers an overview of the mathematical backgrounds of UAV flight control and encoder-decoder-based deep learning algorithms. Chapter 3 describes autonomous flight UAV methods and deep learning-based crack damage detection. Chapter 4 describes the limitations of existing deep learning-based damage detection methods and suggests a hybrid crack segmentation algorithm for complex scenes with an improved crack quantification method. Chapter 5 proposes a real-time crack segmentation method using an advanced deep end-to-end CNN network and shows its state-of-the-art performance in crack segmentation. Chapter 6 provides a summary of this study and makes proposals for future research. Table 1-1 shows the objectives and scope of this thesis.

Table 1-1. Scope of the thesis

	Scope	Validation
Chapter 2	Mathematical backgrounds of autonomous UAV flight and deep learning-based damage (crack) detection	
Chapter 3 Kang & Cha, (2018)	Develop an ultrasonic beacon system (UBS)-based autonomous flight algorithm and fabricate physical UAVs for SHM in GPS-denied areas. Present a deep learning-based crack detection method from the UAV images.	Real flight experiments in an indoor GPS-denied environment. Test the deep learning algorithm using images of scenes with cracks that were taken by UAVs.
Chapter 4 Kang et al., (2020)	Suggest a hybrid algorithm to reduce the cost of data acquisition for crack segmentation. Develop a crack quantification algorithm.	Perform a comparison test between existing published methods and the developed algorithm using approximately 100 crack images.
Chapter 5 Kang & Cha, (2021)	Explore the attention-based encoder and decoder for crack segmentation. Suggest an efficient ground truth data acquisition method and build a large and high-quality dataset.	Build an extensive and real-world image database. Perform a comparison between the suggested algorithm and a well-known algorithm.

	Suggest a synthetic image technique for extra augmentation.	

1.5 Research contributions

I have achieved three major technical contributions related to the automation of SHM systems. 1) For the first time, I developed an autonomous flight method for UAVs to monitor structures in GPS-denied environments. 2) I developed a hybrid method of crack segmentation by integrating image processing methods with faster region-based CNN (Faster R-CNN), and developed a crack quantification method to calculate the width and length of cracks. 3) I developed a new deep CNN method for the real-time crack segmentation of complex scenes. The details of the scientific contributions of this thesis are described below.

Even though there are some existing applications of UAVs for SHM purposes, all of those implementations rely on skilled pilots. However, we cannot rely on skilled pilots to monitor numerous bridge systems, since they must conduct on-site visits for every inspection. This makes the work costly and cumbersome, which reduces the frequency of inspections. Therefore, I developed an autonomous flight method for UAVs, which is described for the first time in Chapter 3. Further, while most existing autonomous flight methods for UAVs are based on GPS, some critical structural components that are vulnerable to damage are located in GPS-denied areas, such as beneath a bridge deck. Therefore, another contribution of this thesis is that I introduced a UBS system to replace the GPS system to localize UAVs and eventually localize detected damage with a geo-tagging method.

The autonomous UAV system was developed for GPS-denied areas, and the collected videos from the UAV were processed using traditional CNN with the sliding window method described in Chapter 3. However, the sliding window technique is not sufficient for evaluating detected damage since it depends on fixed-size bounding boxes to localize detected cracks. Detected damage must be quantified to evaluate the status of structures. Therefore, a hybrid pixel-level crack segmentation method was developed for complex scenes (see Chapter 4). The hybrid method was developed by carefully integrating image processing methods (i.e., a modified tubularity flow field (TuFF)) (Mukherjee et al., 2015) and Faster R-CNN (Ren et al., 2015). Compared to the traditional CNN-based method using the sliding window technique, the hybrid

method can segment cracks in complex scenes at the pixel level. The output of the hybrid method is also processed using a damage quantification method that I developed for the calculation of crack width and length.

The limitation of the hybrid method that is introduced in Chapter 4 is that it is not a real-time method. When the autonomous UAV system collects video of structures, the hybrid method depends on post-processing the videos to segment cracks. This entails high computational costs and delays the inspection process. Therefore, I developed a real-time crack segmentation method to process videos taken by autonomous UAVs by designing a new deep learning architecture and implementing various advanced deep learning operators, such as an attention-based encoder and decoder model and focal Tversky loss function (Abraham & Khan, 2019). The new architecture (i.e., semantic transformer representation network (STRNet)) achieved high accuracy with a mean intersection over union (mIoU) score of 92.5% for 47 frames per second (FPS), which is the fastest speed that has been achieved by pixel-level crack segmentation methods.

Chapter 2. Mathematical backgrounds

The UAV and the encoder-decoder based CNNs have various operations. This thesis follows the grouped paper format, with most methodologies explained in Chapters 3, 4, and 5. This chapter is intended to offer further explanation of the mathematical background of this thesis. The proportional–integral–derivative control for UAV and the components of CNN for damage detection are explained in this chapter.

UAVs and deep CNNs for crack or damage detection have been actively researched in recent years. The UAV based structural damage detection system had two major problems. First, most of the UAV research rely on the trained human pilot or GPS-based system (Metni & Hamel, 2007; Eschmann et al., 2012; Morgenthal & Hallermann, 2014; Gillins et al., 2016). Metni & Hamel, (2007) applied the UAV for bridge inspection as the cost-efficient approach. Eschmann et al., (2012) applied the building inspection using GPS-based waypoint navigation flight. However, Morgenthal & Hallermann, (2014) pointed out the limitation of UAV based SHM, such as camera performance, weather conditions, and GPS signal interference. Gillins et al., (2016) also pointed out the GPS signal problem, especially the bridge inspection. During the bridge inspection, the girder of the bridge is hard to inspect because the concreted structure blocks the GPS signal. Most of the UAVs is calculated the current position using by GPS signal. This data is applied to the feedback control loop to fix the error of UAV movement. If the GPS data is not accurate, the autonomous flight would fail during the flight.

In the robotics field, overcoming GPS denied environment has been one of the big topics for autonomous flight UAV. There were various approaches to replace the GPS. However, one of the most stable solutions is the beacon system (Seki et al., 2000; Silvagni et al., 2017; Díaz et al., 2017). A beacon system is composed of multiple stationary beacons and a mobile beacon. Stationary beacons have become the benchmark for distance measurement. This beacon system is applied to various environments. Silvagni et al., (2017) applied the beacon system to calculate the distance between UAVs and victims. Seki et al., (2000) applied the autonomous driving wheelchair-using by the beacon. Also, the current beacon system achieves the cm accuracy (Díaz et al., 2017). The beacon system can be applicable for autonomous structural damage detection systems in the GPS denied environment based on previous research.

Even if the UAV successfully flight autonomously, the structure damage should be detected automatically for the autonomous damage detection system. Structural damage detection with UAV is majorly applied to the image processing technique (Metni & Hamel, 2007; Eschmann et al., 2012; Sankarasrinivasan et al., 2015). However, the UAV application for structural damage detection is the outside environment, and it is hard to control the lighting condition. Metni & Hamel, (2007) applied the color-based threshold approach. The color-based threshold is dependent on the environment, and especially the decolorization is affected by the structure type and lighting

condition. The engineer needs to find and change the specific value of color each time, and they apply the different structures and times. Eschmann et al., (2012) applied edge detection to detect the concrete crack. However, before the edge detection is performed, engineers need to check which image has the concrete damage. Sankarasrinivasan et al., (2015) suggested the ensemble algorithms which combined the multiple algorithms to increase the performance. The two different algorithms, which were hat transform and HSV thresholding were combined to make the crack segmentation map. The image processing software was processed by a telemetry-equipped ground station. The system performed the real time structural damage detection system. However, the engineers need to select the threshold value based on the environment and the pilot need to zoom the structure damage area during the flight.

The image processing technique has the possibility of parts of automated damage detection, but it is affected by environmental changes. Especially the UAV based SHM research is the outdoor environment, and it is difficult to constrain the lighting conditions (Morgenthal & Hallermann, 2014). To overcome this problem, CNN which is one of the deep learning methodologies, is applied for damage detection (Cha et al., 2017). Unlikely human selected feature or threshold, the deep learning-based damage detection shows the robust result about the various lighting conditions without human intervention. Not only the concrete crack damage, but the multiple types of structure damage can also be detected such as concrete crack, steel corrosion, delamination, and bolt corrosion (Cha et al., 2018). For this reason, the deep learning-based damage detection can be applicable for autonomous structural damage detection system.

As a result, the two problems, which are GPS limitation and lighting conditions, could be solved by suggesting two possible approaches, which are UBS and CNN. To develop the autonomous damage inspection system, Kang & Cha, (2018) integrated the two independent solutions and proved the idea. It is successfully overcome the GPS limitation and structure damage detection in various lighting conditions. This section provides fundamental theoretical and mathematical explanations of flight control algorithms and the deep learning method, including its operators.

2.1 Feedback control theory

In this thesis, the UAV uses a proportional–integral–derivative (PID) control, which is a feedback control algorithm (Minorsky, 1922). A fundamental concept of feedback control is to generate a control signal to reduce the discrepancy (i.e., error) between measured output responses and desired output responses of the controlled system. The error is defined at time (t) as,

$$e(t) = R_m(t) - R_d(t), \tag{2-1}$$

where $R(t)$ is the selected response of the controlled system at time (t), and subscript m and d are measured and designed responses of the system. The PID control signal $u(t)$ is calculated with the help of the following equation:

$$u(t) = K_p * e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}, \tag{2-2}$$

where subscripts p, i, and d stand for proportional, integral, and derivative control, respectively. Therefore, the PID control is composed of proportional (P) control, Integral (I) control, and Differential (D) control. The K_p, K_i, and K_d are coefficients that should be determined by trial and error by considering specific control system properties.

2.2 Deep learning operators

Most deep learning networks for classification, object detection, and segmentation purposes consist of several deep learning operators to extract feature maps and eventually improve the efficacy of the network. Since I developed and used my own deep learning networks for structural damage detection, some key deep learning operators are explained in this section.

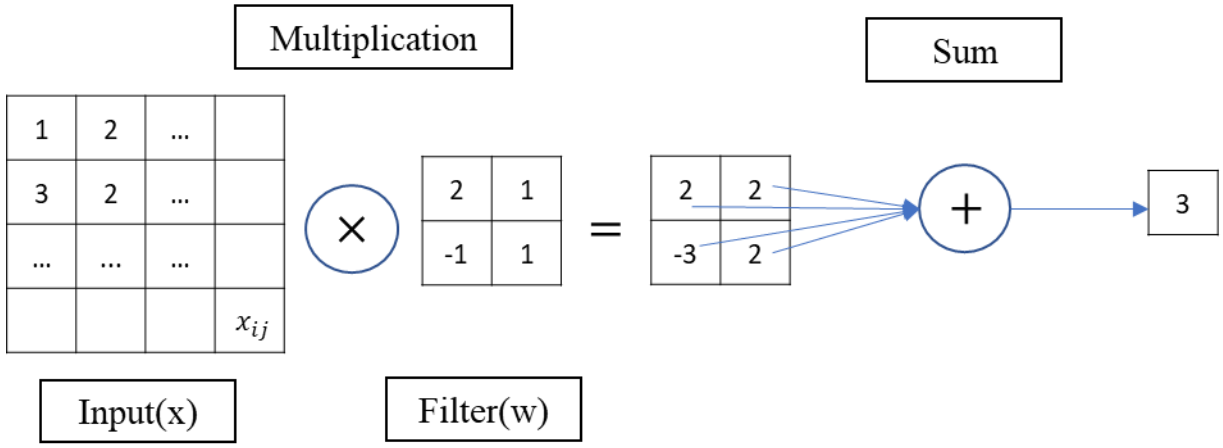
2.2.1 Convolution filter

The mathematical meaning of the convolution operator is a dot product which is the multiplication and summation of the input vector ($x_{(c,i+f_h-1,j+f_w-1)}$) with a learnable filter ($w_{(c,f_h,f_w)}$). The convolution filter calculation can be expressed as follows:

$$z_{ij} = \sum_{c=1}^{c} \sum_{f_h=1}^{n} \sum_{f_w=1}^{m} x_{(c,i+f_h-1,j+f_w-1)} w_{(c,f_h,f_w)} + b, \quad (2\text{-}3)$$

where b is the bias, c is the channel size of the filter, f_h is the height of the filter, f_w is the width of the filter, i is the height of the input, and j is the width of the input. Figure 2-1 shows an example of convolution calculation.

Figure 2-1. Example of convolution operation

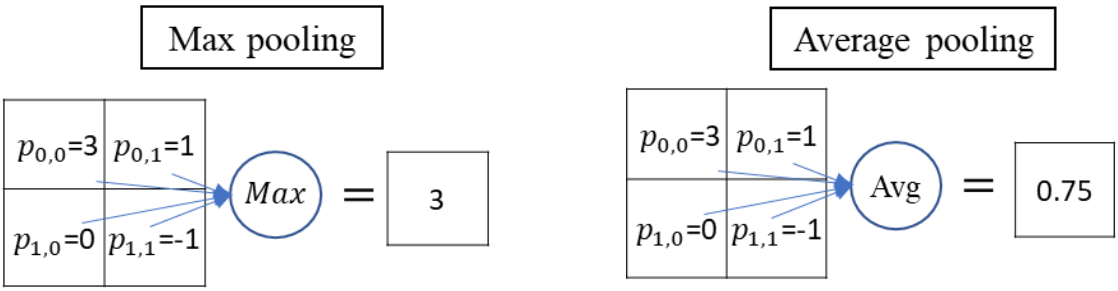


The first calculation for the convolution filter is an element-wise multiplication. The size of the filter is a 2 × 2 matrix. The first input matrix is [1, 2; 3, 2], which is also a 2 × 2 matrix. The output of the first multiplication is [2, 2; -3, 2] as shown in Figure 2-1. Following that, all the values are summed up and result in 3. Then the filter slides to the left side with one column of vector and does the same calculations. Stride is the value of how many pixels (i.e., column vectors) need to be skipped for the next same calculation. If the number of input channels is more than one, the same number of filter channels is required. The desired size of the feature map channels is defined by the number of convolution filters.

2.2.2 Pooling operation

The pooling operator is an important deep CNN operator. Pooling operation helps a network extract the important feature while reducing the spatial dimension of the output feature map.

Figure 2-2. Pooling operations (a)Max pooling, (b) Average pooling



Even though there are different pooling operations, some common and representative functions are introduced in this section. The max pooling (Nagi et al., 2011) is expressed in Equation (2-4),

$$Max\_pooling = max(p_{i,j}, \dots, p_{i+f_w,j+f_h}), \tag{2-4}$$

where p is the input, f_h is the height dimension of the filter, f_w is the width dimension of the filter, i is the height dimension of the input, and j is the width dimension of the input. Figure 2-2 (a) shows an example of max pooling. As shown in Figure 2-2 (a), the size of the filter is a 2 × 2 matrix. The input of the matrix is [3, 1; 0, -1], which is also a 2 × 2 matrix. The max pooling picks the maximum value among the input values and results in 3. The pooling operations reduce the width and height of the feature map size to reduce the computational cost.

Average pooling calculates the average value from the input values, and it is expressed as follows:

$$Average\_pooling = \frac{1}{f_h f_w} \sum_{1,1}^{f_h, f_w} p_{(i+f_w-1, j+f_h-1)}, \quad (2\text{-}5)$$

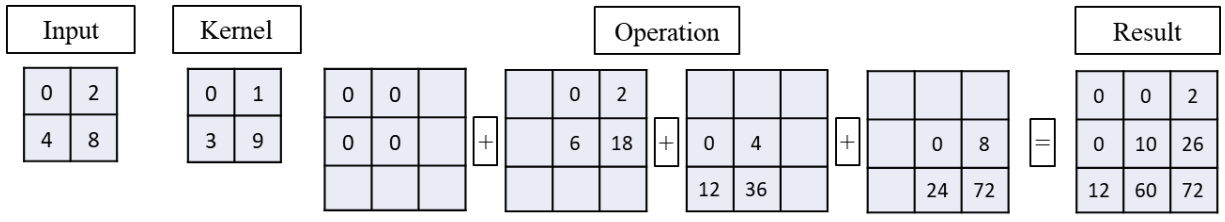
where f_h is the height of the filter, f_w is the width of the filter, i is the height of the input, and j is the width of the input. Figure 2-2 (b) shows an example of average pooling. The size of the filter is a 2 × 2 matrix. Input of matrix is [3, 1; 0, -1], which is also the same size as a 2 × 2 matrix. Following Equation (2-5), the average value is the output of average pooling and results in 0.75.

2.2.3 Transposed convolution and upsampling

In the encoder, the network is composed of convolutional filters to extract feature maps. Due to these convolution operations through the deep hidden layers of the network, there is often a reduction in the extracted feature map size. For the semantic segmentation, the reduced size of the feature map should be restored by application of transposed convolution and/or upsampling to present only the detected objects at pixel level on the original input scene (Dumoulin and Visin, 2016). The process of restoring the size of the feature map, which is the result of encoding to the original input size, is decoding.

Figure 2-3 shows an example of transposed convolution. When the input feature map and filter sizes are a [2 × 2] matrix with stride 1, transposed convolution restores the input feature map to a [3 × 3] matrix. Each element of the filter multiplies each element of the input feature map. The target unassigned [3 × 3] matrix is prepared, which simply adds the elements of the overlapping positions.

Figure 2-3. Example of transposed convolution



Although transposed convolution is applied to the decoder, the upsampling operation has been used in deep learning networks for semantic segmentation. Various upsampling techniques, including bilinear (Smith, 1981), nearest neighborhood (Rukundo and Cao, 2012), bicubic

interpolation (Keys, 1981), and Lanczos interpolation (Lanczos, 1958), have been developed. Based on the literature review, most other crack segmentation algorithms used bilinear upsampling.

Bilinear upsampling is based on linear interpolation. If x_1, x_i, and x_2 are locations in the linear function $f()$ and follow ($x_1 < x_i < x_2$) condition, the value of x_i can be calculated based on the distance between (x_1, x_i) and (x_i, x_2). The x_i value is calculated using Equation (2-10),

$$d_1 = x_2 - x_i, \quad (2\text{-}6)$$

$$d_2 = x_i - x_1, \quad (2\text{-}7)$$

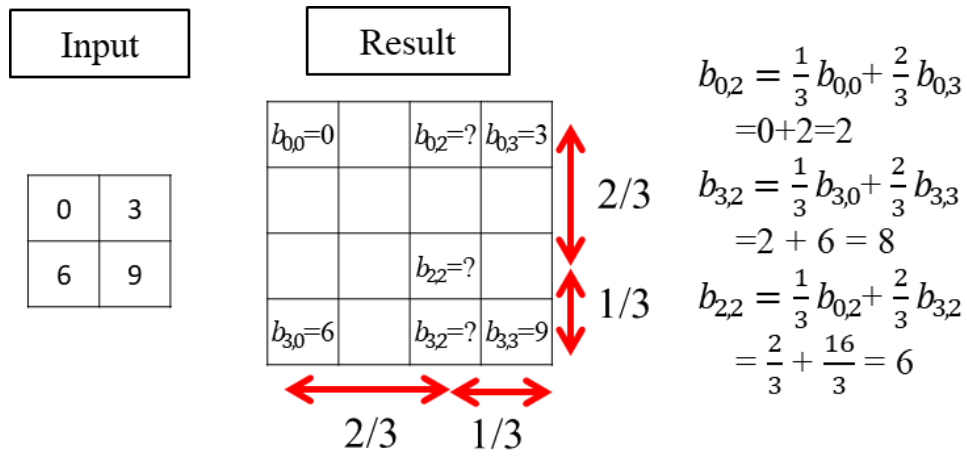
$$\alpha = \frac{d_1}{d_1 + d_2}, \quad (2\text{-}8)$$

$$\beta = \frac{d_2}{d_1 + d_2}, \text{ and} \quad (2\text{-}9)$$

$$x_i = \beta x_1 + \alpha x_2, \quad (2\text{-}10)$$

where d stands for the distance between the arrayed points (x_1, x_i, and x_2) and α and β are interpolation rates. In Figure 2-4, this concept is applied to the two-dimensional vector. Bilinear interpolation can be applied to a 2 × 2 input matrix to build a 4 × 4 output matrix, which requires a two-time interpolation process. To determine $b_{2,2}$, $b_{0,2}$ and $b_{3,2}$ should be determined first. The "1/3=$\frac{d_1}{d_1+d_2} = \frac{1}{1+2}$" and "2/3" in Figure 2-4 are interpolation rates that were calculated as α and β in Equation 2-6. Based on $b_{0,2}$ and $b_{3,2}$, we perform another linear interpolation to determine the value of $b_{2,2}$.

Figure 2-4. Calculation of bilinear interpolation



2.2.4 Batch normalization

The initial filter weights of a neural network are important in deep learning. If some of the weights are too large and others are too small in value, it is difficult to train the network. For this reason, the output of the convolution operation in each layer must be normalized (Nair & Hinton, 2010; Ioffe & Szegedy, 2015). To normalize the output, the average (μ_B) of the output and its variance (σ_B^2) are calculated as expressed in Equation (2-11) and (2-12), where m is the batch (B) size and ε is a small value to prevent an error if σ_B^2 is 0. Then, the trainable parameter scaling (c) and shift (s) values are applied for the movement of the center of distribution. This process increases the stability of network training. The batch normalization follows Equation (2-14).

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \tag{2-11}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2, \tag{2-12}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{2-13}$$

$$y_i = c\hat{x}_i + s, \tag{2-14}$$

2.2.5 Activation function

One of the important operators of deep CNN is the activation function. Activation function helps a network have non-linearity. Activation functions are required to introduce non-linearity between

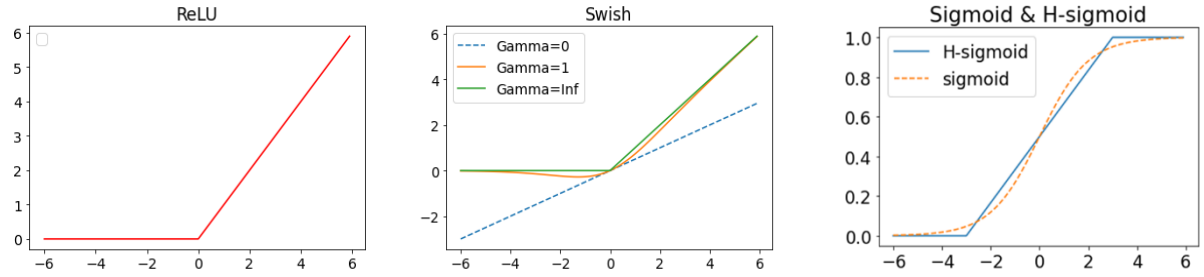
input and output. They should simultaneously facilitate the training of the deep neural network (Nair & Hinton, 2010). Therefore, activation function should have non-linearity and be simple enough for the training process. An activation function is usually implemented after the convolution filter and batch normalization.

Even though there are many different activation functions, some common and representative functions are introduced in this section. The rectified linear unit (ReLU) (Nair & Hinton, 2010) is expressed in Equation (2-15).

$$ReLU(x) = Max(0, x), \tag{2-15}$$

where x is the input value. If the input value is lower than 0, the output becomes 0. Otherwise, the output follows the input value as shown in Figure 2-6 (a). The ReLU is a commonly used activation function in CNN architectures because the value of the image is positive.

Figure 2-5. Activation functions (a)ReLU, (b) Swish, (c) Sigmoid, and H-sigmoid



The Swish activation function (Ramachandran et al., 2017) is composed of the sigmoid activation function as expressed in Equation (2-17),

$$Sigmoid(x) = \frac{1}{1+e^{-x}}, \text{ and} \tag{2-16}$$

$$Swish(x) = x \cdot sigmoid(\gamma x), \tag{2-17}$$

where γ is a trainable parameter. The influence of γ is shown in Figure 2-5. If γ is close to 0, the activation function becomes a linear function. If γ is 1, it has a small curve when the value is close to 0. Finally, when γ is a very big number (i.e., ∞), it becomes ReLU. Applying the Swish activation function allows the network to find the best activation function.

H-sigmoid (Howard et al., 2019) operates similarly to a sigmoid as presented by Equations (2-19). H-sigmoid is based on ReLU6. ReLU6 is similar to ReLU, but the maximum value in the former is limited to 6. The computational cost of ReLU6 is considerably smaller than the sigmoid function; thus, it helps accelerate the network inference speed.

$$ReLU6 = min(Max(0, x), 6), \text{ and} \tag{2-18}$$

$$H - sigmoid = \frac{ReLU6(x + 3)}{6} \tag{2-19}$$

2.2.6 Loss function

The loss function defines the error between the ground truth and the network output. Some common and representative functions are introduced in this section.

The dice coefficient (DC) loss (Milletari et al., 2016) is commonly applied for binary classification or binary segmentation. Equation (2-20) expresses the DC loss.

$$DC\_loss = 1 - \frac{2|X \cap Y|}{|X| + |Y|}, \tag{2-20}$$

where X is an output of deep neural network, Y is a ground truth, and $\cap$ is the calculation of the intersection between the two groups.

Intersection of union (IoU) is a popular evaluation metric used to measure the performance of semantic segmentation algorithms. The IoU loss directly applies the usage of evaluation metrics as the loss function (Rahman & Wang, 2016). IoU loss is calculated by following equation:

$$IoU\_loss = 1 - \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}. \quad (2\text{-}21)$$

The IoU loss calculates the ratio between the output of deep CNN and the ground truth and their combined values. IoU loss usually achieves good performance in the segmentation task, but it does not consider dataset imbalance. Dataset imbalance occurs when the target object has a small portion in the images. The image background strongly influences the training process. To solve this problem, Tversky loss is suggested. Equation (2-22) represents the Tversky loss (Salehi et al., 2017).

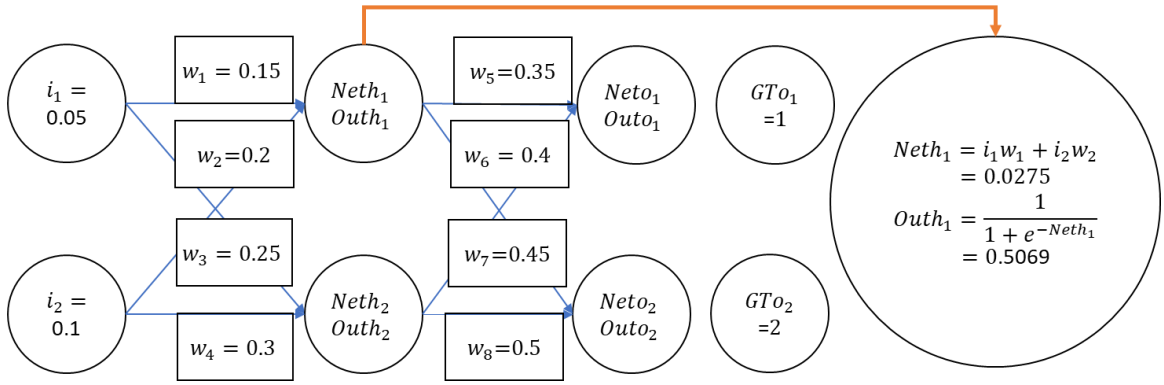
$$Tverskey\_loss = 1 - \frac{|X \cap Y|}{\alpha|X - Y| + \beta|Y - X| + |X \cap Y|}, \quad (2\text{-}22)$$

where α is the hyperparameter to enhance the false positive ($X - Y$) and β is the hyperparameter to enhance the false negative ($Y - X$). α and β are coefficients that should be determined by trial and error after considering the specific domain and dataset.

2.2.7 Training neural network

Sections 2.2.1 to 2.2.6 described the partial derivative of each operation. To train any designed deep learning network using these operators, backpropagation should be done using the gradient descent method. As a simple example, a fully connected artificial neural network is used to explain the training process as shown in Figure 2-7.

Figure 2-6. Fully connected artificial neural network



In the Figure 2-6, i_n and w_n are the input value and the weight value, respectively. Each hidden layer includes a summation of the multiplied input, weight, and sigmoid function as shown in Figure 2-6 and Table 2-1. The detailed calculation result of each node is described in Table 2-1.

Table 2-1. Forward calculation of fully connected artificial neural network

Eq.	$Neth_1 = i_1w_1 + i_1w_2$	$Outh_1 = \frac{1}{1+e^{-Neth_1}}$	$Neto_1 = Outh_1w_5 + Outh_2w_6$	$Outo_1 = \frac{1}{1+e^{-Neto_1}}$
Result	0.0275	0.5069	0.3817	0.5943
Eq.	$Neth_2 = i_1w_3 + i_2w_4$	$Outh_2 = \frac{1}{1+e^{-Outh_2}}$	$Neto_2 = Outh_1w_7 + Outh_2w_8$	$Outo_2 = \frac{1}{1+e^{-Neto_2}}$
Result	0.0425	0.5106	0.4834	0.6186

As shown in Figure 2-7, the ground truth values (GTo_1, GTo_2) are 1 and 2, respectively, but the actual output values are 0.5943 and 0.6185, respectively. Through the backpropagation process, the weight values in each layer should be updated to reduce the error between the actual output values and their ground truth values. The error is calculated by a loss function, and based on the error, the gradient descent mechanism is used to find the optimal values of the updated amount of each weight value. Equation (2-20) expresses a common L2 loss function.

$$E_{total} = E_{o1} + E_{o2} = \frac{1}{2}(1 - 0.5943)^2 + \frac{1}{2}(2 - 0.6186)^2 = 1.0364, \quad (2\text{-}23)$$

where E_{o1} and E_{o2} are the error in each output node of the last layer in Figure 2-6. The calculated total error, E_{total}, is 1.0364. To determine the increment, $\dot{w}$, of each weight value, w, Equation (2-22) is used.

$$\dot{w} = w - r\frac{\partial E_{total}}{\partial w}, \tag{2-21}$$

where r is learning rate, defined as 0.5 in this example. To minimize the E_{total}, weight $\dot{w}_5$ is updated as an example by adding the gradient value $\frac{\partial E_{total}}{\partial w_5}$.

This gradient calculation is a backward calculation to accomplish the chain rule. To calculate the w_5 with respect to E_{total}, we want to know to what extent w_5 affects the E_{total} and accordingly update the w_5 to minimize the E_{total}. The $\frac{\partial E_{total}}{\partial w_5}$ is difficult to calculate directly. Therefore, Equation (2-24) is derived for w_5. The E_{total} with respect to $Outo_1$ is calculated with the help of multiple connected steps. The $Outo_1$ with respect to $Neto_1$ is the next step. As shown in Figure 2-7, the sigmoid activation function is expressed as $Outo_n$. The derivation of sigmoid activation function follows Equation (2-26). Equation (2-27) is used to calculate the $Neto_1$ with respect to w_5. The -0.0495 is the result of the gradient value. Equation (2-29) shows the update of w_5:

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial Outo_1}\frac{\partial Outo_1}{\partial Neto_1}\frac{\partial Neto_1}{\partial w_5} \tag{2-24}$$

$$\frac{\partial E_{total}}{\partial Outo_1} = -(GTo_1 - Outo_1) = -(1 - 0.5943) = -0.4057 \tag{2-25}$$

$$\frac{\partial Outo_1}{\partial Neto_1} = Outo_1(1 - Outo_1) = 0.5943(1 - 0.5943) = 0.2411 \tag{2-26}$$

$$\frac{\partial Neto_1}{\partial w_5} = \frac{\partial(Outh_1 w_5 + Outh_2 w_6)}{\partial w_5} = Outh_1 = 0.5069 \tag{2-27}$$

$$\frac{\partial E_{total}}{\partial w_5} = -0.4057 \times 0.2411 \times 0.5069 = -0.0495 \quad (2\text{-}28)$$

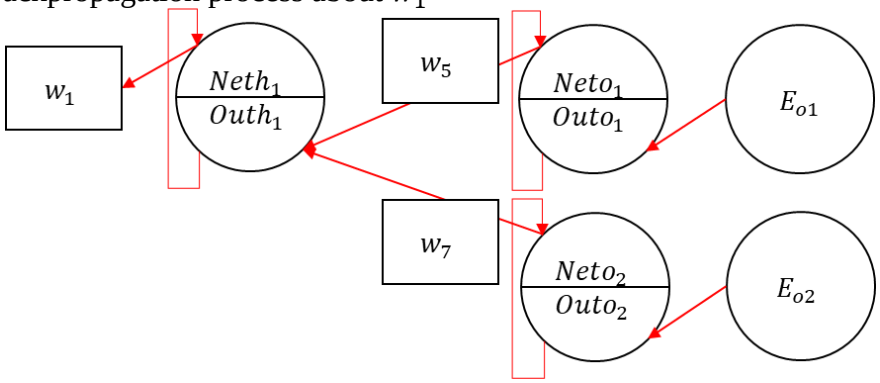
$$\dot{w_5} = 0.35 - (0.5)(-0.0490) = 0.3747 \quad (2\text{-}29)$$

The gradients from w_5 to w_8 are updated similarly as shown in Table 2-2. The weights of the first hidden layer are also updated the same way. As shown in Figure 2-7, the $w_1$is updated by two different errors and two different hidden layers.

Table 2-2. The updated weights of fully connected artificial neural network from $\dot{w_5}$ to $\dot{w_8}$

$\dot{w_5} = 0.3747$	$\dot{w_6} = 0.4249$
$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial Outo_1}\frac{\partial Outo_1}{\partial Neto_1}\frac{\partial Neto_1}{\partial w_5}$ $= (-0.4057)(0.2411)(0.5069)$ $=-0.04958$	$\frac{\partial E_{total}}{\partial w_6} = \frac{\partial E_{total}}{\partial Outo_1}\frac{\partial Outo_1 l}{\partial Neto_1}\frac{\partial Neto_1}{\partial w_6}$ $= (-0.4057)(0.2411)(0.5106)$ $=-0.04994$
$\dot{w_7} = 0.4825$	$\dot{w_8} = 0.5350$
$\frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial Outo_2}\frac{\partial Outo_2}{\partial Neto_2}\frac{\partial Neto_2}{\partial w_7}$ $= (-1.3814)(0.2359)(0.5069)$ $= -0.1651$	$\frac{\partial E_{total}}{\partial w_8} = \frac{\partial E_{total}}{\partial Outo_2}\frac{\partial Outo_2 l}{\partial Neto_2}\frac{\partial Neto_2}{\partial w_8}$ $= (-1.3814)(0.2359)(0.5106)$ $= -0.1700$

Figure 2-7. Backpropagation process about w_1



Equation (2-30) provides the details of the calculations. The E_{total} with respect to w_1 can be calculated using the chain rule based on ($\frac{\partial E_{total}}{\partial Outh_1}$, $\frac{\partial Outh_1}{\partial Neth_1}$, $\frac{\partial Neth_1}{\partial w_1}$). The gradient value, $\frac{\partial E_{total}}{\partial Outh_1}$, comes from two values ($\frac{\partial Eo_1}{\partial Outh_1}$ and $\frac{\partial Eo_2}{\partial Outh_1}$) as shown in Figure 2-9 and Equation (2-31). The two values are calculated by Equation (2-32) and (2-35). $\frac{\partial E_{total}}{\partial Neto_1}$ has been already calculated when w_5

is updated, and it is represented in Equation (2-33). Equation (3-34) represents $\frac{\partial Neto_1}{\partial Outh_1}$. Based on partial differential equations, w_5 is the $\frac{\partial Neto_1}{\partial Outh_1}$. As a result, the $\frac{\partial E_{total}}{\partial Outh_1}$ is calculated. Equation (2-35) can be calculated the same way followed by Equations (2-33) to (2-34). The $\frac{\partial Outh_1}{\partial Neth_1}$ is the derivative of sigmoid. Thus, Equation (2-38) describe the detailed calculation of $\frac{\partial E_{total}}{\partial w_1}$. After that, the w_1 is updated with Equation (2-39)

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial Outh_1}\frac{\partial Outh_1}{\partial Neth_1}\frac{\partial Neth_1}{\partial w_1}, \tag{2-30}$$

$$\frac{\partial E_{total}}{\partial Outh_1} = \frac{\partial Eo_1}{\partial Outh_1} + \frac{\partial Eo_2}{\partial Outh_1}, \tag{2-31}$$

$$\frac{\partial Eo_1}{\partial Outh_1} = \frac{\partial E_{total}}{\partial Outo_1}\frac{\partial Outo_1}{\partial Neto_1}\frac{\partial Neto_1}{\partial Outh_1} \tag{2-32}$$

$$\frac{\partial E_{total}}{\partial Neto_1} = \frac{\partial E_{total}}{\partial Outo_1}\frac{\partial Outo_1}{\partial Neto_1} = -0.4057 \times 0.2411 \tag{2-33}$$

$$\frac{\partial Neto_1}{\partial Outh_1} => w_5 Outh_1 + w_6 Outh_2 => 0.35 \tag{2-34}$$

$$\frac{\partial Eo_2}{\partial Outh_1} = \frac{\partial E_{total}}{\partial Outo_2}\frac{\partial Outo_2}{\partial Neto_2}\frac{\partial Neto_2}{\partial Outh_1} \tag{2-35}$$

$$\begin{aligned}\frac{\partial Outh_1}{\partial Neth_1} &= Outh_1(1 - Outh_1) \\ &= 0.5069(1-0.5069) = 0.25\end{aligned} \tag{2-36}$$

$$\frac{\partial Neth_1}{\partial w_1} => w_1 * i_1 => i_1 = 0.05 \tag{2-37}$$

$$\begin{aligned}\frac{\partial E_{total}}{\partial w_1} &= \left[\frac{\partial E_{total}}{\partial Outo_1}\frac{\partial Outo_1}{\partial Neto_1}\frac{\partial Neto_1}{\partial Outh_1} + \frac{\partial E_{total}}{\partial Outo_2}\frac{\partial Outo_2}{\partial Neto_2}\frac{\partial Neto_2}{\partial Outh_1}\right]\frac{\partial Outh_1}{\partial Neth_1}\frac{\partial Neth_1}{\partial w_1}, \\ &= [(-0.4057 \times 0.2411 \times 0.35) + (-1.3814 \times 0.2359 \times 0.45)] \times 0.25 \times 0.05, \\ &= -0.02034\end{aligned} \tag{2-38}$$

$$\dot{w_1} = 0.15 - 0.5(-0.02034). \tag{2-39}$$

The weights $\dot{w_2}$ to $\dot{w_4}$ can be updated in the same way followed by Equations (2-30) to (2-39). The results from $\dot{w_1}$ to $\dot{w_4}$ are calculated in Table 2-3. Based on the updated weights, the new $Outo_1$ and $Outo_2$ are 0.6004 and 0.6267, respectively. The previous outputs were 0.5943 and 0.6186. Since the ground truth outputs are 1 and 2, there is an improvement of approximately 0.006

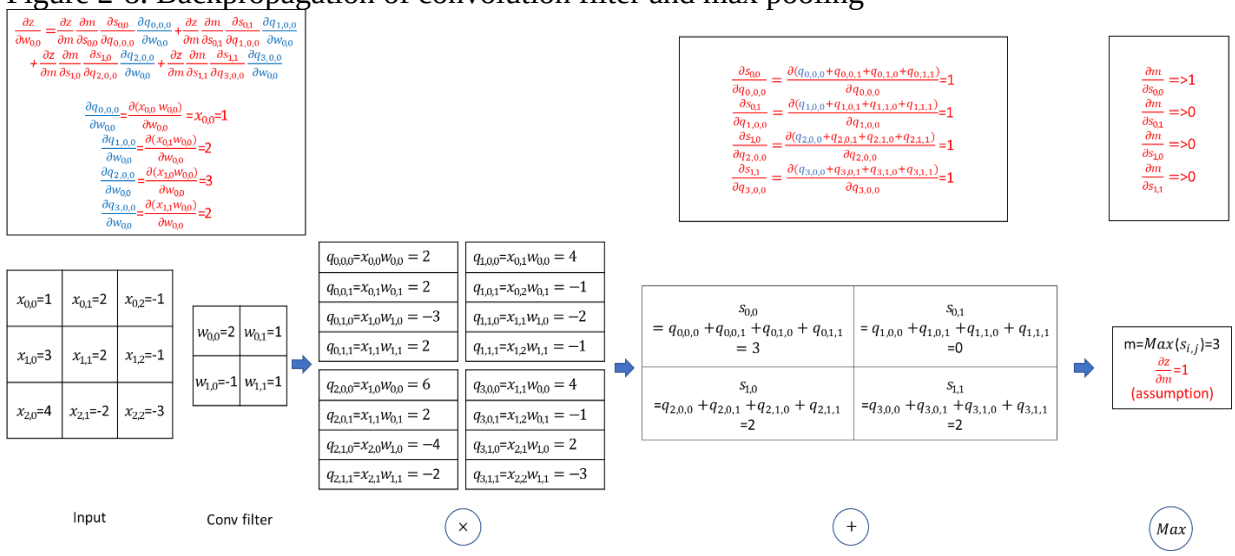
and 0.008 in the ground truth values after this one-time training process. If the number of training loops is increased, it will reduce the loss and meet the target output.

Table 2-3. The updated weights of fully connected artificial neural network from $\dot{w}_1$ to $\dot{w}_4$

$\dot{w}_1 = 0.1511$	$\dot{w}_2 = 0.2189$
$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial Outh_1}\frac{\partial Outh_1}{\partial Neth_1}\frac{\partial Neth_1}{\partial w_1}$	$\frac{\partial E_{total}}{\partial w_2} = \frac{\partial E_{total}}{\partial Outh_1}\frac{\partial Outh_1}{\partial Neth_1}\frac{\partial Neth_1}{\partial w_2}$
$\dot{w}_3$=0.2512	$\dot{w}_4$=0.3025
$\frac{\partial E_{total}}{\partial w_3} = \frac{\partial E_{total}}{\partial Outh_2}\frac{\partial Outh_2}{\partial Neth_2}\frac{\partial Neth_2}{\partial w_3}$	$\frac{\partial E_{total}}{\partial w_4} = \frac{\partial E_{total}}{\partial Outh_2}\frac{\partial Outh_2}{\partial Neth_2}\frac{\partial Neth_2}{\partial w_4}$

To apply this backpropagation concept to deep neural networks that have convolution, transposed convolution, activation function, batch normalization, etc., all these operators should be derived on the basis of partial differential equations as shown in the previous paragraphs. As an example, a backpropagation of convolution is shown in Figure 2-8.

Figure 2-8. Backpropagation of convolution filter and max pooling



The input 3 × 3 $\boldsymbol{x}$ matrix is composed of $x_{0,0}$ and $x_{2,2}$ in Figure 2-8. The weight matrix is composed of $w_{0,0}$ to $w_{1,1}$ as shown in Figure 2-8. The results of max pooling are presented as a 2 × 2 matrix consisting of $s_{0,0}$ to $s_{1,1}$. The "×," "+," and "Max" are the mathematical operations.

In this example, the gradient of $w_{0,0}$ is calculated based on the chain rule expressed as follows:

$$\frac{\partial z}{\partial w_{0,0}} = \frac{\partial z}{\partial m}\frac{\partial m}{\partial s_{0,0}}\frac{\partial s_{0,0}}{\partial q_{0,0,0}}\frac{\partial q_{0,0,0}}{\partial w_{0,0}} + \frac{\partial z}{\partial m}\frac{\partial m}{\partial s_{0,1}}\frac{\partial s_{0,1}}{\partial q_{1,0,0}}\frac{\partial q_{1,0,0}}{\partial w_{0,0}} + \frac{\partial z}{\partial m}\frac{\partial m}{\partial s_{1,0}}\frac{\partial s_{1,0}}{\partial q_{2,0,0}}\frac{\partial q_{2,0,0}}{\partial w_{0,0}} + \frac{\partial z}{\partial m}\frac{\partial m}{\partial s_{1,1}}\frac{\partial s_{1,1}}{\partial q_{3,0,0}}\frac{\partial q_{3,0,0}}{\partial w_{0,0}}, \quad (2\text{-}40)$$

where z is the previous layer of operation, q is the result of multiplication of each convolution filter weight and input matrix component value, s is the summation of the q components, and m is the result of the "Max" pooling operation. $\frac{\partial z}{\partial m}$ is assumed to be 1 as we do not know the exact value that is backwarded from the previous layer of operation in this example. Each component of Equation (2-40) is also determined in the same way presented in Equations (2-24) to (2-28).

2.3 Chapter 2 conclusion

This chapter explained the mathematical elements of the UAV control algorithm and the damage detection algorithm. Although the UAV control algorithm and the CNN algorithm are being continuously improved and researchers frequently suggest new operations, most of these algorithms follow the math in this chapter. The application of these mathematical operations is demonstrated in Chapters 3, 4, and 5.

Chapter 3. Autonomous UAVs for SHM using deep learning and an ultrasonic beacon system with geo-tagging

Chapter 3 has been reprinted from Kang, D. and Cha, Y.J., 2018. *Autonomous UAVs for structural health monitoring using deep learning and an ultrasonic beacon system with geo-tagging*. Computer-Aided Civil and Infrastructure Engineering, 33(10), pp.885-902. Impact factor: 11.775.

Abstract: Visual inspection has traditionally been used for structural health monitoring. However, assessments conducted by trained inspectors or using contact sensors on structures for monitoring are costly and inefficient because of the number of inspectors and sensors required. To date, data acquisition using unmanned aerial vehicles (UAVs) equipped with cameras has become popular, but UAVs require skilled pilots or a global positioning system (GPS) for autonomous flight. Unfortunately, GPS cannot be used by a UAV for autonomous flight near some parts of certain structures (e.g., beneath a bridge), but these are the critical locations that should be inspected to monitor and maintain structural health. To address this difficulty, this article proposes an autonomous UAV method using ultrasonic beacons to replace the role of GPS, a deep convolutional neural network (CNN) for damage detection, and a geo-tagging method for the localization of damage. Concrete cracks, as an example of structural damage, were successfully detected with 97.7% specificity and 91.9% sensitivity, by processing video data collected from an autonomous UAV.

3.1 Introduction

Traditional structural health monitoring (SHM) approaches usually require a dense array of contact sensors to measure vibrations of the structures or human inspector assessments. This makes it expensive to install and maintain a monitoring system. To overcome these issues, many computer vision-based noncontact sensing techniques have been developed (Abdel-Qader et al., 2003; Lee and Shinozuka, 2006; Chen et al., 2015; Cha et al., 2016).

Recently, Cha et al. (2017a) proposed a deep learning-based crack damage detection method with automatic feature extraction and the ability to learn damage-sensitive features with robustness to various types of environmental noise. The convolutional neural network (CNN)-based method effectively extracts and learns damage-sensitive features from input image data. Unlike a standard artificial neural network, it does not require definition of specific damage-sensitive features. Interest in the SHM discipline has been increasing in the application of the powerful CNN approach (Soukup and Huber- Mörk, 2014; Lin et al., 2017; Rafiei et al., 2017). And other recent engineering applications of deep learning have been researched for SHM (Koziarski and Cyganek, 2017; Ortega-Zamorano et al., 2017; Rafiei and Adeli, 2017). Moreover, the faster region-based CNN (Faster R-CNN) method (Ren et al., 2015) has been applied to the detection and localization of multiple damage types for a steel girder bridge (Cha et al., 2017b).

To maximize the use of computer vision sensors, unmanned aerial vehicles (UAVs) have been applied to SHM problems (Metni and Hamel, 2007; Chen et al., 2011; Eschmann et al., 2012; Zhang and Elaksher, 2012; Hallermann and Morgenthal, 2013; Sankarasrinivasan et al., 2015; Gillins et al., 2016). UAVs offer costefficient risk reduction even if the location monitored is isolated or hazardous (Metni and Hamel, 2007). They also provide a time-saving solution (Gillins et al., 2016) for data acquisition (Eschmann et al., 2012; Hallermann and Morgenthal, 2013; Sankarasrinivasan et al., 2015). However, skilled pilots are typically required to run UAVs on-site, even though some techniques have been developed for remote control (Eschmann et al., 2012; Gillins et al., 2016). Autonomous navigation methods of UAVs have been studied to address this drawback.

To the best of our knowledge, there is no clear definition or established concept and theory for the levels of autonomous UAV navigation in the robotics discipline. However, six levels of autonomous vehicle navigation have been defined by the National Highway Traffic Safety

Administration (NHTSA and SAE International, 2014), which is a part of the U.S. Department of Transportation. The six levels of autonomous vehicle navigation are also applicable to autonomous UAV navigation. Level 0 is completely manual control of navigation by pilots. Level 1 is UAV navigation performed by pilots but with some automation applied to specific flight modes, such as holding altitude and hovering. In Level 2 automation, users can define multiple flight modes for automation, and the UAV then navigates based on the scheduled flight modes if there is no unexpected change in the flying environment. In Level 3, a UAV understands changing flying environments and controls flight modes itself to navigate the new environments. In Level 4 navigation, a UAV can adaptively react when there is any system anomaly or a sudden accident, such as a collision with other objects. In Level 5, a UAV can autonomously navigate in all environments and situations. In this article, we focus on realizing Level 2 automation of UAV navigation for SHM in GPS-denied areas or complex geometric navigation environments.

Mapping and localization are critical to realization of Level 2 autonomous navigation. Planning and control of UAV navigation can be accomplished using an existing commercial mission planner and flight controller (i.e., Pixhawk). Multiple types of sensors are available for a UAV to determine its position. These sensors provide vehicle position data for the UAV to conduct its scheduled mission. For example, GPS is the most popular option for position sensors, as GPS sensors are cheaper and easier to use than other types of position sensors. A simple autonomous outdoor navigation by waypoints has been demonstrated using the GPS for localization of the UAV (Carvalho et al., 2017). However, there are several reasons why other localization sensor systems are required. First, the usage of a UAV is often limited to outdoor environments because a GPS signal is not reliable in certain locations like beneath bridge decks. Second, dynamic and complex topographic environments of the navigation space for SHM require higher accuracy in UAV localization than commercial GPS can provide. For example, dynamic water levels under bridge systems and complex indoor geometries of civil infrastructures require localization accuracy exceeding that of GPS systems.

To overcome these problems, a distance sensor and optical flow (Honegger et al., 2013) can be used with the simultaneous localization and mapping (SLAM) technique. However, the performance of SLAM and optical flow are dependent on the environment. For example, if the vision sensors cannot obtain features adequate to identify a UAV’s location, they incur a high

computational cost and accumulate localization errors (Hess et al., 2016). The real-time kinematic global positioning system (RTK GPS) was developed to address this issue (Stempfhuber and Buchholz, 2011). RTK GPS is highly accurate, but it is still not available in GPS-denied environments. As another approach, motion capture-based localization provides inspiration to extensive research in aerial robotics, allowing for complex and high-precision navigation that does not require any satellite signal (Orsag et al., 2013). However, its implementation requires a complex and expensive motion capture system (Deutscher et al., 2000).

An ultra-wideband beacon system was developed to provide high precision positioning to enable a new range of applications in GPS-denied environments (Vossiek et al., 2003; Zwirello et al., 2012; Sung et al., 2016). However, some experiments have shown millimeter-level accuracy of ultra-wideband beacon positioning systems, but the direct application is not practical in UAV systems due to high cost and a lack of integration (Zhang et al., 2006). However, an ultrasonic beacon system (UBS) can be an alternative for a practical mapping and localization system using low-cost hardware. The UBS offers a similar concept and mechanism to the ultra-wideband beacon, but it is cheaper and easier to integrate into UAVs than the ultra-wideband beacon. The UBS provides centimeter-level accuracy with proper parameter tuning (Díaz et al., 2017). For this reason, we chose UBS as the local mapping and localization sensor for Level 2 autonomous navigation of a UAV equipped with a camera for structural damage detection.

In the present study, we propose an autonomous UAV based SHM method using UBS. We used CNN with a sliding window technique (Cha et al., 2017a) as an example damage detection method. The detected damage is localized by geo-tagging method. Section 2 describes the UBS-based autonomous UAV system and CNN layers that we used for concrete crack detection. In Section 3, experimental tools, including UAV fabrication and UBS are described and test scenarios and the results of UAV autonomous navigation experiments are discussed. Section 4 provides conclusions, discusses study limitations, and suggests future improvements.

3.2 Methodology

To develop autonomous navigation for a UAV, UBS was used for local mapping and localization positioning sensors, a ground station including a mission planner was used to assign a navigation

plan, and two UAVs were fabricated and equipped with a flight controller and an action camera. Figure 3-1 depicts the architecture of the autonomous UAV using a Pixhawk 2.1 flight controller. A commodity computer can serve as a ground station, where the role of mission planner is to assign a navigation plan and monitor the UAV. The first system used Pixhawk 2.1, which is commercial flight controller hardware (Proficnc, 2017). The Pixhawk 2.1 has Ardupilot 3.5 (Ardupilot, 2017), which is open source, installed as firmware. Ardupilot 3.5 has a feedback proportional–integral–derivative (PID) controller (Minorsky, 1922), (Lim et al., 2012) that was used for this study as a default controller to control the speed of motors through the electronic speed controllers (ESCs). The position data from a mobile beacon was estimated using an extended Kalman filter 3 (EKF3) algorithm as input to the PID control system. The UBS has multiple mobile and stationary beacons. The EKF3 is an updated version of the original EKF (Smith et al., 1962) for flight control. Image geo-tagging was conducted based on the runtime history of the video footage collected from the action camera and the UAV running time. A CNN with a sliding window technique (Cha et al., 2017) was used for crack detection in concrete as an example of a type of structural damage. A geographic information system (GIS) was also used to collect all video data of damage detected in the structure, along with location information. The details of autonomous flight, using UBS and CNN, are explained in the following sections.

Figure 3-1. Overall architecture of an autonomous UAV using a Pixhawk 2.1 flight controller (Kang and Cha, 2018)

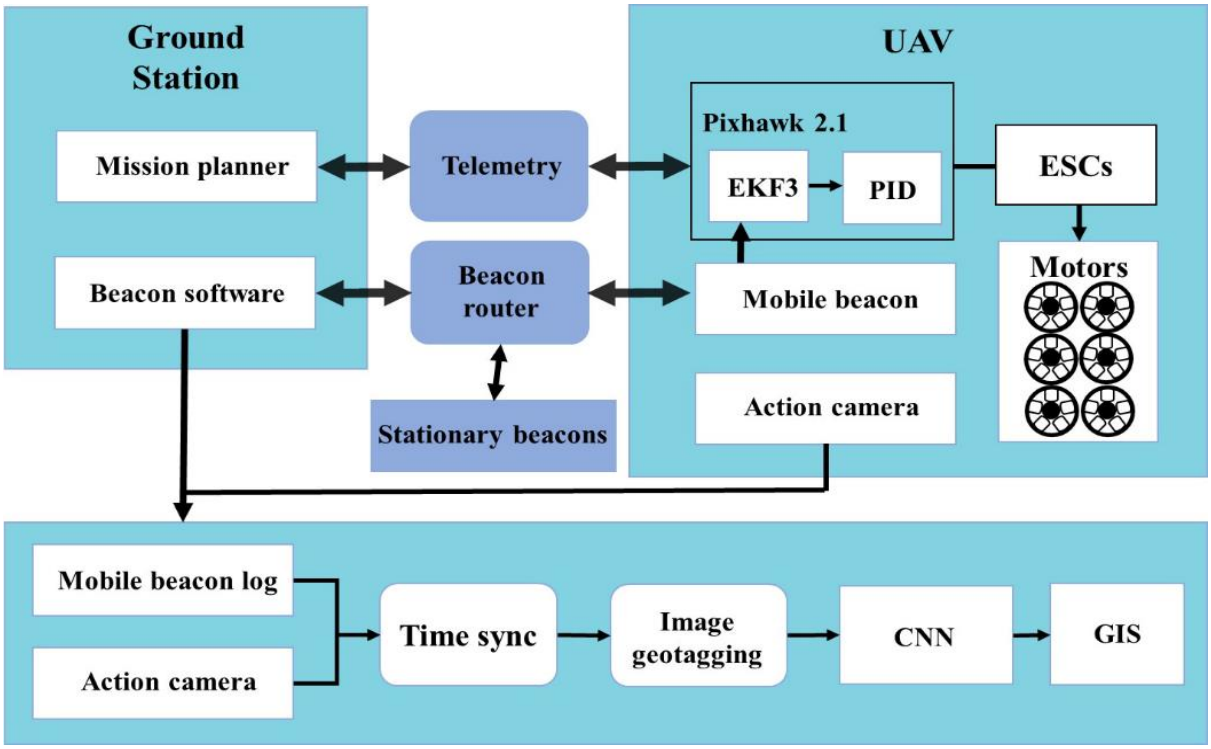
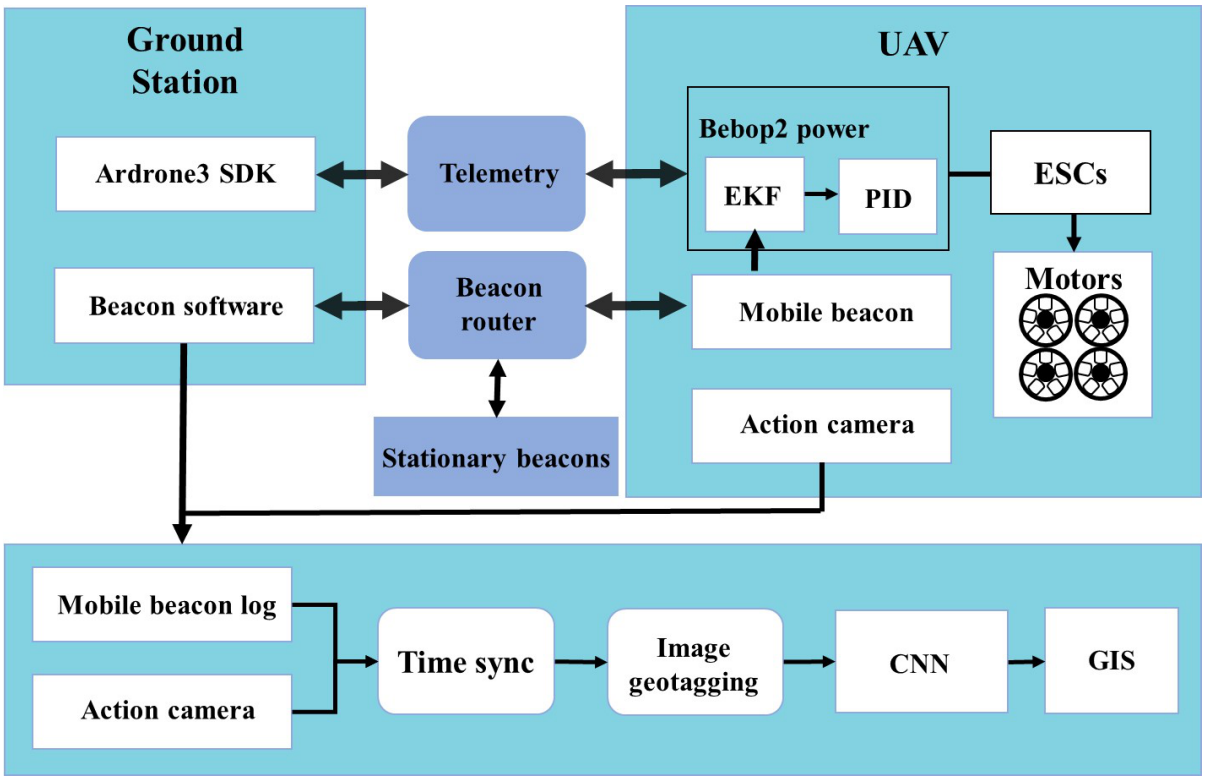


Figure 3-2 depicts the architecture of an autonomous UAV using the commercially available drone, Parrot Bebop2 Power (Bebop2). The only difference between this drone and the first UAV is that the commercial Bebop2 UAV has an imbedded controller. It was adopted because it is capable of more complex autonomous navigation than the previous UAV, and the experimental space at the University of Manitoba was limited. The previously described Pixhawk UAV is larger than the Bebop2; therefore, only limited missions were possible with this vehicle.

Figure 3-2. Overall architecture of autonomous commercial Bebob2 UAV (Kang and Cha, 2018)



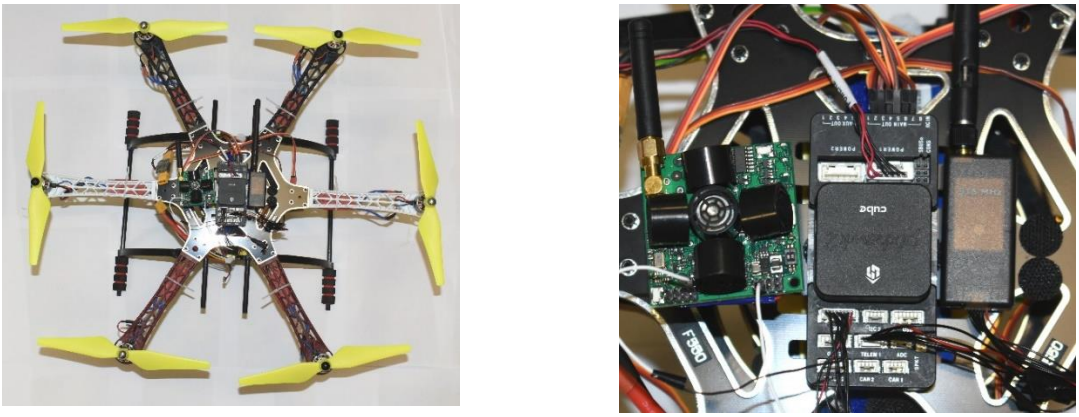
3.2.1 Programmable UAV fabrication and mapping system

As the first step for Level 2 autonomous navigation of Pixhawk UAV, a programmable UAV was fabricated to enable modification of the flight controller source code. The fabricated programmable Pixhawk UAV shown in Figure 2 includes various hardware components: legs and motors, telemetry, electronic speed controllers (ESCs), a flight controller, a mobile beacon, an action camera with vibration damper, and batteries. A detailed view of the components is presented in Figure 3-3.

We modified the DJI F550 frame of a multi-rotor Erle copter by installing six propeller motors as the UAV frame. It can carry a 2kg payload for 15 minutes of flight time. The six motors are brushless motors that require 30A ESCs. For telemetry, a 3DR 915 MHz radio is used to communicate between the ground station and UAV. The Pixhawk 2.1 was selected as a flight controller due to its improved inertial measurement unit (IMU) (Meier et al., 2011). The Ardupilot 3.5 open-source code was installed in the Pixhawk. Cube design was applied to the Pixhawk 2.1 to reduce vibration (Proficnc, 2017).

To monitor a region of interest (ROI), a Sony FDR-X3000 action camera was installed in the UAV as shown in Figure 3 (c). Its 60 frames per second (FPS) recording capability ensures stable video even if the camera is vibrated by the UAV motors. The camera supports 4K image resolution, but the 1080pixel resolution was selected due to shutter speed. Sony's Playmemories application allows the user to remotely check and change the camera setting. The lightweight camera (114 g with a battery) is ideal for a UAV application. Anti-vibration foam that serves as a gimbal was installed to reduce the jello-effect.

Figure 3-3. Components of the fabricated Pixhawk UAV (Kang and Cha, 2018)



(a) Fabricated UAV system

(b) Pixhawk 2.1

(c) Sony action camera

(d) Mobile beacon

(e) Telemetry

As the second UAV used in this study, the Bebop2 has its own camera, but the camera angle was modified to a right angle to the ground to collect clear images of the concrete surface and detect cracks, as shown in Figure 4. It uses a removable battery (3,350 mAh), which allows the UAV to fly up to 30 min. A mobile beacon was installed on top of the Bebop 2 UAV, as

depicted in Figure 3-4. The Bebop2 is programmable, and the manufacturer also offers a software development kit.

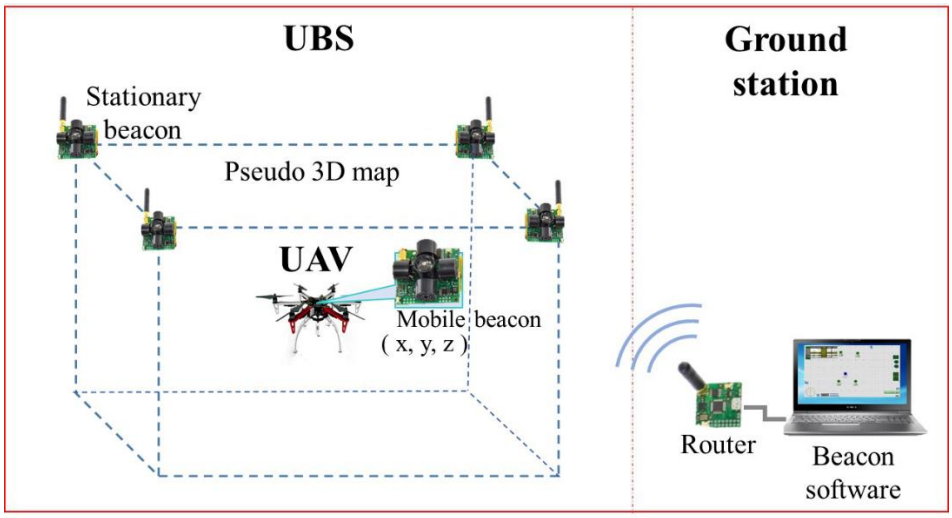
Figure 3-4. Components of the Bebop2 Power (Kang and Cha, 2018)

(a) Bebop2 Power with a mobile beacon

(b) Beacon router

A mobile beacon was installed in the UAV frame to provide 3-dimensional (3-D) position data (x, y, z) of the two UAVs. The Marvelmind Robotics UBS is composed of a mobile beacon, multiple stationary beacons, a router, and the Dashboard beacon software, as shown in Figure 3-5. The UBS generates a local pseudo-3-D map, which is depicted by the blue dotted line in Figure 3-5. The ROI should be located within this pseudo-3-D map. The stationary beacons define the border lines of the map and can be installed on the wall or with a tripod.

Figure 3-5. Relationship between stationary beacons and mobile beacon (Kang and Cha, 2018)



Each beacon has five transceivers. The stationary beacons are similar to a GPS, sending ultrasonic signals and calculating distances to the mobile beacon installed in the UAV through a router. The position of the mobile beacon can be calculated using Equation (3-1) below:

$$p(t) = \sqrt{(x_s - x_m)^2 + (y_s - y_m)^2 + (z_s - z_m)^2} \quad (3\text{-}1)$$

where (x_s, y_s, z_s) represents the stationary beacon, and (x_m, y_m, z_m) represents the mobile beacon's coordinates. The Marvelmind UBS supports the GPS format of the National Marine Electronics Association (NMEA). The mobile beacon should always be within the pseudo-3-D beacon map for navigation to work properly. In this study, the mobile beacon was installed on top of the UAV to avoid blocking the beacon signal. We used the programmable Ardupilot 3.5 open source code in the Pixhawk 2.1 flight controller to program the autonomous navigation of the first Pixhawk UAV based on the pseudo-3-D beacon map. The modified source code was developed in C++. The open source code was injected through the mission planner in the ground station (Carvalho et al., 2017). For the Bebop2 UAV, the firmware provided by the manufacturer was used.

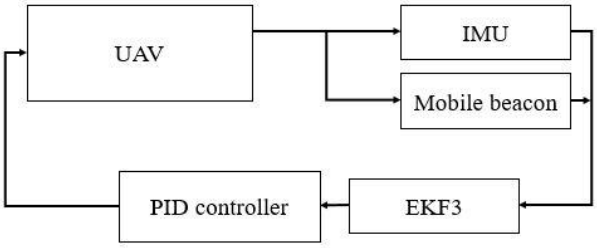
3.2.2 Ground station

A Samsung nt500r5h laptop, a commodity commercial computer, was used as the main ground station computer. It has a 2.2 GHz computer processing unit (CPU) and 8 GB memory. For the Pixhawk UAV, Mission Planner and beacon software were installed at the ground station. The Bebop2 did not require a Mission Planner software but used the same ground station computer. The Mission Planner is open source software that provides a graphical user interface (GUI) to manage and monitor the navigation of the UAV. The Mission Planner has many roles. It displays the status of a UAV through the MAVLink protocol (Meier et al., 2011). The Mission Planner can record and replay the log data of a UAV flight. Error messages associated with the status and navigation plan, including environmental noise such as magnetic interference, can be reviewed by a user.

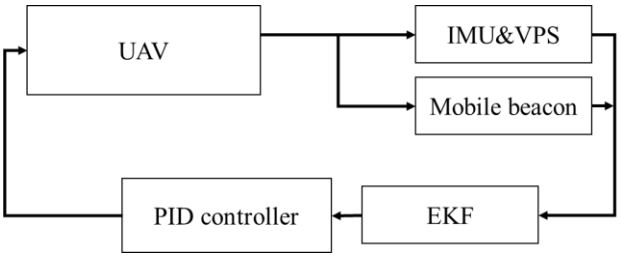
3.2.3 Flight controller

The role of the flight controller is critical in an autonomous UAV. In the present study, For Pixhawk UAV, Pixhawk 2.1 with an embedded Ardupilot 3.5 source code was used for control of the autonomous UAV. The Ardupilot 3.5 uses a PID controller with an EKF3 algorithm to remove noise and enhance the estimation of UAV position measurement data from UBS, as shown in Figure 3-6 (a). A similar controller to that of the Pixhawk UAV was used for the Bebop2, but it had a vision positioning system (VPS) in the control system, as shown in Figure 3-6 (b).

Figure 3-6. Flight control systems of the two UAVs (Kang and Cha, 2018)



(a) Pixhawk 2.1 UAV



(b) Bebop2 UAV

The Pixhawk 2.1 has an embedded inertial measurement unit (IMU) composed of three accelerometers, three compasses based on gyroscopes and magnetometers, and two barometers. For autonomous flight of the UAV in GPS-denied locations, information from the three accelerometers and the compass are used as IMU data. The barometer information was not used for indoor flight because the measured data is not accurate due to the nature of indoor operations. The PID controller normally uses GPS input for outdoor flight when GPS is reliable. However, the mobile beacon data replaced the GPS input data in this study to support operations in indoors or GPS-denied areas. The EKF3 algorithm predicts the current location of the UAV based on the

mobile beacon position data and IMU data. These parameters can be defined by the Mission Planner using MAVLink.

For the Bebop2 UAV, the same beacon system was used but without a mission planner. In order to develop autonomous navigation for the Bebop2 UAV, vision positioning system, altimeter, ultrasound, and beacon data were integrated into the Bebop2's existing controller using PID and EKF, as shown in Figure 3-6 (b). The inertial navigation system of the Bebop2 consisted of a three-axis gyroscope, an accelerometer, and a magnetometer.

3.2.4 Beacon-based geo-tagging

To track the location of the UAVs and localize the damage detected during the deep CNN process, a geo-tagging method is used in this study. From these autonomous UAV systems, video data and GPS coordinates collected from UBS are sent to the base station. Time synchronization is an important aspect of geo-tagging. The time steps of the video and beacon systems are synchronized based on the servo information of the UAVs, which provides the start and finish times of the UAV navigations. For convenience, we have extracted image data by seconds. Since UBS is not an actual GPS system, we need to give GPS coordinates as a starting point for UBS settings. The UBS system uses degrees based on latitude (north) and longitude (west). Therefore, the beacon coordinates x and y can be converted following Equation (3-2) (ArduPilot 2017).

$$\begin{aligned} Lat &= Lat0 - x \cdot \cos\big(radian(Rot)\big) \cdot LatPa + y \cdot \sin(radian(Rot)) \cdot LatPa \\ Lot &= Lon0 - x \cdot \sin\big(radian(Rot)\big) \cdot LonPa + y \cdot cos(radian(Rot)) \cdot LonPa \end{aligned} \tag{3-2}$$

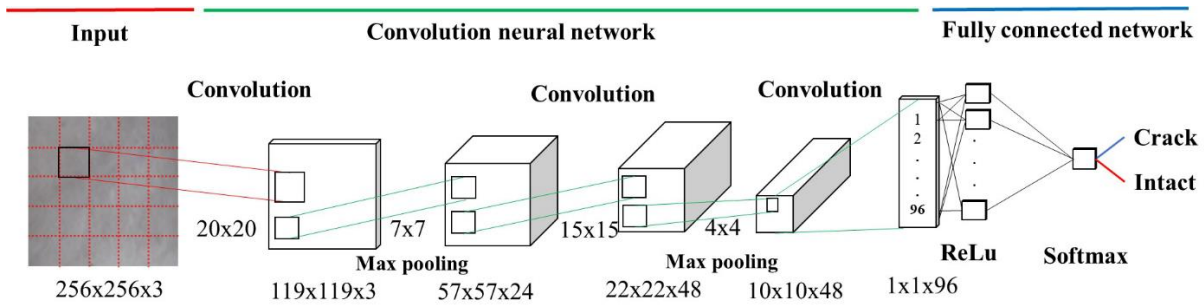
where *Lat0* and *Lon0* are the first beacon's latitude and longitude, respectively. *Rot* is the angle difference between the north of the real map and the north of the virtual map. *LatPa* and *LonPa* are hyperparameters (9.010063270126722e-06, 1.130896616413607e-05).

3.3 Crack detection using deep convolutional neural network

To detect structural damage, a deep convolutional neural network (CNN) with the sliding window technique (Cha et al., 2017) was used to analyze video data collected from the UAV. The sliding window technique uses a predefined size of window to localize the detected damage (Cha et al.,

2017). In the present study, an existing CNN architecture was used. To train the CNN classifier, we used a training data set of raw images of concrete surfaces with a broad range of image variations, including spot lighting and shadows. A Sony FDR-X3000 camera, shown in Figure 3(c), was used for the test dataset because the payload of UAV is small and cannot carry a digital single lens reflex camera. The prepared training image set fed into a CNN to form a CNN classifier to classify intact and cracked concrete areas. The CNN used in this study, shown in Figure 3-7, was composed of input, convolution, pooling, activation, and output layers. Auxiliary layers, such as dropout and batch normalization layers, were also used. The details of the CNN are presented by Cha et al. (2017).

Figure 3-7. CNN architecture (Kang and Cha, 2018)



3.3.1 Convolution layer

A convolution layer performs a dot product between a subarray of an input array and a filter. The initial and bias weight values of the filter are randomly generated. Both values are tuned by training that uses a stochastic gradient descent algorithm. The multiplied values are summed, and bias is added to these values. An additional hyperparameter of the layer is the pixel stride, which indicates how many columns and rows (pixels) slide at a time across the input array's width and height.

3.3.2 Pooling layer

The CNN's pooling layer, which reduces the spatial size of an input array to reduce the computation costs, is also important. Max pooling takes the maximum value from the subarray of the input array, whereas mean pooling takes the mean value. In this thesis, all pooling layers are

used as max pooling because the performance of the max pooling operation is better than that of the other operation (Scherer et al., 2010).

3.3.3 Auxiliary layers

Dropout layers were used to solve the overfitting problem (Srivastava et al., 2014). Training a network with large numbers of neurons often results in overfitting due to the complex connections. The main idea of the dropout technique is to randomly disconnect the connections between neurons of connected layers following the dropout rate. Batch normalization is also used after the first, third, and fifth layers. The batch normalization algorithm is a technique to improve the performance and stability of neural networks (Ioffe and Szegedy, 2015). It normalizes the layer inputs; as a result, this technique facilitates a high-learning rate and leads to much faster network convergence.

3.3.4 Activation function

The most typical way to provide nonlinearity in a standard artificial neural network is to use sigmoidal functions. In this study, rectified linear units (ReLU) were chosen because they achieve better accuracy in the CNN (Nair and Hinton, 2010). Compared to other nonlinear functions, the ReLU has no bounded outputs except for its negative input values. Equation (3-3) represents the ReLU.

$$ReLU = \begin{cases} (x < 0)\ f\ (x) = 0 \\ (x > 0)\ f\ (x) = 0 \end{cases} \tag{3-3}$$

3.3.5 Softmax layer

To classify final output data, the softmax layer, represented in Equation (3-4), is the last layer of the CNN architecture. It classifies input data as either intact or cracked. For the input data from the last layer through the softmax layer, the range of values is 0 to 1.

$$\sigma(z)j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}} \; for \; i = 1, \dots, K \tag{3-4}$$

where K is the number of categories.

3.3.6 Training process

As the initial values of the filter weights in each layer are randomly assigned, the predicted and actual classes do not usually coincide. To calculate the level of deviation between predicted and actual classes, a softmax loss function was used. Logarithmically decreasing learning rates were used to update the gradient descent. As mentioned earlier, we used the pretrained network developed by Cha et al. (2017) to obtain the advantage of its high damage detection accuracy (98%). For training and validation, the pretrained network used 40,000 images of cracked and intact concrete with 256 × 256 pixel resolution, taken at the Engineering and Information Technology Complex at the University of Manitoba. However, these images were not taken in the same classrooms (E2-229, E2-399) where the autonomous UAV tests were conducted for this thesis. Further details of the CNN are available in Cha et al. (2017).

3.4. Case studies

To validate the performance of the proposed method, the structural damage detection method using autonomous UAVs with UBS and deep learning was applied to a concrete crack detection problem in the classrooms at the University of Manitoba. The method, explained in Section 2 can be extended to use beneath bridge systems or other indoor environments using the experimental procedures described in this section.

3.4.1 Experiment setup

Due to the nature of a complex, autonomous UAV systems, a significant amount of hardware and software had to be designed, configured, and/or integrated. The fundamental procedures for experimental settings were:

Step 1: Preparation of ground station, including installing Mission Planner and beacon software

Step 2: Installation of flight controller firmware in the UAV

Step 3: UAV sensor calibrations

Step 4: Modification of Ardupilot 3.5 reinstallation in the UAV

Step 5: Installation of beacon software and physical installation of mobile beacon in the UAV and stationary beacons on-site, including beacon router to generate a pseudo-3-D beacon map through ground station

Step 6-1: Activation of the Pixhwak UAV, flight path planning using the mission planner (for the Pixhawk UAV), and entry of the mission into the flight controller through MAVLink

Step 6-2: Activation of the Bebop2 UAV, flight path planning, and entry of the mission into the flight controller through MAVLink

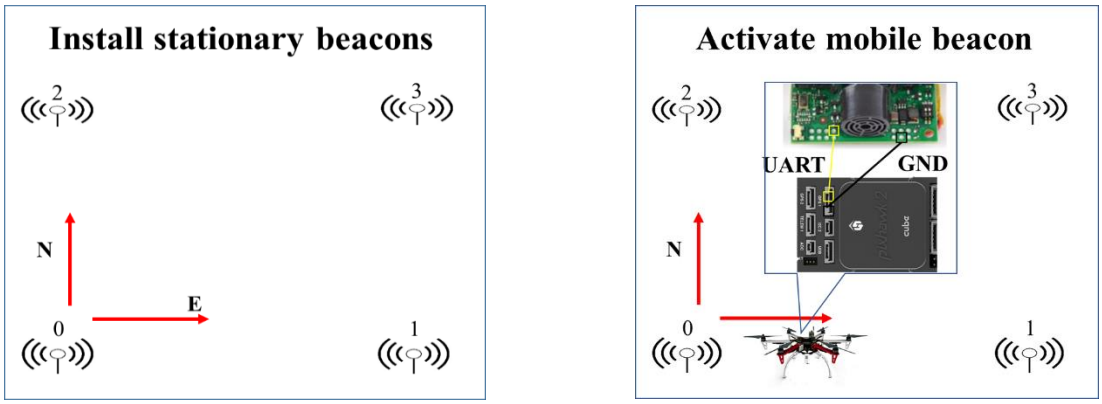
Step 7: Rebooting the UAV, including camera

Step 8: Commanding of the start of autonomous flight through the Mission Planner at the ground station

Since the first three Steps and Steps 7 to 8 in the above procedures are straightforward and general steps for any UAV flight using the ground station, this section focuses on explaining Steps from 4 to 6 as the key steps for UBS-based autonomous flight for this study. In Step 4, hyper-parameters in the flight controllers were redefined from the default setting through the Mission Planner. For example, to activate the EKF3 function, the parameter "Ahrs_EKF_TYPE" was defined as "3" instead of "2," which represents the EKF2 function. EKF3 has many hyperparameters, as shown in Table A1. These parameters should be determined via trial and errors based on limitations of the flight controller (Meng et al., 2010). The Step 4 is unnecessary for Bebop2.

In Step 5, UBS firmware was installed using the Dashboard beacon software through the ground station to calibrate the UBS. The calibrated UBS can generate a pseudo-3-D beacon map based on stationary beacon locations. For this study, four stationary beacons (0-3) were used, as shown in Figure 3-8(a). To generate the pseudo-map, the North and East directions must be defined. Default compass 1 in the Ardupilot 3.5 was nullified by deactivating "Tbn_zero_Yaw" from Euler function in the EKF3. The line between beacons 0 and 1 defines east; based on this east direction, the line between beacons 0 and 2 is automatically determined to be north. The east direction should be set first for the Marvelmind beacon system.

Figure 3-8. Set North in the map (Kang and Cha, 2018)



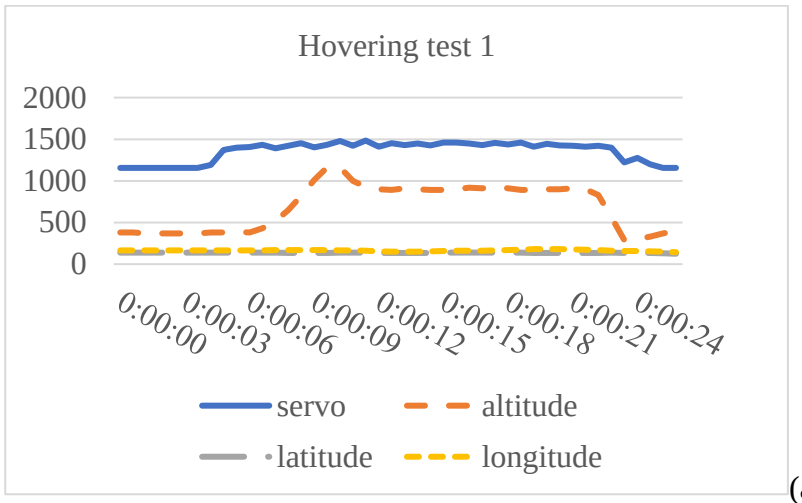
(a) (b)

A data rate of 500 kbps was used in the radio profile of all the beacons for improved position data transmission. The mobile beacon was physically installed in the UAV, and the universal asynchronous receiver/transmitter (UART) and ground (GND) ports of the mobile beacon were connected to the GPS port of the Pixhawk 2.1, as shown in Figure 3-8(b). Next, mobile beacon parameters (i.e., $GPRMC, $GPGGA and $GPVTG) were activated through the Dashboard. Lastly, the UBS router was connected to the ground station.

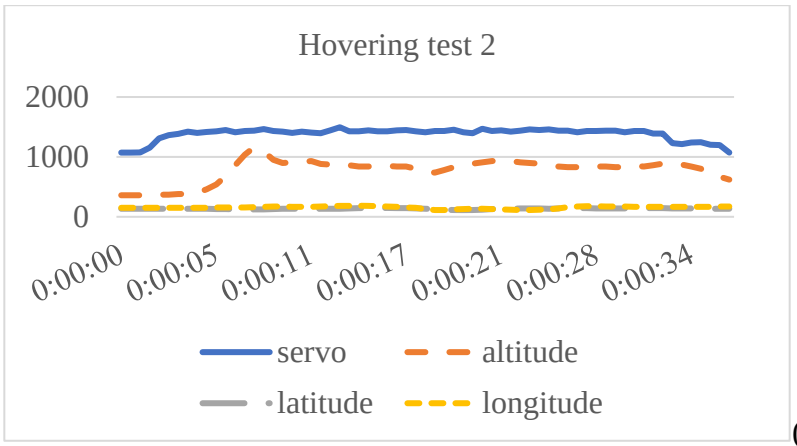
3.4.2 Experimental tests using Pixhawk UAV

Due to flight restrictions by Transport Canada, field testing of the proposed approach was not possible. As an alternative, two different experiments are conducted in Room E2-399 of the engineering building at the University of Manitoba. There is a concrete crack on the floor visible to the human eye, which was good for validation of the proposed method. The first experiment consisted of a hovering test to validate autonomous navigation of the UAV. In a hovering test, the UAV stays in a virtual circle and attempts to stabilize its position in the area of the circle. The Mission Planner in the ground station was used to command a hovering mode for the UAV. In this test, a 50cm diameter circle was set as the hovering area based on practices in the UAV discipline (Teuliere et al., 2015; Carvalho et al., 2017).

Figure 3-9. Hovering tests (Kang and Cha, 2018)



(a)



(b)

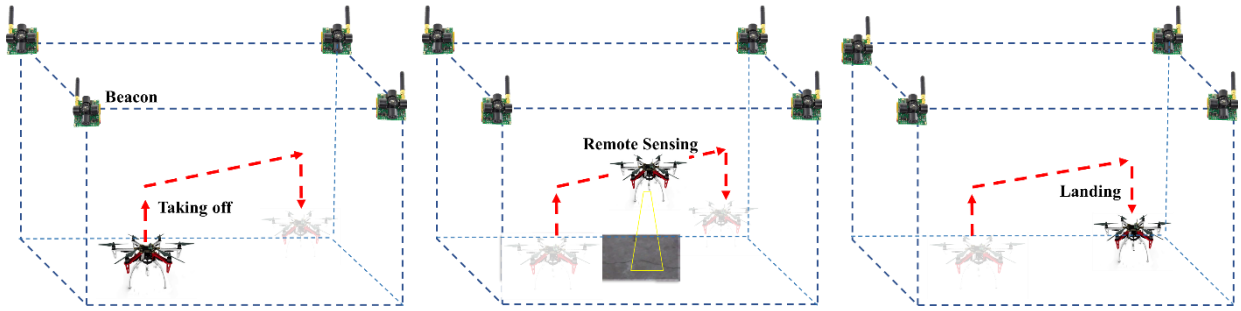
The flight parameters and results of two hovering tests are presented in Table 3-1 and Figure 3-9. The UAV flight was stable, with small errors in latitude, longitude, and altitude. The durations of Tests 1 and 2 were 20 sec. and 35 sec., respectively. The latitude difference did not exceed 10 cm. The maximum longitude error was 15 cm. These errors are quite acceptable in the UAV discipline, passing our 50cm tolerance (Teuliere et al., 2015; Carvalho et al., 2017). In Figure 3-9, the three parameters represent altitude, latitude, and longitude, respectively, measured in mm. Servo, shown on the right-hand side, is a dimensionless variable for pulse width modulation (PWM) control. An ESC determines the proper power to give the motor based on servo which is related to PWM. The overall start and end points of the flight can be monitored based on this line. Based on these hovering tests, the overall setting and performance of the UAV are validated for the practical and complex mission of structural damage detection.

Table 3-1. Experiment 1 hovering test results

	Latitude (y)	Longitude (x)	Altitude (z mm)	Latitude difference	Longitude difference
Test 1	1.37E-05	1.7E-05	1000	0.0000002 (2.0 cm)	0.000001 (10.0 cm)
Error	1.39E-05	1.8E-05	900		
Test 2	1.4E-05	1.7E-05	1000	0.0000005 (5.0 cm)	0.0000015 (15.0 cm)
Error	1.4E-05	1.9E-05	863		

For structural damage detection using this autonomous UAV system, a mission based flight was assigned through the mission planner in the ground station. This mission required the UAV to take off within 1 m, fly straight North for 1.9 m, and land with video data collected with the action camera installed in the bottom of the UAV, as shown in Figure 3-10 and Table 3-2 The GPS coordinates can be converted from 1° to 111 km. Based on this scale, the 3×10^{-7}° creates a difference in length of approximately 3 cm. The results in Table 2 indicate that the overall mission was conducted very accurately.

Figure 3-10. Experiment scenario (Kang and Cha, 2018)



(a) Take off (b) Remote sensing (c) Landing

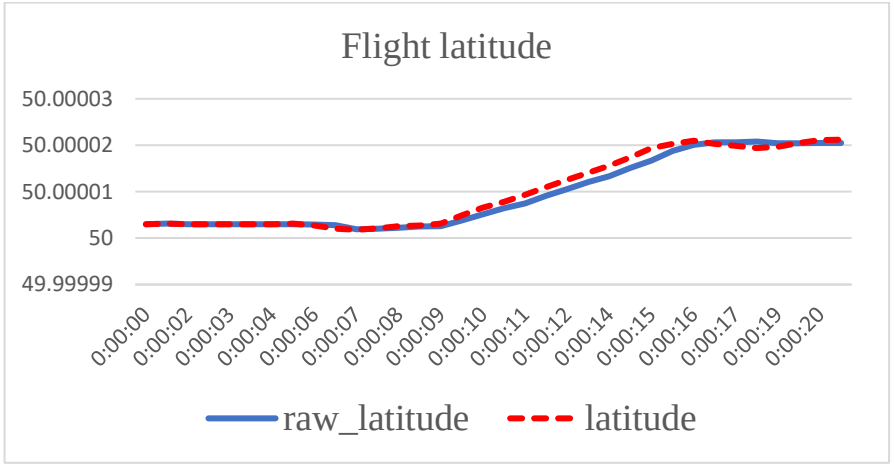
Table 3-2. Experiment 2 test results

	Latitude	Longitude	Altitude	Latitude difference	Longitude difference
Starting point	50.000003	97.0000207	0	0	0.0000003 (2.72cm)
Waypoint	50.0000173	97.0000238	1		
Arrived point	50.0000173	97.0000241	0.98		

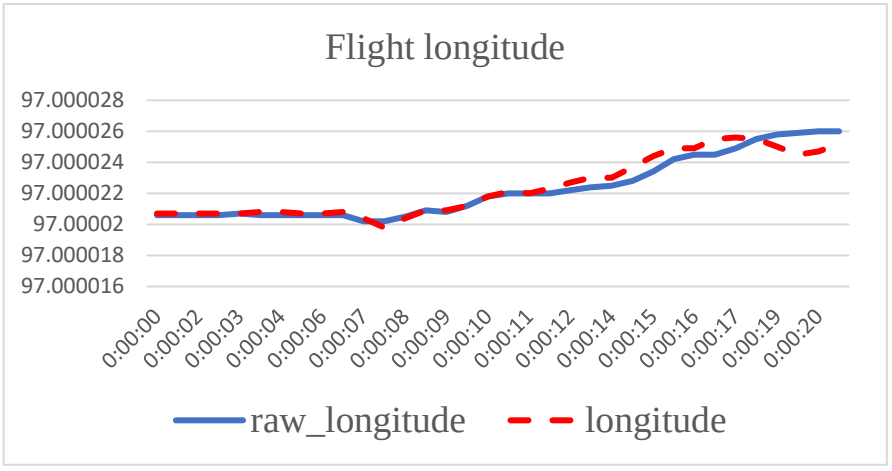
The UAV movements are plotted in Figure 3-11 to validate the overall autonomous flight mission of Test 2. Figure 11 shows the time history of UAV altitude. The raw altitude data

measured from UBS is plotted as a solid blue line and predicted altitude data calculated from EKF3 is plotted as a dotted red line. Figure 3-11 (a) shows the raw latitude time history as a red line and predicted time history from EKF3 as a green line. As described above, the unit of latitude and longitude is degrees (˚). Figure 3-11 (b) shows the time histories of longitude data. The raw and predicted time histories for latitude and longitude data matched well. Using the root mean square error, the first experiment resulted in differences of 7 cm for latitude and 3.1 cm for longitude.

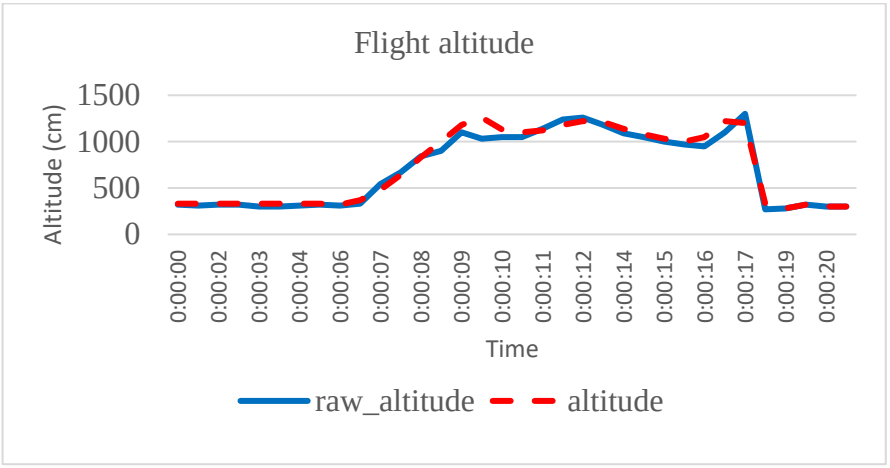
Figure 3-11. Beacon latitude and longitude altitude time history (Kang and Cha, 2018)



(a)



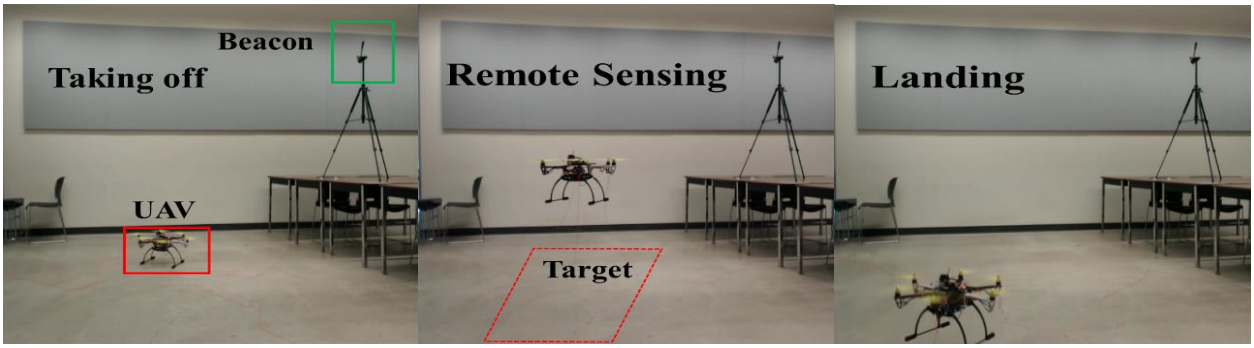
(b)



(c)

Figure 3-12 shows the actual flight of the UAV. The distance from home to the waypoint was 1.9 m, based on the latitudes and longitudes in Table 3-2. There was approximately a 3cm longitude error in the waypoint (i.e., 97.0000238-97.0000241). This result is quite accurate compared to previous research using an ultra-wide band beacon system (Zwirello et al., 2012; Tiemann et al., 2015).

Figure 3-12. Test in classroom (E2-399) (Kang and Cha, 2018)



3.4.3 Experimental tests using the Bebop2 UAV

To conduct complex missions over a large area, the Pixhawk UAV is not appropriate due to the physical size of UAV and safety issues. Therefore, the Bebop2 UAV, which is a small UAV suitable for a classroom, can be used for more complex missions. A large conference room (E2-229) with a wide variety of cracks in the floor was selected, as shown in Figure 3-13. The virtual map size is approximately 10 m × 17 m. In each corner of the virtual map, stationary beacons were

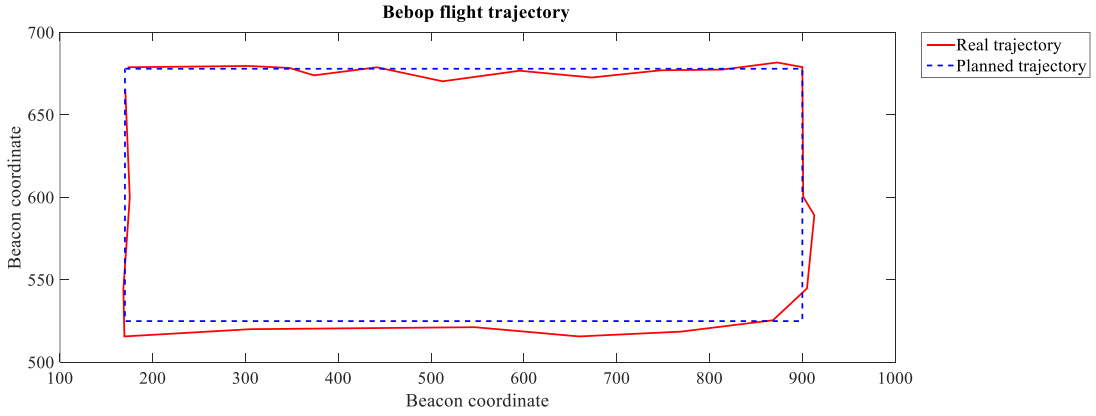
installed on the ceiling. The idea was to make a rectangular trajectory in the center of the virtual map to detect cracks in the concrete floor. The trajectory of the Bebop2 UAV is presented in Figure 3-14. Two different tests were conducted. The actual flight trajectory is presented in Table 3-3 and plotted as solid red lines in Figure 3-14. Based on the planned trajectory, the error is within 20 cm at each waypoint.

Figure 3-13. Large conference room (E2-229) for Bebop2 UAV tests (Kang and Cha, 2018)

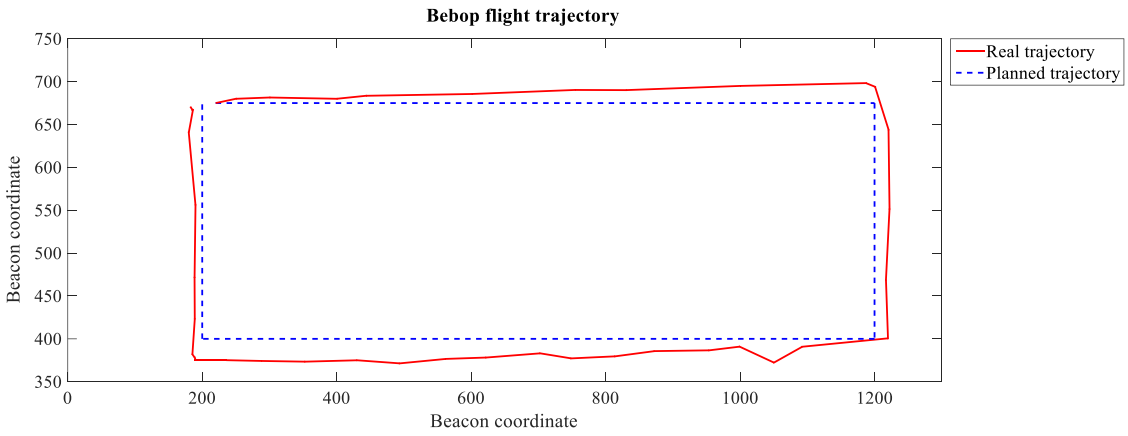
Table 3-3. Three experimental test results

Unit (cm)	starting point	waypoint1	waypoint2	waypoint3	end point
Mission point1	(170,675)	(900,675)	(900,525)	(170,525)	(170, 675)
Arrived point1	(170,675)	(900,679)	(900,535)	(169.4,515)	(171.4,665.1)
Error	(0,0)	(0,4)	(0,10)	(0.6,10)	(1.4,11.9)
Mission point2	(220,675)	(1200,675)	(1200,400)	(200,400)	(200,675)
Arrived point2	(220,675)	(1200,685)	(1217,400)	(190,375)	(183,670)
Error	(0,0)	(0,10)	(17,0)	(10,15)	(7,5)

Figure 3-14. Trajectories of Bebop2 UAV in E2-229 (Kang and Cha, 2018)



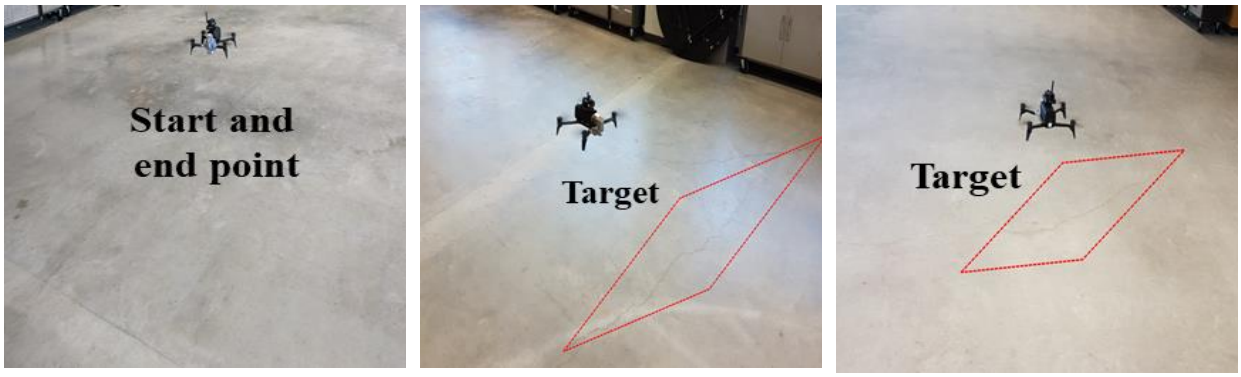
(a)



(b)

The targets (concrete cracks) and positions of the Bebop2 UAV are depicted in Figure 3-15. Figure 3-15 (a) shows the start and end waypoints. Figures 3-15 (b) and (c) show the targets of the UAV for structural health monitoring as an example study.

Figure 3-15. Test in conference room (E2-229) (Kang and Cha, 2018)



(a) (b) (c)

3.4.4 Computer vision evaluation

The sets of video data collected by the action camera and Bebop2 UAV camera during the autonomous flight of the missions described in the previous section is presented. Using the collected video data, the CNN-based concrete crack detection method (Cha et al., 2017a) was applied to detect concrete cracks. The original video data were 2304 × 1296 pixels but was resized to 2304 × 1280 pixels for input to the CNN-based detection method. The results of the CNN-based method using video data collected from the autonomous UAVs are presented in Figure 16 (a). These results from the autonomous UAV navigation are quite similar to results using the video from the manual action camera, shown in Figure 3-16 (b).

The dotted yellow box in Figure 3-16 indicates a false positive image. The dashed blue box on the image indicates a false negative. To compare the accuracy of the results, we determined the sum of true positives (TP) and true negatives (TN) and divided it by the total number of sliding windows (Tnsw) tested, as expressed in Equation (3-5). To compare different types of accuracy measurements, specificity and sensitivity are used, as shown in Equations (3-6) and (3-7). FN is the number of false negatives. FP is the false positives. Here, positive means that the surface is damaged (i.e., a concrete crack), and negative means that the surface is intact.

$$Accuracy = \frac{TP + TN}{Tnsw} \tag{3-5}$$

$$Sensitivity = \frac{TP}{Tp + FN} \tag{3-6}$$

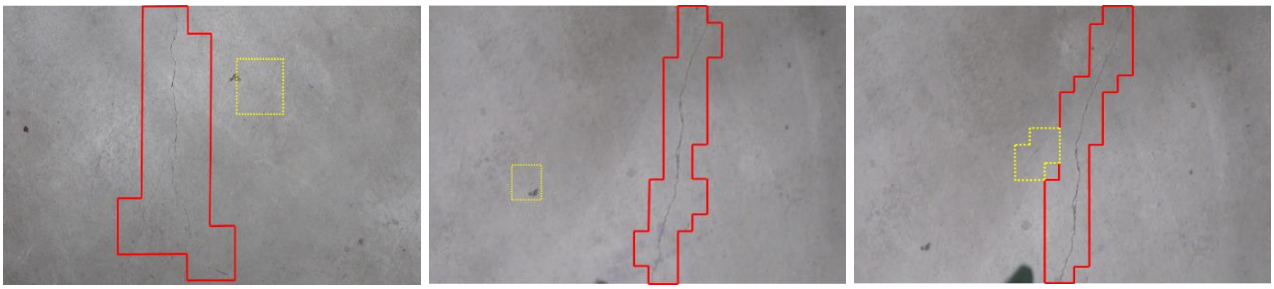
$$Specificity = \frac{TN}{TN + FP} \tag{3-7}$$

Figure 3-16. Concrete crack detection results from E2-399 (Kang and Cha, 2018)

(a) Image from autonomous Pixhawk UAV (b) Image from manual

Both images exhibited the same accuracy, 97.6%. The accuracy determined by this study agrees well with the previous results of Cha et al., (2017). For more extensive testing, we used three additional image frames from the video data collected from the autonomous Pixhawk UAV based on UBS. As shown in Figure 3-17, the concrete cracks were detected well. These results show that autonomous UAV navigation based on UBS is quite promising for structural health monitoring. The results also show that autonomous UAV-based monitoring has significant potential for future infrastructure health monitoring.

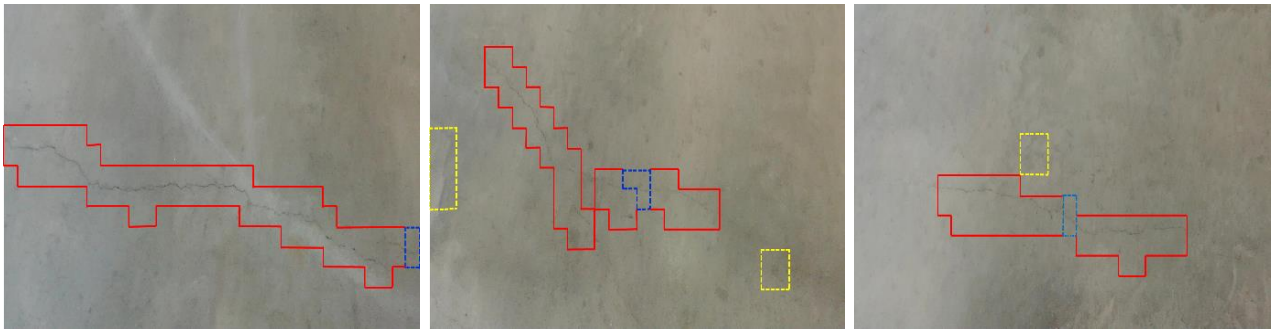
Figure 3-17. Concrete crack detection results from E2-299 (Kang and Cha, 2018)



(a) (b) (c)

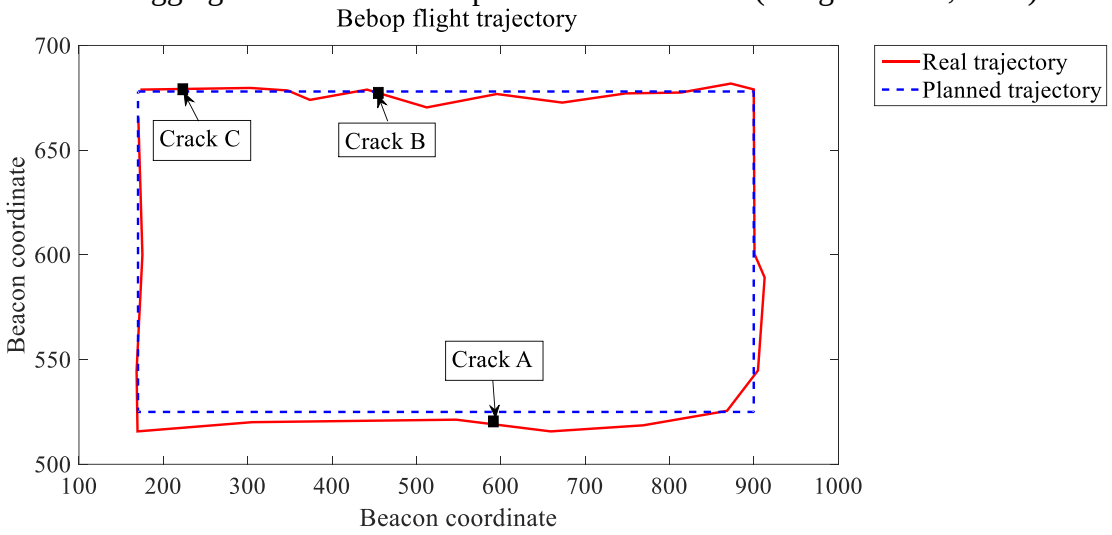
Based on the above achievements, we carried out more complex missions with longer navigation distances in the conference room (E2-229), as shown in Figure 3-14. The detected concrete cracks are presented in Figure 3-18. Based on the results obtained from the autonomous flight of the Bebop2 UAV, the accuracy was 96.6%, the sensitivity was 91.9%, and the specificity was 97.9%. All of these detected cracks were localized by the geo-tagging method described in Section 3.2.4. The localized damage information was plotted in Figure 3-19 for the two autonomous navigation cases.

Figure 3-18. Concrete crack detection result (Kang and Cha, 2018)

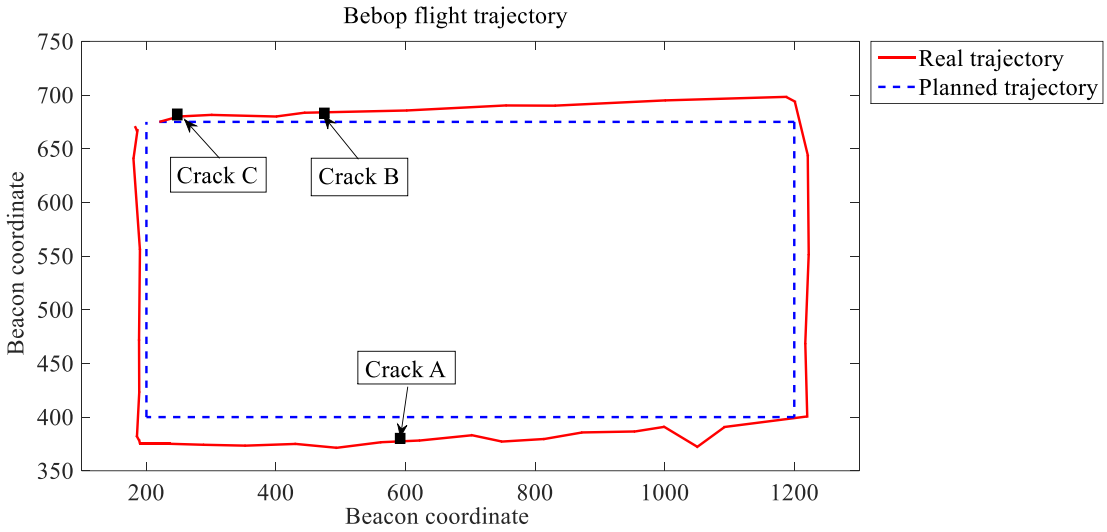


(a) Crack A (b) Crack B (c) Crack C

Figure 3-19. Geo-tagging results of the Bebop2 autonomous UAV (Kang and Cha, 2018)



(a) Case 1



(b) Case 2

There are possible limitations in this work. The flight controller (ArduPilot 3.5) works by trial and error to tune its parameters. We also used a cheap frame and batter which can reduce the flight time of the UAV. The manufacturer suggests that the Pixhawk UAV can fly for 15 min, but it does not guarantee the actual flight time. In most cases of normal use, only 70% of this projected flight time is achieved. The potential payload is 2 kg, but the actual payload is around 80% of this figure. Even though video data can be transferred in real time from a UAV to a base station, we did not pursue real-time processing for the CNN analysis because the CNN requires at least 8

seconds to analyze a one frame of image with the current pixel resolution. However, deep learning is an emerging area of research, and we expect that, in the near future, real-time processing will become feasible. The first limitation of UBS is that it requires an installation before data acquisition can be performed. However, modern SHM still needs sensors to be installed for data acquisition under the bridge. The second limitation of UBS is that the coverage area of a set beacon system is 30 m × 30 m, meaning that the beacon system is particularly well suited for short-distance applications (Perez-Grau et al., 2017). It is possible to increase this coverage by installing additional stationary beacons. The third limitation of UBS is that ultrasonic signals cannot penetrate walls or other obstacles, but this is similar to any other GPS, beacon, or Vicon system. However, a virtual map can be easily expanded by installing more stationary beacons.

3.5 Conclusion

This research proposed an autonomous UAV-based damage detection method using an UBS for indoor environments and areas in which GPS is denied or unreliable. Based on our extensive literature review, there are no published papers that propose autonomous navigation methods in GPS-denied infrastructure areas for structural health monitoring. The main contributions of this article are as follows: (1) it is the first application of ultrasonic beacon for UAV navigation for a GPS-denied environment, such as indoors and beneath a bridge (which is a critical area that should be monitored) or indoors for SHM, (2) we examined the possibility of using the video data collected from the UAV for deep learning-based automatic damage detection (Video data collected from the UAV has vibration issues, including the jello effect. Until now, little research has been done to detect structural damage using video data collected from a UAV with deep learning. Our previous deep learning-based damage detection (Cha et al., 2017a) employed only hand-held camera data that had no vibration issues.), (3) the detected damage was localized using the geo-tagging method, and (4) all these advanced technologies were integrated to realize autonomous UAV-based damage detection for a GPS-denied environment.

To realize this proposed method, we conducted the following: (1) fabricated a UAV using various parts that are commercially available, such as a frame, a flight controller, a telemeter, and an action camera, instead of using a premanufactured UAV, for which it is not generally possible to modify the source code for autonomous navigation; (2) modified the source code of the flight

controller firmware; (3) integrated an UBS with the autonomous flight controller; (4) replaced the GPS coordinates with a UBS signal in the image metadata for geotagging; and (5) adopted a small, commercially available Bebop2 UAV for a more complex mission with longer distance navigation, integrating the UBS and modifying the source codes to control its flight.

To demonstrate the proposed approach, three different indoor tests were conducted: (1) a hovering test to validate autonomous mission flight, (2) a waypoint based mission test with a specific waypoint, and (3) a complex mission with long-distance navigation using the commercial Bebop2 UAV with geo-tagging for damage localization. The autonomous flight of the UAV was successfully validated based on flight log data from these tests. The overall flight was quite accurate, but there were some fluctuations in altitude. As the final objective of this autonomous UAV, a concrete crack was detected with high accuracy (96.6%), sensitivity (91.9%), and specificity (97.9%) using video data collected from the autonomous UAV.

This CNN performance was well matched to the author's previous results using deep CNN-based damage detection (Cha et al., 2017a). The results of the image collected from the UAV were also compared to the results of manual image collection to validate the potential of the autonomous UAV-based health monitoring of infrastructure. The results from both images agreed well with high accuracy. As a future study, an additional obstacle avoidance sensor will be installed in our UAV to avoid the obstacle, and a more complex autonomous mission will be conducted for the SHM application. Real-world application will be carried out in the future to examine environmental effects, such as temperature and wind.

Chapter 4. Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning

Chapter 4 is reprinted with permission from Elsevier from Kang, D., Benipal, S.S., Gopal, D.L. and Cha, Y.J, 2020. *Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning.* Automation in Construction, 118, p.103291.

Abstract: This paper proposes an automatic crack detection, localization, and quantification method using the integration of a faster region proposal convolutional neural network (Faster R-CNN) algorithm to detect crack regions. The regions were located using various bounding boxes and a modified tubularity flow field (TuFF) algorithm to segment the crack pixels from the detected crack regions. A modified distance transform method (DTM) was used to measure crack thickness and length in terms of pixel measurement. To validate the proposed method, 100 images were taken in different places with complex backgrounds containing different angles and distances between the camera and the objects. The results obtained from the Faster-R-CNN-based crack damage detection had a 95% average precision. The pixel-level segmentation performance of the modified TuFF algorithm exhibited an authentic outcome, with 83% intersection over union. Finally, the modified DTM algorithm provided 93% accuracy with respect to crack length and thickness with a 2.6 pixel root mean square error.

4.1 Introduction

Cracks in a concrete surface are a common symptom and precursor of the degradation of concrete structures. Civil infrastructure systems are subjected to cyclic loading, fatigue stresses, and undesirable long-term environmental conditions, which lead to structural deterioration and, ultimately, a reduced lifespan. Maintenance and inspection deficiencies are also key reasons for structural deterioration. Early assessment and investigation enable the use of safety measures to forestall damage and failure. Periodic and real-time inspection enhances the longevity of structures and their service lives (McCrea et al., 2002). There is an indispensable demand to develop authentic, reliable, and decisive approaches for the monitoring of these structures. During the last three decades, a number of contact- or imbedded-sensor-based methods for detecting cracks have been published in the field of structural health monitoring (Kee & Zhu., 2013; Zoidis et al., 2013). However, contact-sensor-based approaches have some constraints in terms of data reliability, environmental unfavourability, and vulnerability to variations in temperature and humidity (Li et al., 2015; Xia et al., 2012).

To overcome these shortcomings, some automated vision-based techniques for damage detection have been developed (Adhikari et al., 2014; Alam et al., 2015; Cha et al., 2016; Hutchinson & Chen 2006; Ramana et al., 2018; Pacheco et al., 2014; Valenca et al., 2013). These vision-based techniques detect structural damage, such as cracks, spalling, exposed aggregates, and loosened bolts, using bounding boxes. Recently, deep-learning-based methods for detecting concrete cracks and different types of damage have been also developed (Ali & Cha, 2019; Beckman et al., 2019; Cha et al., 2017; Cha et al., 2018; Kang & Cha, 2018). These deep-learning-based approaches, which used bounding boxes to detect multiple types of damage simultaneously, showed extremely good results in detection and localization.

Some pixel-level methods for detecting concrete or pavement cracks using deep convolutional neural networks (CNNs) have been proposed (Cheng et al., 2018; Dung & Ahn, 2019; Escalona et al., 2019; König et al., 2019; Li et al., 2019; Liu et al., 2019b; Yang et al., 2018; Zhang et al., 2019a). For crack segmentation, pretrained U-Net (Ronneberger et al., 2015) architecture was applied by Cheng et al., (2018) and Escalona et al., (2019), and Li et al., (2019) used a pretrained Densenet 121 (Huang et al 2017) for semantic segmentation. The results of this study show high performance for the detection of different types of damage (such as cracks,

spalling, holes, and efflorescence) in a concrete surface. A fully convolutional neural network was applied by Yang et al., (2018) and Zhang et al., (2019a). Dung and Anh, (2019) tested various images for semantic segmentation results, and Liu et al., (2019b) suggested new customized architecture for crack detection.

However, these studies have some limitations, as follows. (1) Most of them have focused on crack detection across monotonous backgrounds, such as pure concrete member surfaces and pavement surfaces. Moreover, finding the optimal network architecture to segment cracks with such complex backgrounds is difficult, resulting in more realistic and practical problems. (2) In the traditional deep-learning only- based method for crack segmentation, a tremendous amount of time is required to build enough training data, which is a huge obstacle to applying these deep-learning-based crack segmentation methods. To overcome the aforementioned limitations, we propose a new pixel-level method for detecting concrete cracks. This method can determine, segment, and quantify cracks across varied and complex backgrounds, including different objects such as windows, corrosion, shoes, and wire, except notable dust. The proposed method is designed to enable the investigation of more realistic conditions of civil structural damage detection problems and reduce the cost of building datasets for deep learning training.

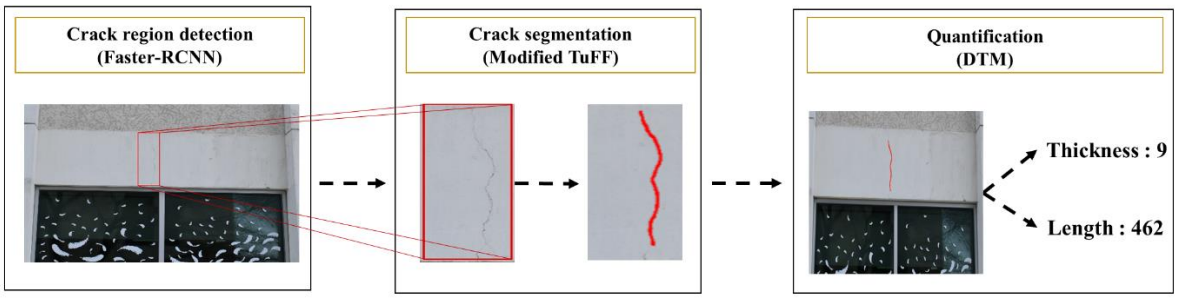
To realize this objective, we carefully integrated three independent computer vision algorithms: (1) crack detection using a faster region proposal convolutional neural network (Faster R-CNN; Ren et al., 2015; Cha et al., 2018) in terms of bounding boxes; (2) pixel-level crack segmentation using a modified tubularity flow field (TuFF) from the bounding boxes provided by the Faster-R-CNN-based crack detection method; and (3) damage quantification using a modified distance transform method (DTM) to calculate crack thickness and length from the segmented cracks noted by the modified TuFF. The original DTM (Paglieroni, 2011, Zhu, 2011) and original TuFF (Mukherjee et al., 2015) were modified to improve the performance of crack segmentation with thickness and length measurement for the proposed method.

This paper is organized as follows: Section 4.2 describes the details of the proposed method, Section 4.3 discusses related studies and testing results, and the Conclusion summarizes the studies and results.

4.2 Hybrid method of crack segmentation and quantification

To detect, localize, and quantify concrete cracks across varied and complex input image backgrounds, we propose the use of a fully automated method with the careful integration of a Faster-R-CNN-based crack detection method, a modified TuFF for crack segmentation from the detected cracks, and a modified DTM for measuring the thickness and length of the segmented cracks from the modified TuFF. An overall schematic view of the proposed hybrid method is presented in Figure 4-1.

Figure 4-1. Overview of proposed hybrid crack detection and quantification method (Kang et al., 2020)



The Faster-R-CNN-based method (Cha et al., 2018) detects concrete crack regions in a digital image using bounding boxes. The Faster R-CNN algorithm was initially developed for multiclass object detection and verified for structural damage detection (Cha et al., 2018). In this paper, Faster R-CNN is used to localize cracks using bounding boxes on an image, as shown in Figure 4-1. The second image in Figure 4-1 displays the regions of cracks detected using bounding boxes, which are then cropped in order to be fed into the modified TuFF algorithm. The original TuFF algorithm (Mukherjee et al., 2015) was modified to segment the cracks at the pixel level from the bounding boxes. Lastly, a modified DTM is applied to determine the thickness and length of the segmented cracks from the modified TuFF. In this process, the image binarization is performed using filters, such as Weiner and Gaussian filters (Das et al., 2015), which help to minimize noises and noninterest regions in images. The main advantage of this newly proposed method is that it is significantly less costly to build a training dataset for the Faster-R-CNN-based approach compared to the deep-learning-only-based crack segmentation method, because drawing

bounding boxes on training images to make ground truth is substantially easier than accurately marking all pixels of cracks on images. To prepare ground truth data for segmentation, 30–40 min per image is needed (Ren et al., 2020). However, the preparation of ground truth data in the proposed method only takes 0.5 min per image. The modified TuFF provides segmented cracks from the crack regions detected by the Faster R-CNN method. The Faster R-CNN can detect crack regions very well even on complex backgrounds (Cha et al., 2018; Beckman et al., 2019). The details of each part of the proposed method are explained in the following subsections.

4.2.1 Faster R-CNN for crack detection

In the first step of the proposed method, a Faster R-CNN is used to detect concrete cracks in images of varied and complex backgrounds. The Faster R-CNN is composed of two different networks: a region proposal network (RPN) and a fast region-based convolutional network (Fast R-CNN; Girshick, 2015). The RPN provides possible object locations using various bounding box sizes, and as a classifier, the Fast R-CNN proposes the probability of the object. The Faster R-CNN processes input images quickly, because the RPN and Fast R-CNN share the base CNN with the help of a graphics processing unit.

Figure 4-2. Schematic representation of Faster R-CNN architecture (Kang et al., 2020)

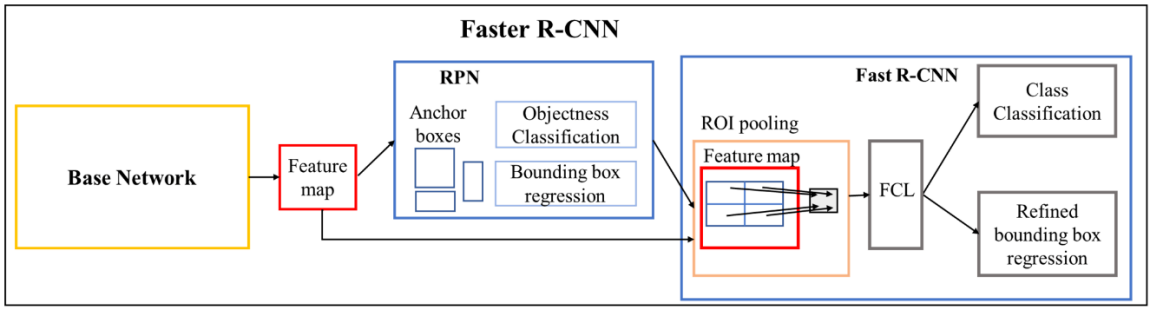


Figure 4-2 illustrates the architecture of the Faster R-CNN used in our proposed method. The input can be a target image or video frame. The purpose of the base network (i.e., CNN) is targeted feature extraction, which is similar to other object detection methods, whereas a deeper network can usually provide more accurate features (Szegedy et al., 2015). Original Faster R-CNN uses the VGG 16 (Simonyan and Zisserman, 2014) as a base network. Base networks are improved

using various techniques to make deeper networks and improve object detection accuracy. In this paper, we applied a pretrained ResNet-50 network (He et al., 2016) with a Microsoft COCO dataset (Lin et al., 2014) to the base network. To extract the feature map, images were fed to the base network, and the output of the base network was used as the input of the RPN to provide possible locations of detected objects.

To make region proposals, an n × n spatial window was slid across the feature map, and at each scanning point, multiple region proposal regressions were performed. For each regression part, nine different anchor sizes were generated and selected. Classification layers distinguished between objects and backgrounds. Detected objects were the output of the object classifier and the refined bounding box regressors.

To detect concrete cracks across varied backgrounds, the RPN and Fast R-CNN should be trained using a labelled image dataset. During the RPN training, classification layers use anchors and ground truth boxes, which are manually labelled. If the overlap rate between anchor and ground truth box is more than 0.7 (Ren et al., 2015), the anchor box is regarded as an object. Otherwise, it is a background. Regression layers provide a bounding box with dimensional information, such as center coordinates (x and y) and box size (width and height). As a result, the RPN provides the object proposals and scores.

The Fast R-CNN used a fully connected layer (FCL), in which the detection layer cannot handle the differences of input size. Therefore, the Fast R-CNN adopted region of interest pooling (ROI pooling), which identifies different proposed region sizes from the RPN and the base network. ROI pooling is able to accept different input sizes, and the results of the FCL are class classification and bounding box regression. The Faster R-CNN has two different classifiers, the object classifier in the RPN and the class classifier in the Fast R-CNN. The object classifier in the RPN is a binary classifier; that is, it classifies between objects and backgrounds. The class classifier in the Fast R-CNN provides detailed classification results from different classes. In this research study, the object classifiers distinguished between cracks and backgrounds.

For the Faster R-CNN training, a four-step alternating training procedure was used (Ren et al., 2015). First, the RPN was trained without the Fast R-CNN part for object and background classification. Second, the Fast R-CNN part was trained. Third, the RPN was fine-tuned using a

second-step network, and the weights of the base network were updated. Fourth, the Fast R-CNN was fine-tuned using the base network from the third step. More detailed information about the Faster R-CNN is available in the authors' previous study (Cha et al., 2018). The training image dataset for this Faster R-CNN is developed and tabulated in Table 4-1. An image augmentation technique was applied to the original set of 400 training images to increase the number of training images to 1,200.

Table 4-1. Image dataset for faster R-CNN

	Training	Validation	Test
# of images	1200	100	100
Resolution	1,920 × 1,080	1,920 × 1,080	3,200 × 4,800, 1,960 × 4,032

The training and validation images were collected from previous studies (Cha et al., 2018; Kang & Cha, 2018). The test images were newly collected near the University of Manitoba, the Bridgewater Forest area in Winnipeg, and from the Internet. Thirty percent of the test images were taken from an indoor site at the EITC in the University of Manitoba, whereas 70% were outdoor images. All the datasets for the training, validation, and test were completely different from one another and were randomly selected. Most of the cracks are wide cracks, which are more than 2 mm thick. In 100 images, 126 cracks exist. The number of thin, medium, and wide cracks is 10, 25, and 91, respectively. Thin cracks are less than 1 mm thick, whereas medium cracks are 1 to 2 mm thick. The training and validation images were randomly selected, and the training, validation, and test image datasets did not use the same images.

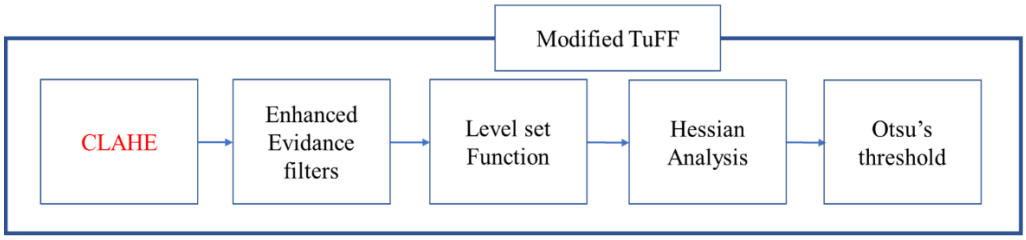
4.2.2 Modified TuFF method for pixel-level crack segmentation

The TuFF method was originally developed for the segmentation of neurons from confocal microscopy images (Mukherjee et al., 2015). This method was previously applied to detect vessels, but in our paper, it is applied to segment the detected cracks at the pixel level from the Faster-R-CNN-based crack detection method using bounding boxes, as shown in Figure 4-1. The crack regions bounded by the boxes have an uneven contrast of concrete surfaces, which result in poor crack segmentation. Therefore, contrast limited adaptive histogram equalization (CLAHE; Reza, 2004) is integrated to improve the segmentation results of the TuFF, as shown in Figure 4-3 (a).

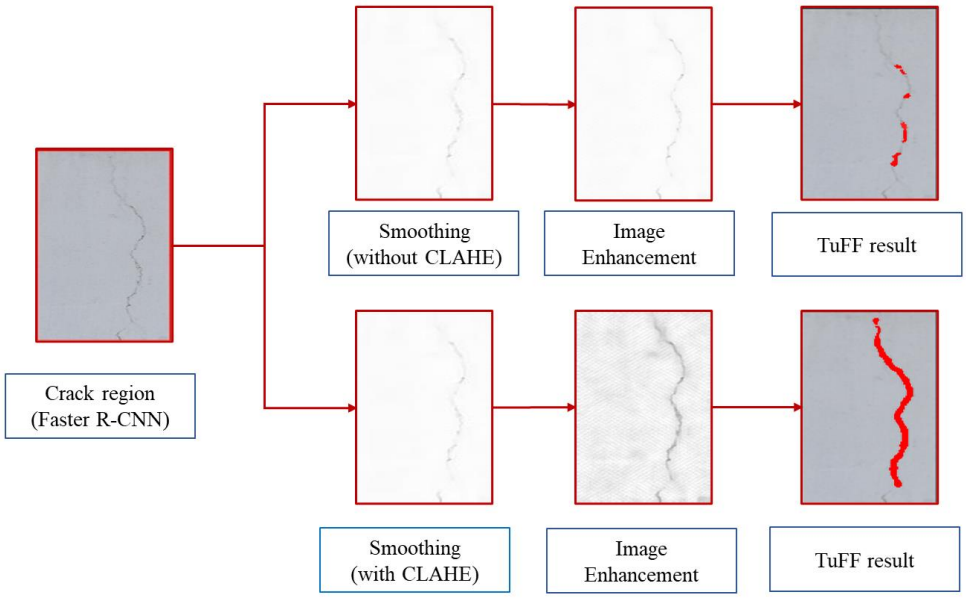
The row input images provided by the Faster R-CNN were processed by CLAHE. The image processed by CLAHE is fed into TuFF to perform the semantic detection, as shown in Figure 4-3 (b). Additional comparative studies of the traditional TuFF and modified TuFF with CLAHE are described in Section 4.2.1.

This improved TuFF technique with CLAHE performs segmentation through an evolution of the level set function. To evolve the level set function, an improved Hessian matrix analysis based on a Gaussian filter was used. The directional vectors from the TuFF make the contour active towards the boundary of the object. This active contour provides the information about the object boundary for the segmentation of crack regions, because it can delineate the object boundaries with higher subpixel accuracy (Mukherjee et al., 2015).

Figure 4-3. Details of modified TuFF (Kang et al., 2020)



(a) Procedure of modified TuFF



(b) comparison TuFF and modified TuFF

The second-order Gaussian derivative Hessian matrix is applied to input image Ω:

$$H_\sigma(x,y) = [h]_{i,j}\ (1 \le i,j \le 2), (x,y) \in \Omega, \tag{4-1}$$

$$[h]_{i,j} = \frac{\partial^2 G(\sigma)}{\partial x_i \partial x_j} * f(x,y), \tag{4-2}$$

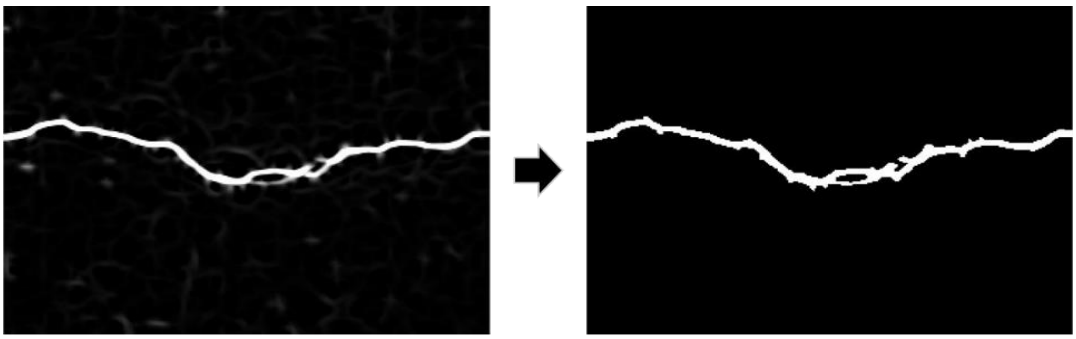
where, G(σ) is the Gaussian kernel. σ refers to standard deviation (zero-mean normalized). f(x,y) is the pixel coordinates corresponding to the position (x,y)ϵ Ω. To evaluate the vesselness (i.e., crack, in this instance) of a particular pixel f(x,y), the following vesselness equation was used:

$$v = \begin{cases} 0 \\ \exp\left(-\frac{R_B^2}{2\beta^2}\right)\left(1-exp\left[-\frac{S^2}{2c^2}\right]\right) \end{cases} \text{if } \lambda_2 > 0, \tag{4-3}$$

where λ_1 and λ_2 refer to the eigen values from the results of the Hessian analysis. D is the dimension of the image based on the nature of the eigen value. $R_B = \frac{|\lambda_1|}{|\lambda_2|}$, $s = \sqrt{\sum_{j \le D} \lambda_j^2}$, β and c are the thresholds, determined by trial and error, that control the sensitivity of vesselness.

If λ_1 and λ_2 are low value and high positive value, respectively, or low value and high negative value, respectively, the pixel is included in the tubular structure (Frangi et al., 1998). The norm of the Hessian is calculated to consider that the backgrounds are brighter and their eigenvalues are relatively large compared to that of the darker area (crack).

Figure 4-4. Process after crack indication is achieved using the enhanced TuFF (Kang et al., 2020)



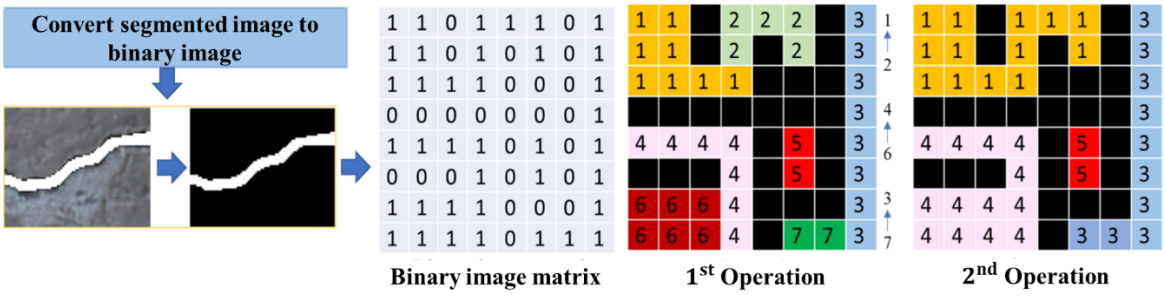
(a) Crack indications (b) Otsu's threshold

However, this traditional TuFF is not robust to the detection of complex tubular regions, such as vessel bends and junctions in the image, because the Hessian-matrix-based tubular regions are constructed as a piecewise rigid template. This drawback results in discontinuities in the tubular output. Therefore, vessel evidence filters for the detection of tubular structures were proposed (Hao et al., 2017) by integrating two local filters to create an enhanced TuFF method for the segmentation of vessel regions. The local filters were generated by steering the mother filter Equation (4-1) to improve the connectivity of the detected pixels by considering angles with neighbor pixels. Therefore, the enhanced evidence filter, which can consider the local neighborhood of the testing pixel, was defined by superimposing the mother filter and two local filters. To highlight only the crack regions and normalize any pixel in a non-interested region, Otsu's thresholding method was applied to assist with converting any area with few tubular pixels (small, insignificant spots of the white region in Figure 4-4(a)) to background (black) pixels, as shown in Figure 4-4 (b).

4.2.3 Crack quantification using a modified DTM

The last step of the proposed method is a quantification of the segmented cracks from the improved TuFF method. The first step of a traditional DTM is converting an input image to a binary image that expresses "0" or "1" for black or white, respectively (as shown in Figure 4-5) using the inbuilt MATLAB function "bw" (Mathworks, 2018).

Figure 4-5. Traditional DTM procedure (Kang et al., 2020)



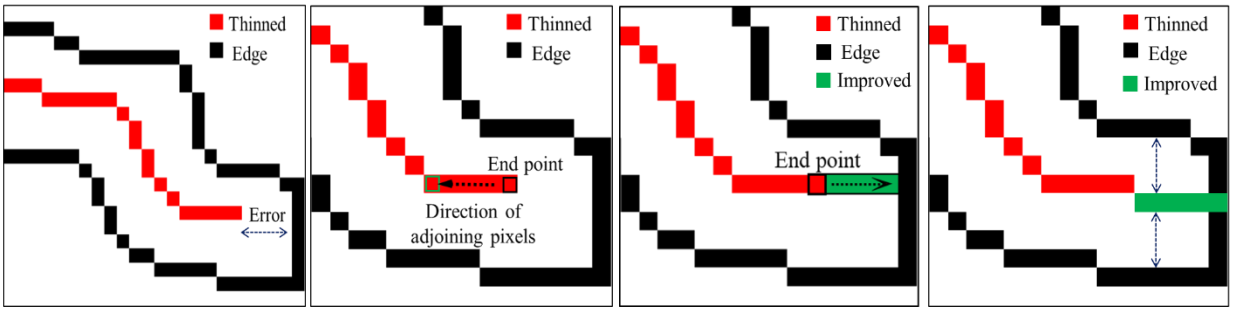
To calculate the thickness and length of cracks, the first step is to identify the connected pixels in an image and label them with a numeric value. A labeling operator goes through all the pixels to find clusters of the same values (i.e., 1 or 0) and assign each cluster a unique value. Figure 4-5 illustrates the detailed labelling process for a binary image matrix. Every image labelling

process starts at the top-left corner. For example, in the third image of Figure 4-5, a yellow pixel (1,1) is labelled as 1 and copied at (1,2). The next pixel (1,3) is a region of non-interest and marked as 0. Therefore, the labelling operator understands that there is a new cluster and labelling operator assign value 2 for this cluster and the operator continues to label the subsequent pixels [(1,5) and (1,6)]. After pixel (1,6), there is another background pixel followed by a crack pixel (1,8), which should be marked as 3. The first pixel (2,1) of the second row starts with 1 because it is connected to the pixel (1,1). In this manner, entire pixels are assigned cluster numbers (1–7). However, some crack pixels have different cluster numbers despite being connected with each other. To merge these pixels, the same operator (i.e., the second operation shown in Figure 4-5) is processed again, and eventually the final matrix of Figure 4-5 is achieved.

To calculate the length and thickness of a crack, the center pixels in each row of each cluster determined using the previous labeling operator should be identified; this process is called thinning. Based on these center pixels, the thickness and length of the cracks are easily calculated using the pixel level. Therefore, to identify the center pixels, a thinning process is carried out (Lee et al., 1994) to calculate the relative distance between both edges and to find the center pixel in each column of the image matrix. However, a thinning algorithm has local branch issue. To remove unnecessary local branches, the “bwmorph” imbedded function is used to prune the branch (Mathworks, 2018). One of the results of a traditional thinning process is shown in Figure 4-6 (a). The red and black pixels of Figure 4-6 represent the center and edge pixels of a crack, respectively. This set of processes is the procedure of the traditional DTM method.

The length of a crack is generally calculated by considering the center pixels, but the traditional thinning process exhibited a number of errors that cannot be neglected, as shown in Figure 4-6 (a). Therefore, the end pixel of a thinned line is the starting point for reaching the edge of the crack pixels. As modified DTM, in Figure 4-6 (b), the direction of adjoining pixels was found from the end point. Subsequently, we added the pixels in the direction opposite to the previously identified direction until we reached the edge of crack, as shown in Figure 4-6 (c), which is the result of the improved DTM method. In Figure 4-6 (d), we checked the distance around the edge pixel and extended the thinned line to the edge pixel.

Figure 4-6. Improved thinning process: (a) the result of traditional thinning at the end of a crack, (b) checking the direction of adjoining pixels, (c) extending the thinned line to the edge of a crack, (d) the improved result (Kang et al., 2020)

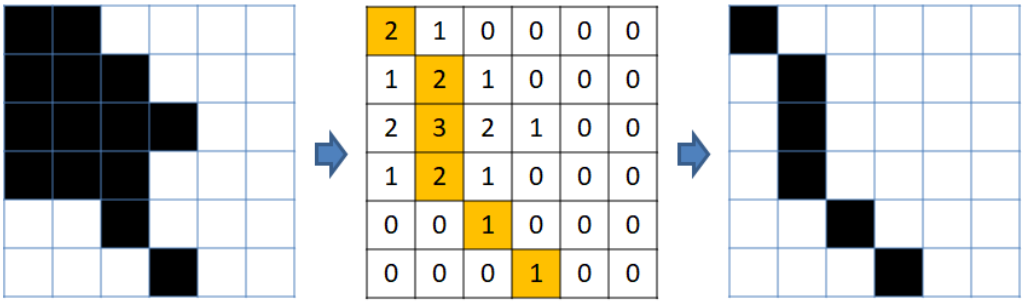


(a) (b) (c) (d)

As shown in Figure 4-7, the result of the improved TuFF was fed into the modified DTM and produced counted pixel values for the thickness. However, to calculate the final thickness in each row and column, Equation (4-4) is proposed. The x value represents the count from the edge of the crack to the center pixel of the crack width. The eventual length of the crack is calculated using the MATLAB inbuilt function "NNZ" (Mathworks, 2018) and the results of the improved thinning method.

$$Thickness = Max(x) \times 2 - 1 \tag{4-4}$$

Figure 4-7. Overall process of quantification (Kang et al., 2020)



4.3 Case studies and discussion

The proposed method used a Faster R-CNN technique to detect cracks using bounding boxes, a modified TuFF algorithm with CLAHE to segment the region of the cracks at the pixel level, and

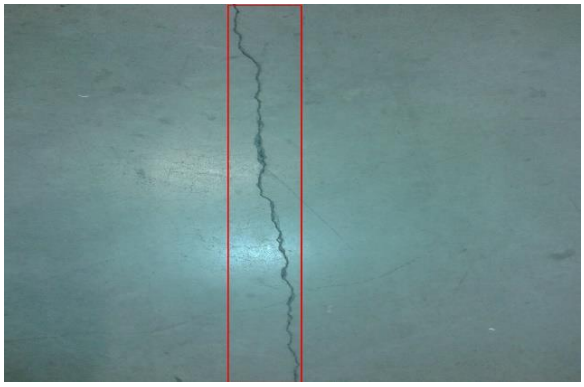
a newly improved DTM to measure crack thickness and length. To validate this method, various backgrounds and real structural cracks were tested. Moreover, two well-known deep learning networks, DeepCrack (Liu et al., 2019b) and Mask R-CNN (He et al., 2017), were used for a comparative study.

4.3.1 Faster R-CNN and improved TuFF results

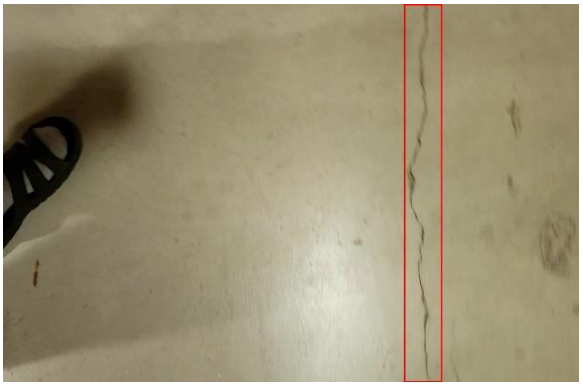
The first experimental tests were conducted in the Engineering and Information Technology Complex (EITC) at the University of Manitoba. Figure 4-8 shows examples of concrete cracks and the results of the Faster-R-CNN-based method, which detected most indoor and outdoor cracks accurately with spot lighting, blurry images, and different backgrounds.

Figure 4-8. Results of Faster-R-CNN-based crack damage detection (Kang et al., 2020)

(a) Outside of EITC: pavement



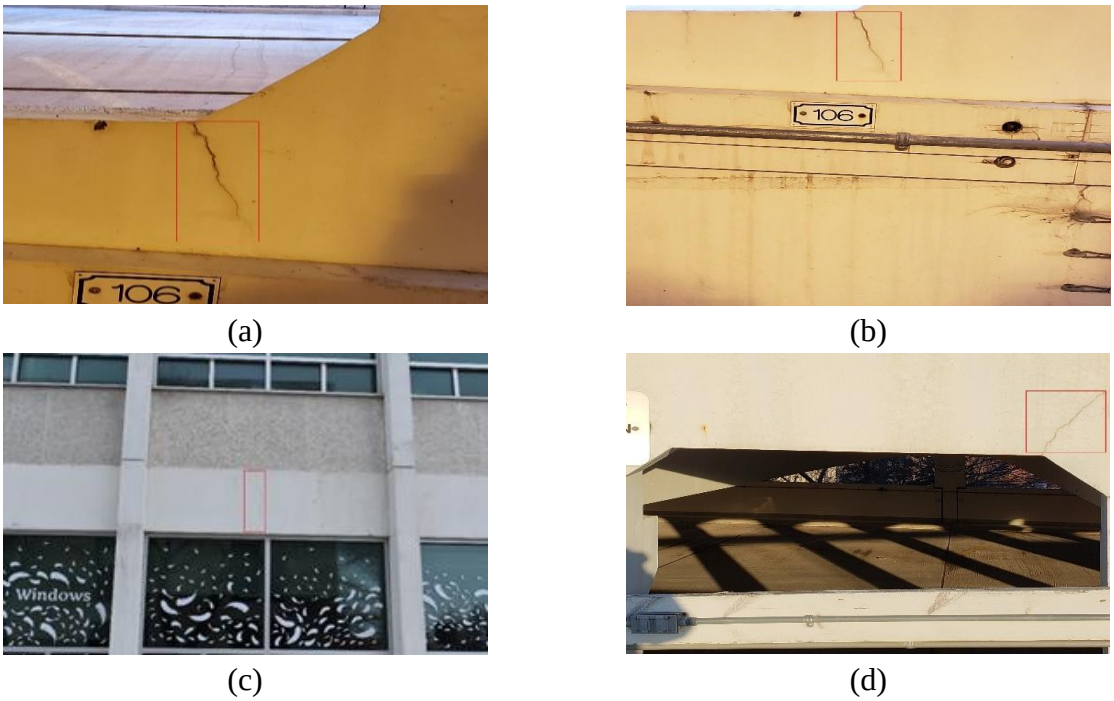
(b) Indoor corridor I of EITC: lightening



(c) Indoor corridor II of EITC: blurry image

(d) Indoor corridor III of EITC

Figure 4-9. Faster R-CNN for crack damage detection (Kang et al., 2020)



(a) (b)

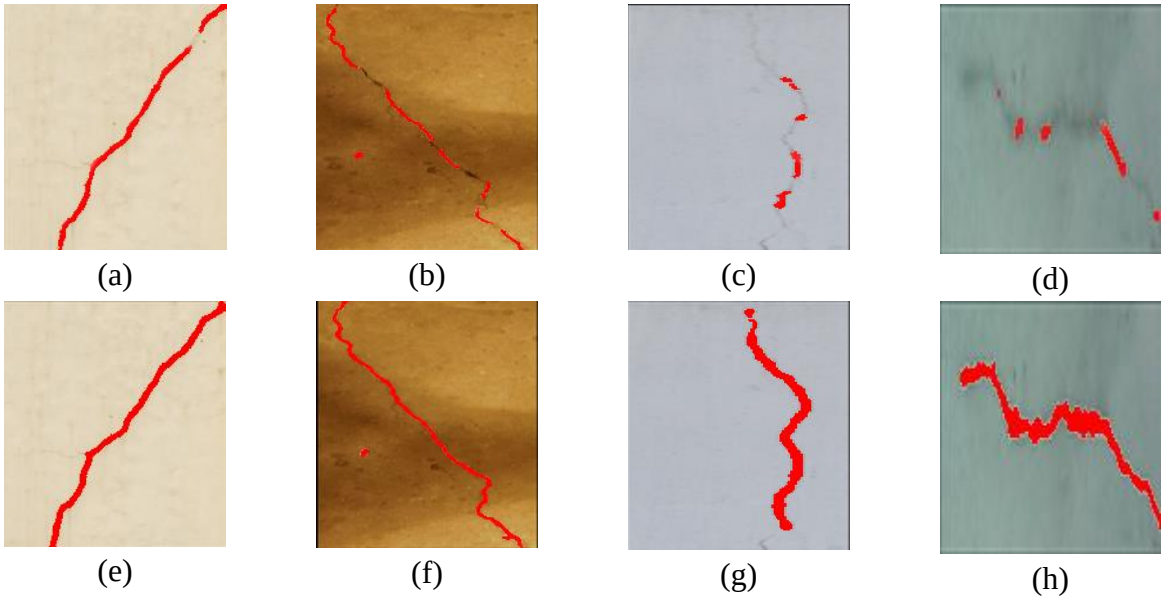
(c) (d)

Figure 4-9 shows outdoor cases at the University of Manitoba to provide more complex backgrounds, such as steel beam, markings, shadows, and rods. Images were collected in the presence of natural light and the width of cracks varied from few millimeters to centimeters. The testing results exhibited 95% average precision (AP) for 100 test images. The images used to test the Faster R-CNN, modified TuFF, and modified DTM were collected using a Nikon d7200 camera (Nikon, 2019) and a Galaxy s9 camera (Samsung, 2019) with resolutions of 3,200 × 4,800 pixels and 1,960 × 4,032 pixels, respectively. Publicly available crack images from the Internet were also used.

As Figures 4-8 and 4-9 show, the Faster-R-CNN-based method was validated as properly detecting cracks under various backgrounds and environmental conditions. However, it is not sufficient to quantify the detected cracks; therefore, the results of the Faster-R-CNN-based method were cropped with bounding box coordinates. The cropped images were then inputted into the modified TuFF with CLAHE to segment the cracks only at the pixel level. A Gaussian-filtering-based de-noising algorithm embedded in the original TuFF method not only removed the noises, but also blurred the gradient of cracks in the images, and the contrast of concrete surfaces varied.

Therefore, CLAHE was used to improve the contrast of the cracks (Reza, 2004). Figure 4-10 shows the subsequent effectiveness of CLAHE.

Figure 4-10. Comparisons between original and modified TuFF. Images (a)–(d) are original TuFF results, and images (e)–(h) are modified TuFF results (Kang et al., 2020)

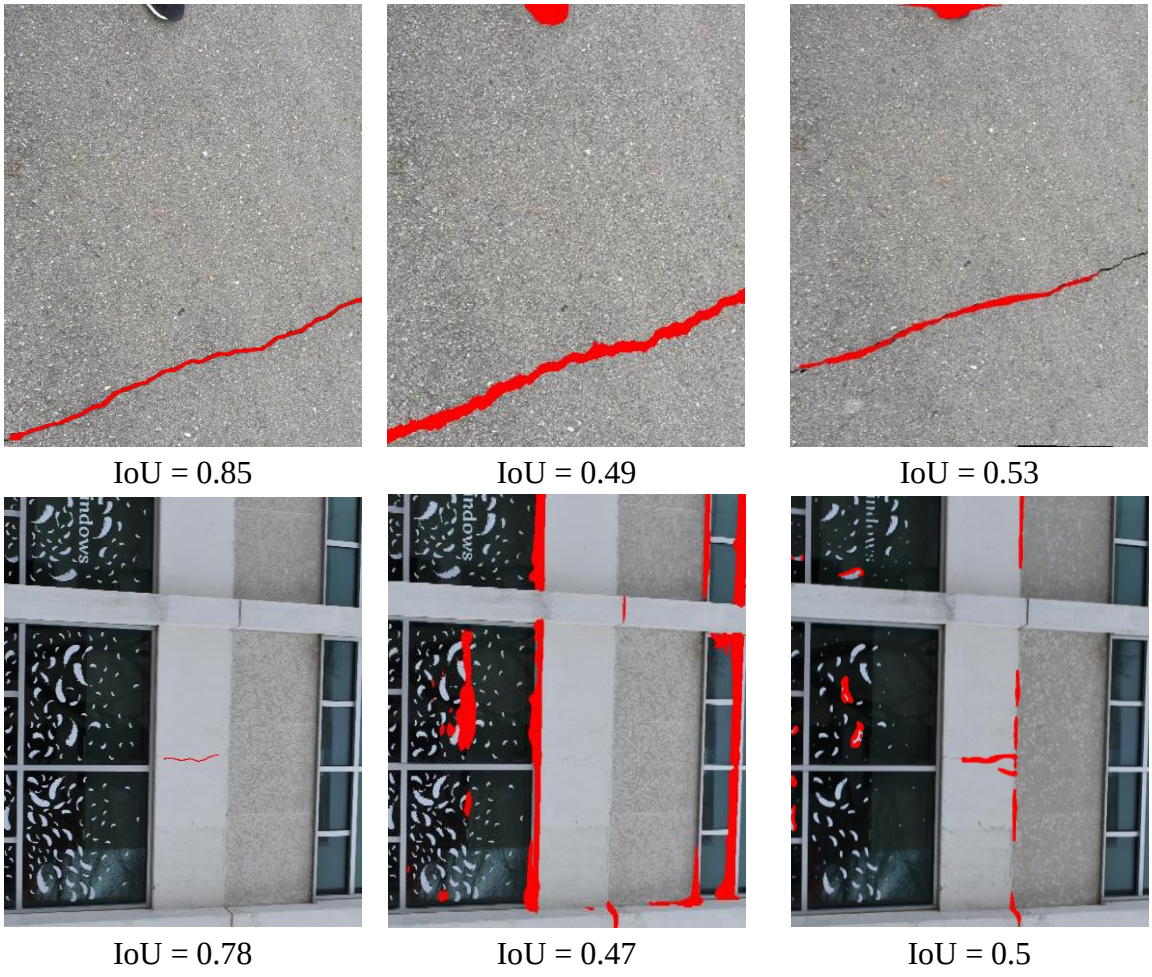


The results of the Faster-R-CNN-based method were inputted into the TuFF with and without CLAHE, as shown in Figure 4-10. Figure 4-10 (a–d) shows the results of original TuFF, and Figure 4-10 (e–h) shows the results of the TuFF with CLAHE. The TuFF with CLAHE exhibited significantly improved results compared to those of the original TuFF, increasing the total intersection over union (IoU) metric by 10%. To calculate the IoU, the number of pixels with the same value and same location from the target and prediction masks was divided by the total number of pixels present. To quantify the crack pixels in the original target images, the non-zero elements of a test image crack are counted. In the binary labeling, "0" represents the background pixels, and "1" represents the crack pixels. The IoU equation is

$$IoU = \frac{target \cap prediction}{target \cup prediction} \tag{4-5}$$

Figure 4-11 shows some examples of the results of the modified TuFF with CLAHE using the results of the Faster-R-CNN-based method. The cracks were from indoor and outdoor environments across varied complex backgrounds. The cracks were detected and segmented accurately under different lighting conditions. To compare the performance of our hybrid method to that of existing deep-learning-based crack segmentation methods, we used a DeepCrack network (Liu et al., 2019b) and Mask R-CNN (He et al., 2017). In order to compare the performance of these methods, we simply used the trained DeepCrack network by using 300 images as described in Liu et al., 2019b, and we trained a Mask R-CNN using the publicly available crack segmentation datasets (Eisenbach et al., 2017; Liu et al., 2019b; Shi et al., 2016; Zou et al., 2012).

Figure 4-11. Comparative studies using our hybrid method, DeepCrack, and Mask R-CNN1 (Kang et al., 2020)



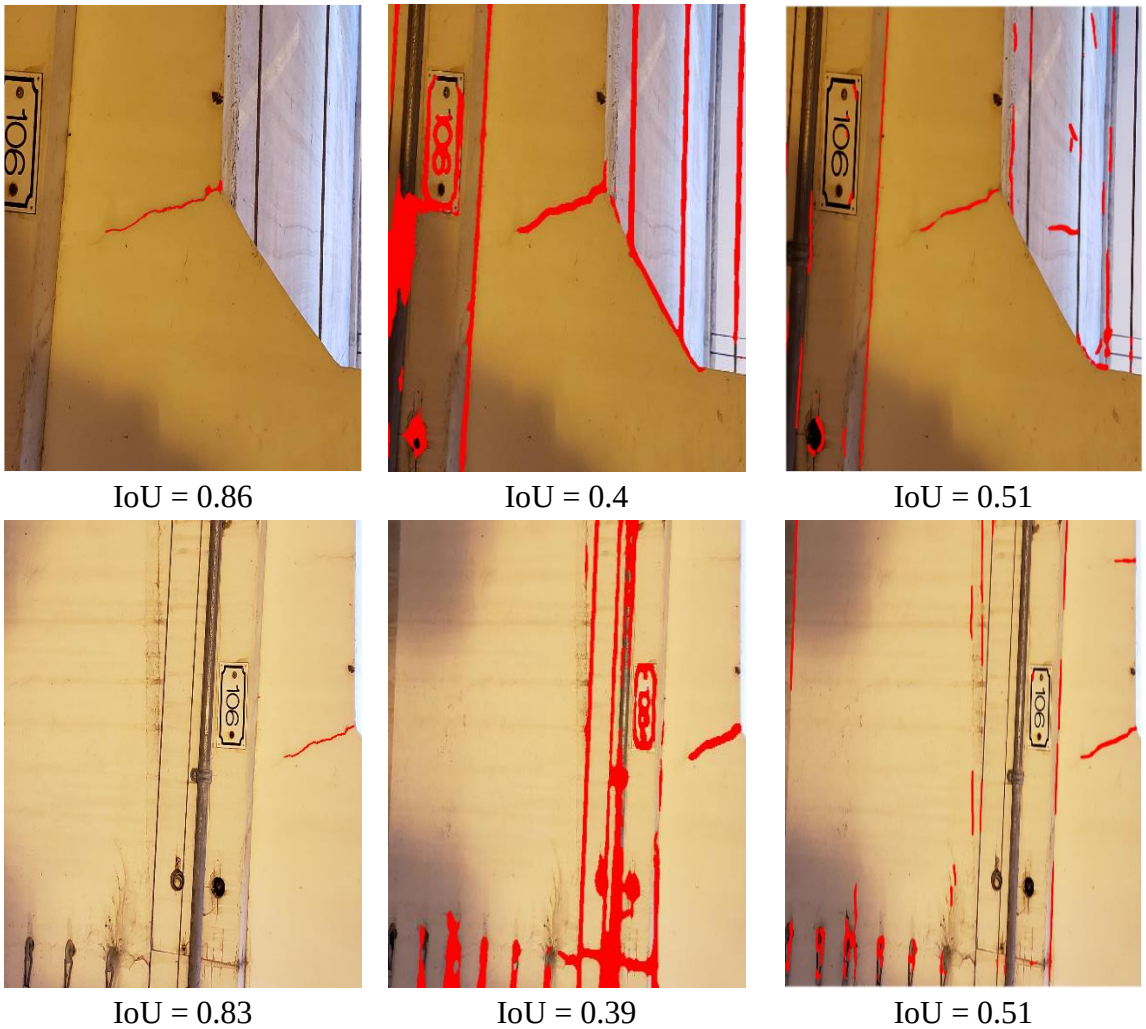
IoU = 0.83 IoU = 0.37 IoU = 0.51

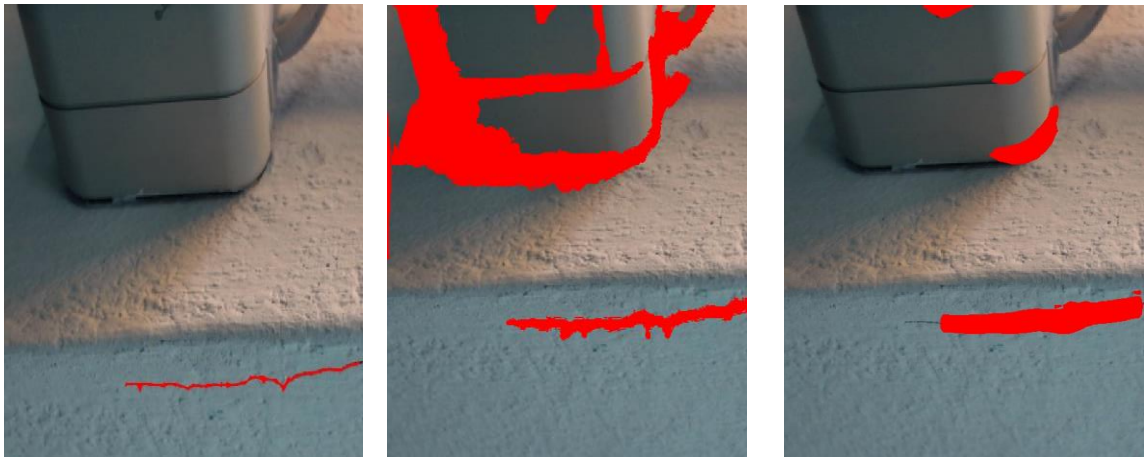
The results of the comparative studies are presented in Figures 4-11, 4-12 and Table 4-2. We used 100 testing images for this comparative study. Our hybrid method exhibited an 83% average IoU, DeepCrack exhibited a 45% average IoU, and Mask R-CNN exhibited a 61% average IoU. Our hybrid method demonstrated much better performance than that of Mask R-CNN and the DeepCrack network in the 100 testing images with complex background images. The DeepCrack network was not trained to consider complex backgrounds, and its training set did not include images with complex backgrounds. Therefore, this network has relatively lower average IoU which is reasonable even compared to recently published work having average IoU about 78% (Bang et al., 2019).

Mask R-CNN was also selected for the comparison. It is not a semantic segmentation method but an instance segmentation method. It can detect the boundary of the target object independently. Due to the nature of Mask R-CNN, segmenting cracks is a rather challenging task for this method, because it is not easy to clearly define the boundary of each crack if the crack line is long or complex. Therefore, we modified Mask R-CNN codes to enable the algorithm to perform semantic segmentation instead of instance segmentation. As there is no change in network structure and any hyperparameter setting, the performance of Mask R-CNN is maintained, and our evaluation process is followed. The backbone architecture of this modified Mask R-CNN is the ResNet-50 network (He et al., 2016), which is pretrained by ImageNet (Deng et al., 2009). This Mask R-CNN is trained by publicly available datasets (Eisenbach et al., 2017; Liu et al., 2019b; Shi et al., 2016; Zou et al., 2012).

Although the deep-learning-only-based crack segmentation methods (DeepCrack and Mask R-CNN) exhibited good performance in trained environments such as monotonous backgrounds, their performance was inferior to that of our proposed hybrid method when the networks were tested on untrained complex backgrounds, as shown in Figures 4-11 and 4-12. However, our hybrid method uses Faster R-CNN to find the crack region instead of segmentation from various complex backgrounds, and the detected crack region is inputted to the modified TuFF, which does not require any training data to segment the crack in the image.

Figure 4-12. Comparative studies using our hybrid method, DeepCrack, and Mask R-CNN2 (Kang et al., 2020)





IoU = 0.74 IoU = 0.41 IoU = 0.51

The training dataset for Faster R-CNN was easily built by drawing bounding boxes instead of using pixel-level marking. The time required to prepare the ground truth for DeepCrack, Mask R-CNN, and our algorithm is approximately 30, 30, and 0.5 min, respectively, as shown in Table 4-2. In order to prepare the mask, Affinity Photo, which is a commercial tool for processing photos, was used to draw an accurate mask (Serif, 2020), whereas Labeling was used to draw the bounding box (Tzutalin, 2015). Both data were prepared in a laptop that has Intel Core I7-6700HQ CPU, and 16 GB memory. Faster R-CNN, Mask R-CNN, and DeepCrack took 9, 9.5, and 14.5 s, respectively, when each algorithm was tested using the same image size of about 800 × 800 RGB. Our hybrid method used TuFF for segmentation, and it took around 4 s per one bounding box of 100 × 100 in size.

Table 4-2 shows that Mask R-CNN is used to more training data than DeepCrack network is, so it achieves better performance. However, DeepCrack and Mask R-CNN do not achieve better results than our methods in the same 100 test image set. Therefore, our method significantly reduces the cost of dataset building and provides a much better solution than traditional deep-learning-only approaches do.

Table 4-2. Training and testing information and result of average IoU

	DeepCrack	Mask R-CNN	Ours
Training data	300	833	400
Data augmentation	2400	2819	1200
Preparation training	Pixel mask	Pixel mask	Bounding box
data per image	30 min	30 min	0.5 min
Test data	100	100	100
Average IoU	45%	61%	83 %

4.3.2 Case studies of modified DTM

The proposed method successfully detected cracks at the pixel level. The detected cracks were then processed by a modified DTM to measure the thickness and length of cracks in terms of pixels. To verify the modified DTM algorithm, we ran our algorithm with a ground truth dataset. This dataset consisted of 100 numbers from binary images. These images were also used to test the modified TuFF. However, in this experiment, we did not consider the angle and distance between the camera and the objects.

The traditional thinning algorithm was still useful in obtaining the average thickness of the cracks, but, calculated entire crack lengths were inaccurate due to a local branching issue as depicted by the red bounding boxes in Figure 4-13 (b) in the “Original algorithm”. Our modified algorithm uses a “bwmorph” to remove unnecessary local branches. The right-side images in Figure 4-13 (c) show the results of branch pruning algorithm.

Figure 4-13. Removed local branches (Kang et al., 2020)

(a) Images (b) Original algorithm (c) Ours

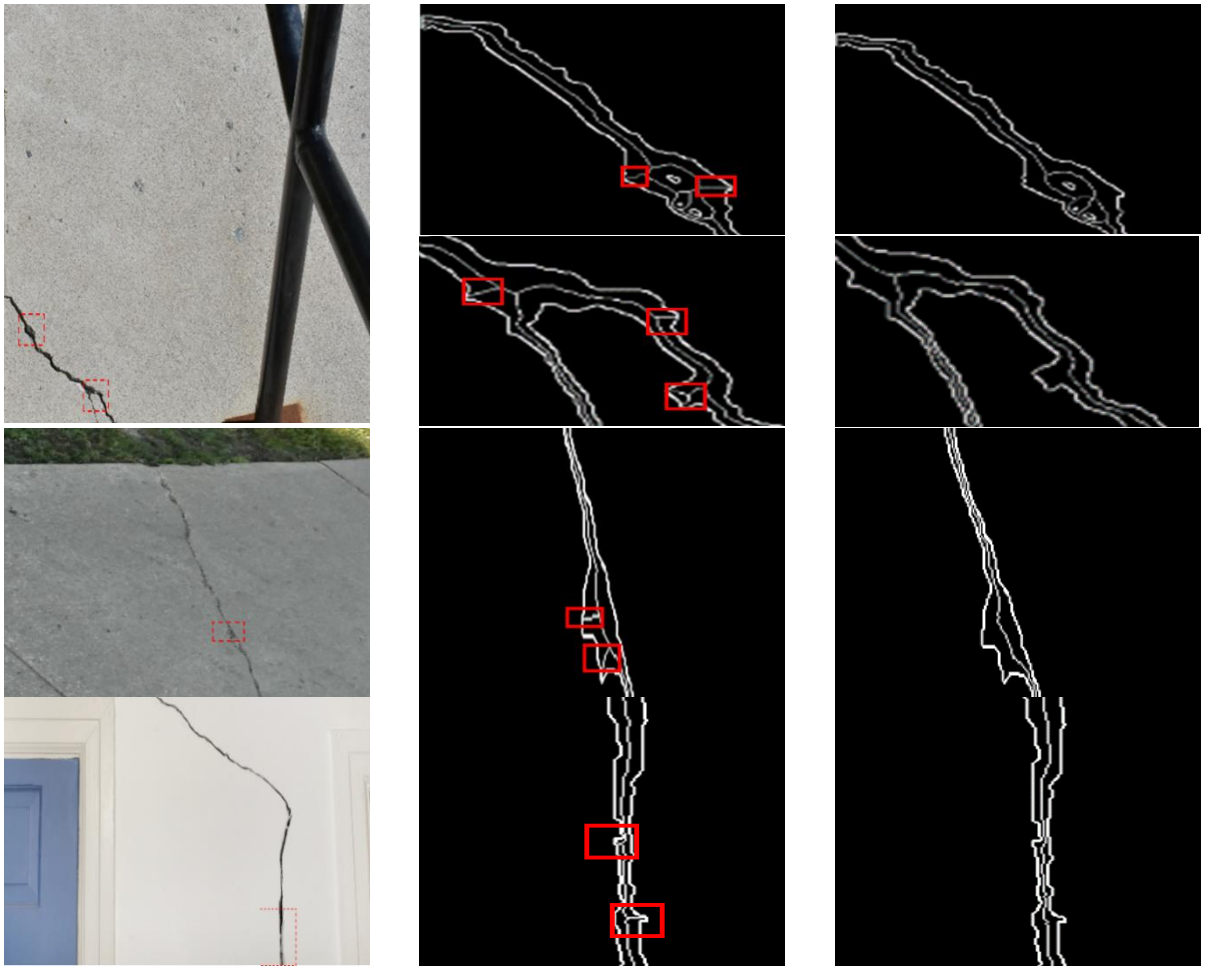
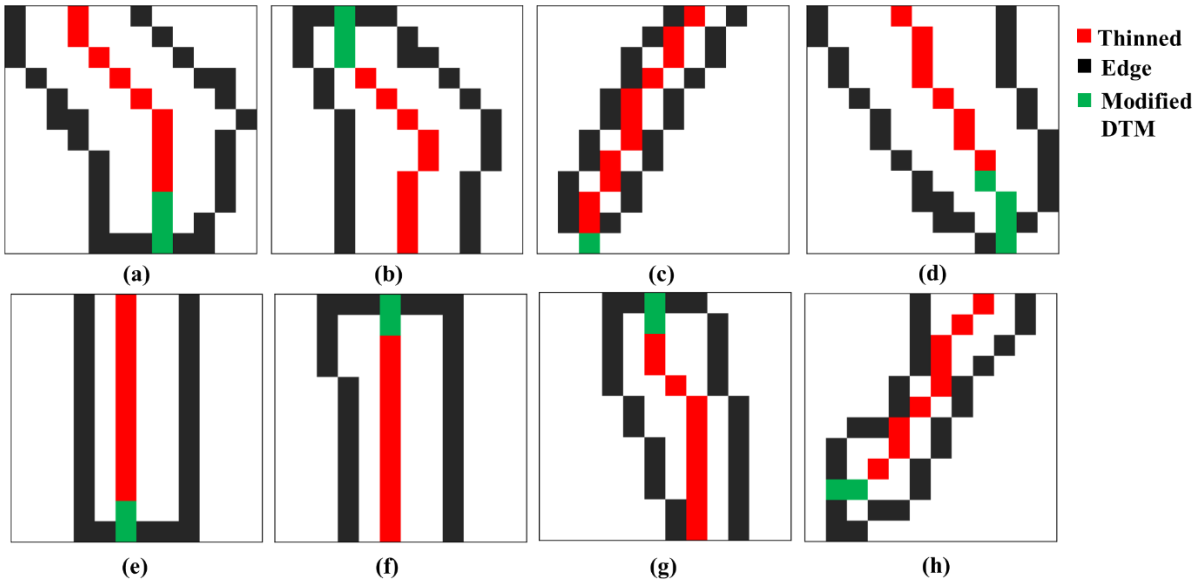


Figure 4-14 shows the detailed results of our DTM algorithms. The images are binary images, but we changed their color in order to present the results more clearly. The black dots in Figure 4-14 represent the edge extraction results, the red dots represent the thinning algorithm results (Mathworks, 2018), and the green dots represent our modified DTM results. The error occurred only at the ends of cracks, as illustrated in Figure 4-14.

Figure 4-14. Example results of modified DTM for crack quantification (Kang et al., 2020)



Twenty images sampled from 100 test images of the modified DTM results are presented in Table 4-3 as examples. The presented values refer to the minimum and maximum numbers of the pixels of a crack thickness and the number of pixels of a crack length. Furthermore, crack thickness was evaluated by root mean square (RMS) error, and crack length was evaluated by accuracy. Table 4-3 shows that the errors of the modified DTM in terms of pixel numbers are very small. Our modified DTM algorithm resulted in 99% accuracy in terms of length and 2.6 pixels RMS error in terms of thickness using the 100 test image set compared to the ground truth. The original thinning algorithm without our modification resulted in 98% accuracy in the 100 testing images, but this original DTM cannot calculate crack thickness. The RMS equation is

$$RMS\ error = \sqrt{\frac{\sum_{i=1}^{n}(Predicted - target)^2}{n}} \tag{4-6}$$

where, n is the total number of testing images (i.e., 100), $predicted$ represents the modified DTM result, and $target$ represents the ground truth."

This accuracy is quite high as same as Fast Marching based crack quantification (Tsai et al 2013). When the modified DTM algorithm is tested on the results of the TuFF, the accuracy is reduced to 93%. It is reasonable because our proposed TuFF method has accuracy of 83% in terms

of average IoU score which is quite superior to the other existing current state of the art as we explained in the previous sections (Table 4-2).

Table 4-3: Result comparison of manually measured and modified DTM

Case	Manual measurement	Proposed method	Error	Case	Manual measurement	Proposed method	Error
C1	(4,9,729)	(3,9,729)	(1,0,0)	C11	(2,11,325)	(2,9,326)	(0,2,1)
C2	(4,14,307)	(3,13,304)	(1,1,3)	C12	(2,13,305)	(2,13,306)	(0,0,1)
C3	(2,18,484)	(2,17,482)	(0,1,2)	C13	(2,11,322)	(2,10,321)	(0,0,1)
C4	(6,19,376)	(5,19,374)	(1,0,2)	C14	(4,11,300)	(3,11,300)	(2,1,1)
C5	(2,14,394)	(2,12,392)	(0,2,2)	C15	(3,11,522)	(2,12,520)	(1,1,2)
C6	(2,11,408)	(2,11,404)	(0,0,4)	C16	(2,17,1110)	(2,18,1113)	(0,1,3)
C7	(4,10,675)	(3,9,673)	(1,1,2)	C17	(2,20,1245)	(2,20,1245)	(0,0,0)
C8	(2,10,317)	(2,7,313)	(0,3,4)	C18	(3,26,1083)	(2,26,1083)	(1,0,0)
C9	(4,9,301)	(2,9,301)	(2,0,0)	C19	(1,4,105)	(2,4,107)	(1,0,2)
C10	(2,6,669)	(2,7,670)	(0,1,1)	C20	(2,18,673)	(2,20,675)	(0,2,2)

* The order of number is min, maximum thickness of crack and length of crack.

4.3.3 Limitations and future work

The proposed method performs competently for crack detection, segmentation, and quantification for images with a complex background. It also efficiently decreases the time required to prepare the ground truth for detection. However, the proposed method also has some limitations; Although Faster R-CNN provides cropped crack images with 95% accuracy, the modified TuFF may provide poor results for the remaining 5% of images. It requires hyperparameter tuning based on the environment of the captured images. As the proposed method takes advantage of image processing techniques, the camera should be able to obtain a clear view of the crack. The proposed method is useful for concrete cracks only. Therefore, its applicability for the detection of other crack materials might be limited. The DTM can properly measure cracks with high accuracy when the minimum crack thickness is greater than 1 pixel. The quantification result uses pixel as the unit. In future studies, the algorithm can be tested using a real-world unit, such as mm.

4.4 Conclusion

A crack detection, localization, and quantification method were developed by integrating three different methods (i.e., a Faster R-CNN, a modified TuFF method, and a modified DTM) for application to realistic and practical problems that have various complex backgrounds in different environmental conditions. The Faster-R-CNN-based method provided a bounding box level of crack detection, and the bounding boxes were inputted into the modified TuFF for crack segmentation at the pixel level. The segmented cracks were then processed by the modified DTM to measure their thicknesses and lengths. To realize this, the following new technical contributions were achieved:

1) The limitations of the independent approaches (the Faster R-CNN and the TuFF) were removed; the Faster R-CNN could not detect cracks at the pixel level, but used bounding boxes to enclose the cracked area, whereas the original TuFF could detect cracks at the pixel level but was vulnerable to the effects of the light, shadows, and various noises.
2) The performance of the original TuFF was improved by integrating CLAHE; the original TuFF used a smoothing function for neuron tubular detection, but the neuron tubular shapes and crack shapes were different. The smoothing function was improved by using CLAHE for the proposed method of concrete crack segmentation.
3) The proposed method segmented the concrete cracks at the pixel level very well across varied and complex backgrounds and environmental conditions. The AP of the Faster R-CNN was 95%, and the mIoU of the modified TuFF with CLAHE was 83%.
4) The proposed hybrid method overcame the limitations of traditional deep-learning-only-based crack segmentation methods: (1) it reduced the cost of building a training dataset for crack segmentation because the hybrid method uses Faster R-CNN, which requires only training data based on bounding boxes on images, and (2) it outperformed DeepCrack and Mask R-CNN in segmenting cracks on complex backgrounds.
5) The original DTM method was improved to measure the thickness and length of the segmented cracks. It measured the thickness of the cracks with an RMS error of 2.6 pixels, providing a crack length accuracy of 93%.

Chapter 5. Efficient Attention-based Deep Encoder and Decoder for Automatic Crack Segmentation

Chapter 5 is reprinted from Kang, D. and Cha, Y.L.

Chapter 5 has been reprinted from Kang, D. and Cha, Y.J., 2021. *Efficient attention-based deep encoder and decoder for automatic crack segmentation*. Accepted. Structural health monitoring. Impact factor: 5.929. This chapter has been reproduced with permission from the copyright holder SAGE publishing.

Abstract: Recently, crack segmentation studies have been investigated using deep convolutional neural networks. However, significant deficiencies remain in the preparation of ground truth data, consideration of complex scenes, development of an object-specific network for crack segmentation, and use of an evaluation method, among other issues. In this paper, a novel semantic transformer representation network (STRNet) is developed for crack segmentation at the pixel level in complex scenes in a real-time manner. STRNet is composed of a squeeze and excitation attention-based encoder, a multi head attention-based decoder , coarse upsampling, a focal-Tversky loss function, and a learnable swish activation function to design the network concisely by keeping its fast-processing speed. A method for evaluating the level of complexity of image scenes was also proposed. The proposed network is trained with 1,203 images with further extensive synthesis-based augmentation, and it is investigated with 545 testing images (1,280 × 720, 1,024 × 512); it achieves 91.7%, 92.7%, 92.2%, and 92.6% in terms of precision, recall, F1 score, and mIoU (mean intersection over union), respectively. Its performance is compared with those of recently developed advanced networks (Attention U-net, CrackSegNet, Deeplab V3+, FPHBN, and Unet++), with STRNet showing the best performance in the evaluation metrics-it achieves the fastest processing at 49.2 frames per second.

5.1 Introduction

Deep learning-based approaches were introduced to overcome the limitations of the traditional image processing based damage detection approaches in recent years. Cha et al., (2017) proposed the detection of structural damage using deep a convolutional neural network (CNN). They designed a unique CNN, and it was trained and tested to detect concrete cracks in the various image conditions that have real and uncontrolled lighting conditions including blurry and shadowed. For practical applications, the network has been examined using the images coming from an unmanned aerial vehicle (UAV) for concrete crack detection (Kang & Cha, 2018). The network adopted a sliding window technique to localize the detected cracks, but this technique requires heavy computational cost, and defining the proper size of the sliding window is another issue by considering camera and lens properties, camera and object distance, size of cracks. Instead of the sliding window approach, faster region-based convolutional neural network (Faster R-CNN) (Ren et al., 2015) were applied for damage detection and localization (Cha et al., 2018; Xue & Li, 2018). This Faster R-CNN proposes various sizes of bounding boxes to detect and localize different sizes of damage. The network uses the same base network for detection and localization; therefore, it is faster than the other types of localization methods (e.g., sliding window technique) and became the mainstream in the deep learning-based multiple types of damage detection problems (Maeda et al., 2018; Beckman et al., 2019; Xie et al., 2019; Lu et al., 2019; Liu et al., 2019c).

Localization of structural damage with bounding boxes is not enough for damage quantification. Specifically, it is too coarse to use bounding boxes or sliding window to measure the thickness and length of detected concrete cracks. U-net (Ronneberger et al., 2015) was applied for pixel-level crack segmentation (Zhang et al., 2017). However, this method was only applied to pure asphalt surfaces without any complex objects or background scenes. There are numerous similar studies in this crack segmentation problem. From extensive literature reviews, we recognized four major limitations or disadvantages of existing studies that should be overcome or improved:

1) Although monitoring pavements without considering complex scenes may not constitute a serious problem, detecting structural damage such as concrete cracks is a major limitation if the network cannot detect only cracks in the complex scenes since many structures are located within various different visual scenes. Many researchers worldwide have conducted pixel-level detection

of cracks and reported results as shown in Table 5-1. Only SDDNet (Choi & Cha, 2019), HBFasterRCNN (Kang et al., 2020), and Resnet150 (Bang et al., 2019) considered cracks in the complex scenes.

Table 5-1. Crack segmentation networks

Author	Complex scenes	Network	Train	Val	Test	F1 Score	mIoU	Test input size	FPS
Bang et al., 2019	Yes	Resnet 150	427	-	100	-	59.7	1920×1080	0.22
Benz et al., 2019	No	Crack NausNet	1303	487	115	82.9	-	512×512	-
Choi & Cha, 2019	Yes	SDDNet	160	-	40	-	84.6	1024×512	36
Dung & Anh, 2019	No	FCN	400	100	100	89.3	-	227×227	13.8
Ji et al., 2020	No	Deeplab v3+	300	50	80	-	73.3	512×512	-
Jiang & Zhang, 2020	No	SSDLite MobileNetV2	1030	-	300	-	-	640×480	24
Kang et al., 2020	Yes	HBFaster RCNN	400	-	100	-	83	512×512	0.3
König et al., 2019	No	Attention_Unet	95	-	60	92.8	-	48×48	-
Liu et al., 2019a	No	Unet	38	19	27	90.0	-	512×512	8
Liu et al., 2019b	No	Deep Crack	300	-	237	86.5	85.9	544×384	10
Liu et al., 2020	No	UNet, ResNet-34	770	257	257	95.75	-	800×600	4
Mei et al., 2020	No	DenseNet	700	100	200	75.4	-	256×256	0.25

Nayyeri et al., 2019	No	RTV	352	-	352	75.0	-	500×400	-
Ni et al., 2019	No	GoogLeNet	65K	-	32K	-	-	224×224	-
Ren et al., 2020	No	Crack SegNet	307	-	102	74.6	59.1	512×512	11
Tong et al., 2020	No	CNN, SVM	5292	-	1764	-	75.8	-	6.1
Yang et al., 2019	No	FPHBN	200(1896)	50	908 (1124)	-	-	480×320	-
Zhang et al., 2017	No	CrackNet	1800	-	200	88.8	-	1024×512	0.34
Zhang et al., 2019a	No	SegNet	-	-	155	79.4	-	256×256	1.42
Zhang et al., 2019b	No	CrackNet-R	300	-	500	91.84	-	1024×512	1.4

2) Another limitation is that most existing studies did not use proper evaluation metrics. Rather, most used accuracy, precision, recall and F1 score as presented in Table 5-1. However, accuracy is not proper for this crack evaluation because the size of the crack is usually too small compared to the background scenes; therefore, it usually provides a high score if the size of the crack is small. The precision and recall do not properly consider false positive and false negative detections and the F1 score can control these with parameter changes. One of the reasonable and accurate evaluation metrics at the moment is mean intersection over union (mIoU), which can consider false positive and negative accurately. Therefore, many studies in the areas of computer vision and deep learning also use mIoU as an evaluation metric and loss function to efficiently train their networks. However, for crack segmentation, only seven studies (Choi & Cha, 2019; Kang et al., 2020; Bang et al., 2019; Ji et al., 2020; Liu et al., 2019b; Ren et al., 2020; Tong et al., 2020) used IoU as an evaluation metric. However, most of the claimed IoU performances should be improved.

3) Most of the existing studies used heavy networks or existing traditional networks that were originally developed for the segmentation of many objects; therefore, these networks need

inherently and unnecessarily heavy computational cost due to their excessive learnable parameters. Therefore, it is impossible for real-time processing with relatively large input images or video frames (e.x.,1000 × 500) that have 30 frames per second (FPS). Fast processing is an important aspect of civil infrastructure monitoring due to its large scale and is required to process many images to inspect large-scale structures. It does not necessarily process in a real-time manner, but it reduces overall monitoring costs and provides fast updates of the structural states. For example, as presented in Table I, DeepCrack used VGG16 as the backbone network. Liu et al., (2019a) used U-net (Ronneberger et al., 2015) architecture for concrete crack detection, Dung et al., (2019) used fully convolution network (FCN) (Long et al., 2015), K*ö*nig et al., (2019) used Attention network (Oktay et al., 2018), Bang et al., (2019) used Resnet (He et al., 2016), Mei et al., (2020) used DenseNet (Huang et al., 2017), (Ji et al., 2020) used DeepLabV3+ (Chen et al., 2018), and reference (Ren et al., 2020) applied SegNet (Badrinarayanan et al., 2017). Among all these networks, only SDDNet could do real-time processing with 36 FPS for 1024 × 512 RGB images.

4) Some studies used a too small number of training and testing data sets with small sizes of input images. This results in the high possibility of overfitting for the specific types of cracks with specific image conditions. For example, reference (Ji et al., 2020) used a total of 84 images of relatively small sizes (i.e., 512 × 512), and SDDNet also used only 40 images for testing with relatively large input image (1024 × 512). Further, most of the studies used very small testing input image sizes which are all below 1000 × 500 except those conducted by reference (Choi & Cha, 2019; Bang et al., 2019; Zhang et al., 2019a). Testing input image of small sizes also has the possibility of overfitting to specific types of cracks. It is also not efficient to monitor large-scale civil infrastructure, and it is also very limited in terms of detecting thin cracks in a relatively long distance of camera and object.

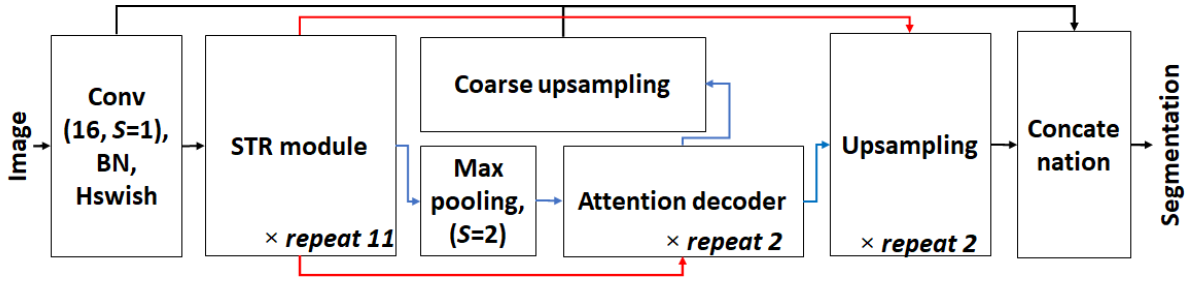
Based on our extensive literature review described above, we propose a new deep encoder and decoder based network with an improved/increased data set and performance to resolve the four issues mentioned above in this pixel-level crack detection problem in complex scenes. In order to realize this, we propose the use of sematic trainable representation network (STRNet) to improve performance in terms of mIoU by keeping the real-time network processing speed for a relatively large size of testing input image frame (1024 × 512) from Tesla V100 GPU. Also, we establish a large ground truth dataset (i.e., 1748 RGB images with sizes of 1024 × 512, 1280 ×

720) for training and testing purposes to consider complex background scenes for robust detection by avoiding overfitting to specific types of cracks and background scenes. We used some publicly available datasets - deep crack (Liu et al., 2019b) and concrete crack segmentation datasets (Özgenel, 2019) - by fixing severe errors. Some ground-truth images of these datasets were coarsely labelled. This can cause poor training results, even when an advanced network is designed and used. Therefore, the images of these existing datasets were re-annotated to reduce annotation errors. To improve the network's performance, we also used focal-Tversky loss function (Abraham & Khan, 2019) and adopted image synthesis techniques to augment the prepared ground truth training data to negate and detect crack-like features on complex scenes.

5.2 Proposed STRNet

An architecture named STRNet of deep convolutional neural network is proposed to segment concrete cracks on complex scenes in pixel-level in a real-time manner (i.e., at least 30 FPS) with a testing input size of 1024 × 512 RGB images/videos. The STRNet is composed of a new STR module-based encoder, an Attention decoder with coarse upsampling block, a traditional convolutional (Conv) operator, a learnable Swish nonlinear activation function (Ramachandran et al., 2017), and batch normalization (BN) to segment only cracks in complex scenes with real-time manner. The schematic view of the STRNet is shown in Figure 5-1. In order to develop this high-performance network with low computational cost, many advanced networks were investigated to figure out their strengths and weaknesses.

Figure 5-1. The overall architecture of STRNet (Kang & Cha, 2021)

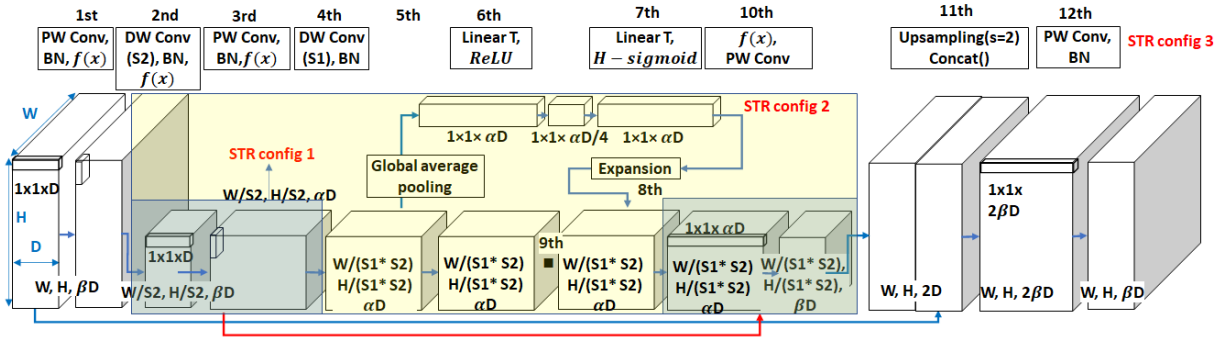


STRNet processes an input image by 16 Conv filters with a size of 3 × 3 × 3 with stride (S) 1, BN (Ioffe & Szegedy, 2015) and Hswish (Avenash & Viswanath, 2019; Howard et al., 2019) activation function with a *skipped connection*. The result of these processes in the first block of Figure 5-1 is inputted to a newly designed STR module and final 'Concatenation block' as shown in Figure 5-1. The STR module is repeated 11 times, and afterward, the feature map is fed into '*Maxpooling*' operator and is then forwarded to the newly designed 'Attention decoder' and '*Upsampling*' module. The result of Max pooling goes through the 'Attention decoder' two times, and the output is fed into '*Upsampling*' and '*Coarse upsampling*' modules. The outputs of the final upsampling and coarse upsampling modules are concatenated with the output of the first Conv block as shown in Figure 5-1. The concatenated features are processed by pointwise convolution (PW) to match the output to the input image size for final pixel-level segmentation. The details of the developed modules and their roles are described in the following subsections.

5.2.1 STR module

The STR module is newly developed in this paper to improve the segmentation accuracy by reducing the computational cost for real-time processing on the complex scenes. The STR module is composed of pointwise convolution (*PW*), depthwise convolution (*DW*), *BN*, *Swish* activation function, squeeze and extension-based attention module as shown in Figure 5-2. STR module has three different configurations (i.e., "*STR_config 1*", "*STR_config 2*" and "*STR_config 3*") as shown in Figure 5-2. STR_config 1 has simple processes of 3rd block, 4th block, and 10th block with *PW*, *DW*, *BN*, and rectified linear unit (*ReLU*) activation function, illustrated as the dark greenish block shown in Figure 5-2. *STR_config 2* is combined with *STR_config 1* and squeeze and excitation-based attention (SEA) module with ReLU illustrated as the yellowish block shown in Figure 5-2. *STR_config 3* is the entire network of the STR module with blocks from 1st to 11th. STR module is repeated 11 times, and different configuration is operated in each repeat as presented in Table 5-2.

Figure 5-2. The detail structure of STR module (Kang & Cha, 2021)



All these arrays of configurations are new and unique with different DW convolution sizes, different stride sizes (S_1, S_2), with/without SEA module, *ReLU / Swish* activation function, and skipped connection. The "*Con.*" in Table 5-2 indicates the skipped connection with the red arrow line as shown in Figure 5-2, which only happens with "*US*" and "*AD*" which stand for upsamping and attention decoder, respectively. Therefore, the Connector is only used in the repeats in 3 and 11 to keep multi-level features. Publicly available segmentation networks usually apply stride 16 or 32 to the feature map in an encoder module, which means that the extracted feature map size is reduced to 16 or 32 times smaller than the original input image size. However, these large spatial contractions of the extracted feature maps compared to the input size may cause the loss of important features. This issue is found throughout our extensive experimental studies to develop this unique network, although it might be only applicable to this unique crack segmentation problem. Due to the nature of cracks with very long and thin shapes, a network may need a slightly larger feature map. Therefore, we applied stride 8 (i.e., S_1^3), since we have three "2" in Table 5-2, but this small stride causes the high computational cost through deep hidden layers of the proposed network.

Table 5-2. Detailed hyperparameter for STR module

STR module iteration								
Repeat #	DW	α	β	S_1	S_2	Connector	$f(x)$	config
1	3×3×1	1D	1D	2	1	no	ReLU	2
2	3×3×1	4.5D	1D	1	1	no	ReLU	1
3	3×3×1	5.5D	1.5D	1	1	yes (US)	ReLU	1
4	5×5×1	6D	2.5D	2	1	no	Swish	2
5	5×5×1	15D	2.5D	1	2	no	Swish	3

6	5×5×1	15D	2.5D	1	2	no	Swish	3
7	5×5×1	7.5D	3D	1	1	no	Swish	2
8	5×5×1	9D	3D	1	2	no	Swish	3
9	5×5×1	18D	6D	2	1	no	Swish	2
10	5×5×1	36D	6D	1	2	no	Swish	3
11	5×5×1	36D	6D	1	2	yes (AD)	Swish	3

To resolve this issue, and to maintain important features and real-time processing, we use different STR_configuration (i.e., configs 1 to 3) in each repeat as presented in Table 5-2. Through the *STR_configs 1 and 2*, we extracted features by keeping its relatively large feature map, but these large feature maps require large computational costs compared to small feature maps through the deep layers of the network. Therefore, to reduce its feature map by keeping the important features, we used *STR_config 3* with squeeze and excitation-based attention operation.

The role of squeeze and excitation operation is to extract important features. In order to squeeze the extracted feature map, global average pooling at the 5th block is applied in *STR_configs 2 and 3*. The global average pooling performs the average pooling operation entire W (input width) and H (input height) size in each feature channel, so the output feature map becomes 1×1×αD at the 6th block. The physical meaning of this global average pooling is the extraction of important (i.e., mean) features from the extracted features. Here, α is given in Table 5-2, and D is 16 since we conducted traditional Conv 16 times, as shown in Figure 5-1. This process is called squeeze process, and it extracts important features while compressing information. This feature is fed into two linear functions (Linear T) (Paszke et al., 2017). with *ReLU* and *H-Sigmoid* (Courbariaux & David, 2015; Howard et al., 2019).

$$H - Sigmoid(x) = \frac{ReLu6(x+3)}{6}, \tag{5-1}$$

where *ReLu6* is an embedded activation function in Pytorch. *ReLu6* has a unique shape with a maximum output value 6 for all inputs greater than or equal to 6. The excitation process recovers the squeezed feature map to the original size by reproduction of the squeezed feature map (1 × 1

× αD). The *H-Sigmoid* expressed in (5-1) provides the bi-linearity activation function. The output of *DW* from 4th block is multiplied (■) by the output of excitation at 8th block.

The role of *STRconfig3* restores an important feature map using a skip connection illustrated with a thick blue line at the bottom of Figure 5-2. It reduces computational cost compared with the two other configurations of the STR module, which can be validated by the following equations. The *PW* and *DW* are formulated as follows:

$$PW\_Conv(w,x)(i,j) = \sum_{c}^{C} x_{i,j,c} w_c, \tag{5-2}$$

$$DW\_Conv(w,x)(i,j) = \sum_{u,v}^{k,k} x_{c,i+u,j+v} w_{u,v}, \tag{5-3}$$

where w and x are weight and input, respectively. *i*, *j*, and *c* are 2D coordinates and the input channel number, respectively. *u* and *v* are the width dimension and height dimension of the input, respectively. The computational costs of *DW Conv* and *PW Conv* can be calculated using Equation (5-4) and (5-5).

$$PW\_Conv = CHWO, \tag{5-4}$$

$$DW\_Conv = CHWk^2, \tag{5-5}$$

$$DW\_Conv + PW\_Conv = CHW(k^2 + O). \tag{5-6}$$

where *C*, *H*, *W*, *k*, and *O* are the number of input channels, the height dimension of input, the width dimension of input, the filter size, and the output channel size, respectively. The approximate number of calculations of the *STRconfig1* is the summation of two *PW Conv* and one *DW Conv* as expressed in Equation (5-7).

$$STR\_config1 = PW_1 + DW_1 + PW_1, \tag{5-7}$$

$$= C_1 H_1 W_1 (k^2 + 2O_1).$$

where the subscript number (i.e., 1) stands for the config number (i.e., 1). The *STR_config2* has considerably a small number of calculations than the others. Therefore, the approximate number of calculations of the *STR_config3* is expressed as Equation (5-8).

$$\begin{aligned} &ST\text{R\_config3} \\ &= (PW_2 + DW_2 + PW_2) + (PW_3 + DW_3 + PW_3), \\ &= C_2 H_2 W_2 (k^2 + 2O_2) + C_3 H_3 W_3 (k^2 + 2O_3), \\ &= \frac{C_1 H_1 W_1 (k^2 + 2O_1 - 1)}{6} + \frac{C_1 H_1 W_1 (k^2 + 2O_1)}{4}, \\ &= \frac{C_1 H_1 W_1 (5k^2 + 10O_1 - 2)}{12}. \end{aligned} \tag{5-8}$$

where $C_1 = 6C_2 = C_3$, $H_1 = H_1 = 2H_3$, $W_1 = W_2 = 2W_3$, and $O_1 = O_2 = O_3$. The discrepancy between Equation (5-7) and Equation (5-8) is expressed as Equation (5-9). The Equation (5-9) is clearly a positive value, which means that the number of calculation of the STR_config3 is smaller than that of the *STRconfig1*, which contributes the real-time processing of the STRNet.

$$STR\_config1 - STR\_config3 = \frac{7k^2 + 14O_1 + 2}{12}. \tag{5-9}$$

Another technical contribution of this STR module is the implementation of a non-linear activation function. Most recently, proposed networks in this area typically only use $ReLU$ because of its simplicity in differential calculation for backpropagation and to reduce computational cost and automatic hibernation of unnecessary learnable parameters in the network. However, our objective is to develop a concise and efficient network by using a smaller number of hidden layers, meaning most of the assigned learnable parameters in each filter in each layer should be fully used to extract multiple levels of features for high performance of the pixel-level segmentation.

Therefore, using $ReLU$ is no longer a viable option for this concise and light objective specific network. We only used this $ReLU$ for the first three STR module repetitions for the stable training process as presented in Figure 5-2. After that, we used a learnable Swish nonlinear activation function Ramachandran et al., 2017 to resolve this issue in the STR module.

$$Swish(x) = x \cdot sigmoid(\gamma x). \tag{5-10}$$

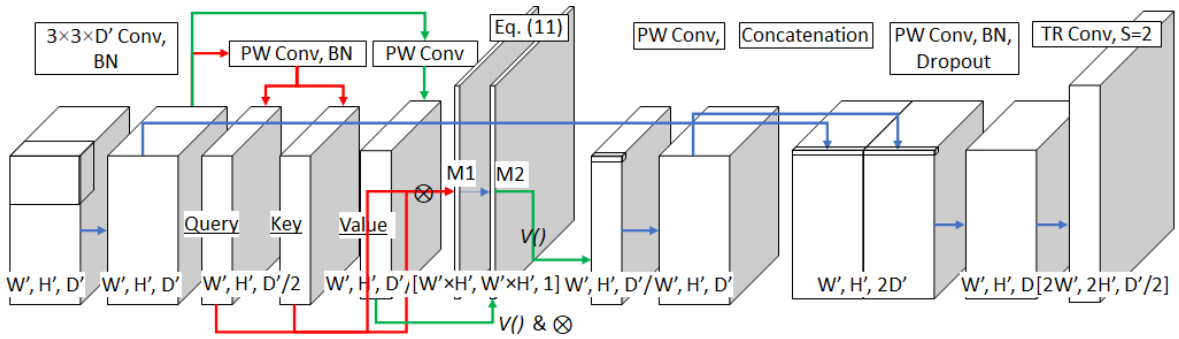
where γ is a learnable parameter of the Swish activation function. The major benefit of this learnable Swish activation function is that it can be converted from scaled linear $ReLU$ to a non-linear function by changing the γ from 0 to ∞. Due to the dynamic shape of the activation function, this network is able to extract features more efficiently and precisely. However, it also may cause an unstable training process; therefore, as we described, the first three repetitions of the STR module use $ReLU$. The result of PW convolution in the 10th block in Figure 5-2 is upsampled to the input feature size (1st block) of W and H. The input of the STR module and the upsampled result are densely concatenated to keep the different multi-level levels of features in the 11th block. This process recovers the loss of features from the 2 strides S_2 of DW convolution in the 2nd block. After, the densely piled features are processed by PW convolution to restore the D channel value, which serves to facilitate the repetition of the STR module.

5.2.2 Attention decoder

The role of traditional decoders in this pixel-level segmentation problem is to recover the size of the extracted feature map from well-designed encoders. However, the performance of the encoders is not usually high enough to achieve a very high level of segmentation as we previously discussed in the Introduction section. Therefore, in this paper, we developed a unique attention-based decoder to support the role of the STR encoder to screen wrongly extracted features in the encoding process. Initially, we used existing attention decoders (Vaswani et al., 2017; Yuan & Wang, 2018), but due to their heavy computational cost, real-time processing was impossible. Therefore, we designed a unique decoder by configuration of Attention decoder, Upsampling and Coarse

upsampling by using the attention operation minimally to reduce the heavy computational cost to keep its real-time processing performance as shown in Figure 5-3.

Figure 5-3. Newly designed attention decoder (Kang & Cha, 2021)



The role of `Attention decoder' shown in Figure 5-3 is to screen wrongly extracted features from the STR encoder and to recover the reduced feature spatial size from STR module by keeping its unique features from the original input size. Usually, an attention decoder is repeated more than 4 times in publicly available networks (Oktay et al., 2018; Zeng et al., 2019). However, we only repeated it two times to reduce computational cost, and we used Upsampling and Coarse upsampling operators to supplement this reduced number of attention decoder repeat as shown in Figure 5-3.

In Figure 5-3, the first input size ($[W', H', D'] = [64,32,96]$) is the final output of the encoder with the result of 2 × 2 max pooling. This input is applied to 3 × 3 convolution and BN. This result is processed by PW with/without BN and produces $Query$ $[\frac{D'}{2}, W', H']$, Key $[\frac{D'}{2}, W', H']$ and $Value$ $[\frac{D'}{2}, W', H']$. These maps are then reshaped using embedded function View ($V()$)of Pytorch from 3-D to 2-D is resulted in $[W^{'} \times H^{'}, \frac{D^{'}}{2}]$, $[\frac{D^{'}}{2}, W^{'} \times H^{'}]$, and $[W^{'} \times H^{'}, \frac{D^{'}}{2}]$, respectively. The $Query$ and Key are multiplied (symbolized as $\otimes$) and result in M1 attention map. The M1 attention map is filtered by Equation (5-3) and output M2. The reshaped $Value$ is multiplied with the M2 attention map which is attention process.

$$\mathrm{M2} = \mathrm{softmax}(\frac{M1}{\sqrt{D'}}). \tag{5-11}$$

The object context produced by attention process and the output of first Conv operation from the first block of the overall architecture of the STRNet as shown in Figure 5-1 are concatenated as shown in Figure 5-3. *PW Conv* condenses this concatenated feature map, and dropout is applied to prevent overfitting. Finally, the transposed convolution restores the semantic mask (Dumoulin and Visin, 2016).

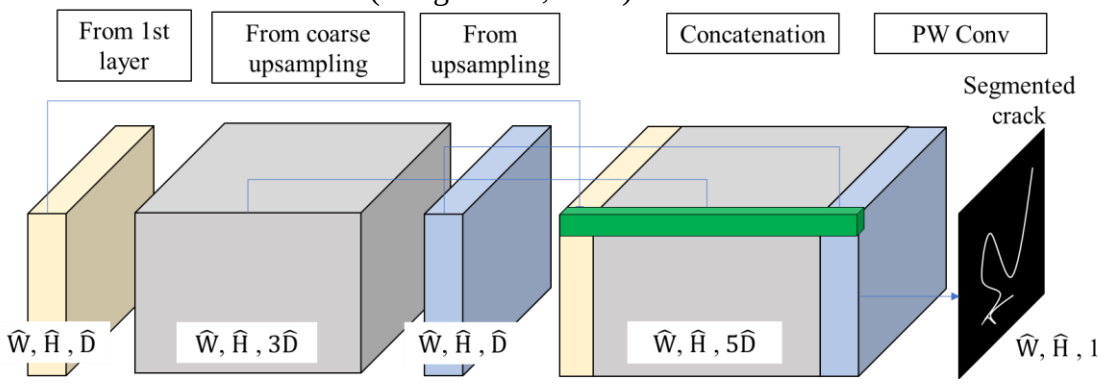
5.2.3 Upsampling and coarse upsampling

The Upsampling layer is intended to double the dimensions of input, and it is commonly used in any segmentation network (Long et al., 2015; Ronneberger et al., 2015; Chen et al., 2018). The input feature passes the bilinear upsampling. Bilinear upsampling increases width and height two times. After that, the 3×3 convolution, *BN*, and *ReLU* activation function are performed to reduce the depth of the map. The size of upsampling output follows the size of original input image.

5.2.4 Concatenation block

Skip connection or simple bilinear upsampling has been widely used for encoder and decoder-based networks (Chen et al., 2018, Oktay et al., 2018) to keep multi-level features. We also use the multiple skip connections to obtain better segmentation as shown in Figure 5-1.

Figure 5-4. Concatenation block (Kang & Cha, 2021)



As shown in Figure 5-4. we concatenate the results of the traditional Conv block, Attention decoder with Coarse upsampling, and Upsampling. The $\widehat{W}$, $\widehat{H}$ and $\widehat{D}$ are 1024, 512, and 16, respectively.

The concatenated feature map is processed by PW convolution to have the same depth size compared to a binary ground truth.

5.3 Established Data Bank

To train the developed STRNet for crack segmentation on various complex scenes, we prepared ground truth data from various sources. A total of 1784 images sized 1024 × 512 and 1280 × 720 were prepared. Some (612) of them came from existing available datasets. The raw images of these existing datasets were re-annotated to reduce annotation errors (Liu et al., 2019b; Özgenel, 2019). Some (300) of them came from our previous studies (Choi & Cha, 2019; Kang et al., 2020), and new datasets (836) from various structures and locations was established. The detailed information of the developed datasets is presented in Table 5-3. To minimize the time and effort to prepare training image data, we took advantage of using our previous SDDNet (Choi & Cha 2020). The raw images were initially processed by this network and the output errors such as false positives and false negatives were fixed manually.

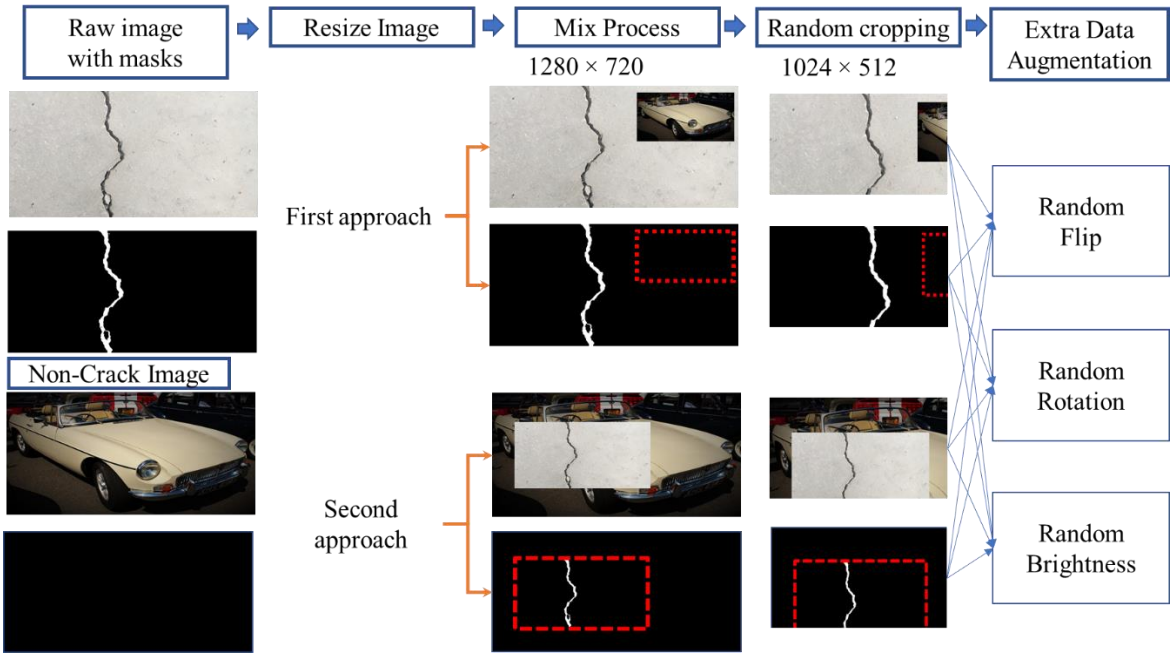
Table 5-3. Developed datasets for training and testing

	Training	Testing	Total
Size	1,024 × 512	1,280 × 720, 1,024 × 512	1,280 × 720, 1,024 × 512
Number of images	1,203	545	1,748
Number of augmented images	12,030		

5.3.1 Data augmentation

The prepared ground truth data presented in Table 5-3 is not enough to achieve high performance segmentation which can negate the detection of any crack-like features on the complex scenes. Therefore, traditional data augmentation skills such as random rotation and random cropping were conducted. Moreover, synthesis techniques of ground truth images to generate cracks on complex scenes were also applied by inserting an object of interest into another non-target image with complex scenes that would allow us to achieve a robust classifier. Figure 5-5 shows two approaches to generating the procedure and synthetic images.

Figure 5-5. Two image synthesis approaches for training data generation (Kang & Cha, 2021)



The first approach is that the image with cracks is set as a background image, and a non-target image having complex scenes but without cracks is inserted in the background image as shown in Figure 5-5. The second approach is vice versa. After, the synthesized images are further processed with random flipping, rotation, and brightness operations, and they are resized to 1024 × 512. The complex non-target images without cracks are collected from Open Image Dataset v4 (Kuznetsova et al., 2018). We used 1203 images from 99,999 images. In order to crop the area having crack pixels in ground truth images, the "*CropNonEmptyMaskIfExists*" function from Albumentation (Buslaev et al., 2020) was used, and the cropped crack area was patched to a non-target complex background image as shown in Figure 5-5. The cropped crack image size is randomly selected from 300 × 204 to 400 × 512, and the location of insertion is also randomly selected. Therefore, the eventual total number of augmented images for training is 12,030 as presented in Table 3.

5.3.2 Complexity of the proposed dataset

Considering complex background scenes of structural damage in the real world is critical, as mentioned in the Introduction section. However, the evaluation of these scenes' complexity levels

can be subjective if no quantitative evaluation method is used. For this reason, we put forward an algorithm for evaluating the complexity of an image dataset. The fundamental concept of the complexity check evaluation algorithm is to count the number of objects in a scene. The higher the object number, the greater the complexity level.

To count the number of objects in an image, we used Felzenszwalb's graph segmentation method (Felzenszwalb & Huttenlocher, 2004). Felzenszwalb's algorithm verifies the relationship between pixels and edges in an image. For example, if an edge appears between two pixels, these two pixels are assumed to be located in different clusters (i.e., objects), C_n, as expressed in Equation (5-12). If no edge appears between pixels, then they are assumed to be located in the same object.

$$GraphSeg(p_i, p_j) = \begin{cases} if = (p_i \le e_n \le p_j) \\ else = (p_i p_j) \subset C_n \end{cases}, \quad (5\text{-}12)$$

where p_i and p_j are the input pixels, and e_n is the pre-defined edge value, which should be defined by a user. In this study, we defined this value as constituted by color and intensity. C_n is the object cluster, that is, the cluster of pixels, as expressed in Equation (5-13).

$$C_n = (p_0, \dots, p_n), \quad (5\text{-}13)$$

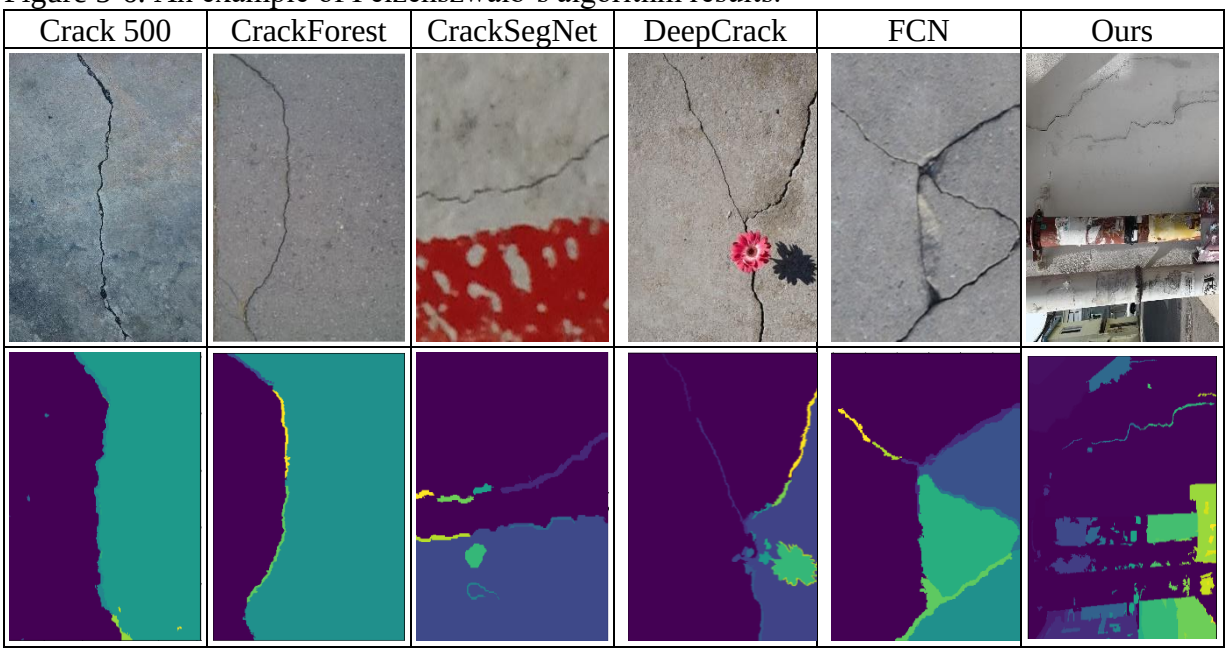
C_n is too fine a level of object segmentation without consideration of image noise. Therefore, a smooth function was adopted, as expressed in Equation (5-14).

$$Felzenszwalb(C_i, C_j) = \begin{cases} True = if \;\; Min(C_i - C_j) < Max(C_i(p_i) - C_i(p_j)) \\ False = otherwise \end{cases}, \quad (5\text{-}14)$$

where $Min\ (C_i - C_j)$ expresses the minimum pixel value difference between two clusters via pixel-to-pixel comparisons of cases. $Max(C_i(p_i) - C_i(p_j)$ denotes the maximum pixel value differences

within the same cluster that should be tuned. If the calculated minimum value is lower than the calculated maximum value, the two clusters are merged. On the basis of this rule, the algorithm assigns all pixels and generates an object segmentation map. As shown in Figure 5-6, Felzenszwalb's algorithm produces an object segmentation map and identifies the boundary of objects in images obtained from our dataset and existing crack datasets (i.e., Crack 500, CrackForest, CrackSegNet, DeepCrack, and FCN).

Figure 5-6. An example of Felzenszwalb's algorithm results.



Using the results of Felzenszwalb's algorithm, we assigned an object number to each object within an image. To calculate the complexity score (i.e., the level of complexity), the number of unique numbers for each image was determined. To evaluate the level of complexity of the available crack datasets, all the images were evaluated, and the average was calculated for each dataset. The results are presented in Table 5-4. Crack 500 had a slightly higher complexity score than those generated by the other datasets (except ours), even though it comprised only pure pavement surface images. This result is attributed to the fact that most of the images were asphalt surface images with high asperity due to the ingredients of coarse pavement material. Nevertheless, our dataset showed the highest complexity score.

Table 5-4. Comparison of complexity scores

	Author	Complexity score	# of images
Crack500	Zhang et al., 2016	27	494
CrackForest	Shi et al., 2016	3.57	118
CrackSegNet	Ren et al., 2020	6.69	813
DeepCrack	Liu et al., 2019b	11.4	527
FCN	Dung, 2019	7.3	776
Ours		41.23	1748

5.4 Training details

This section describes the details of the training process and hardware. Python programing language (Python, 2020) with Pytorch 1.6 deep learning library (Paszke et al., 2017) was used to code the STRNet. The STRNet was trained in a graphic processing unit (GPU) equipped workstation. The workstation specifications are Intel Core i7-6850K CPU, Titan XP GPU, and 128GB RAM. To train our models, we set up the 4 Titan XP GPU using Nvidia Apex distributed data parallel (DDP) training library. The input image size is 1024 × 512, which is randomly cropped if the image size is bigger than the input size. The use of proper loss function is crucial; therefore, we investigated several recently developed functions such as cross entropy loss, dice cross entropy loss, and mIoU. Eventually, focal-Tversky loss function was used for training. The focal-Tversky loss was used as a combination of the loss function (Abraham & Khan, 2019) as follows,

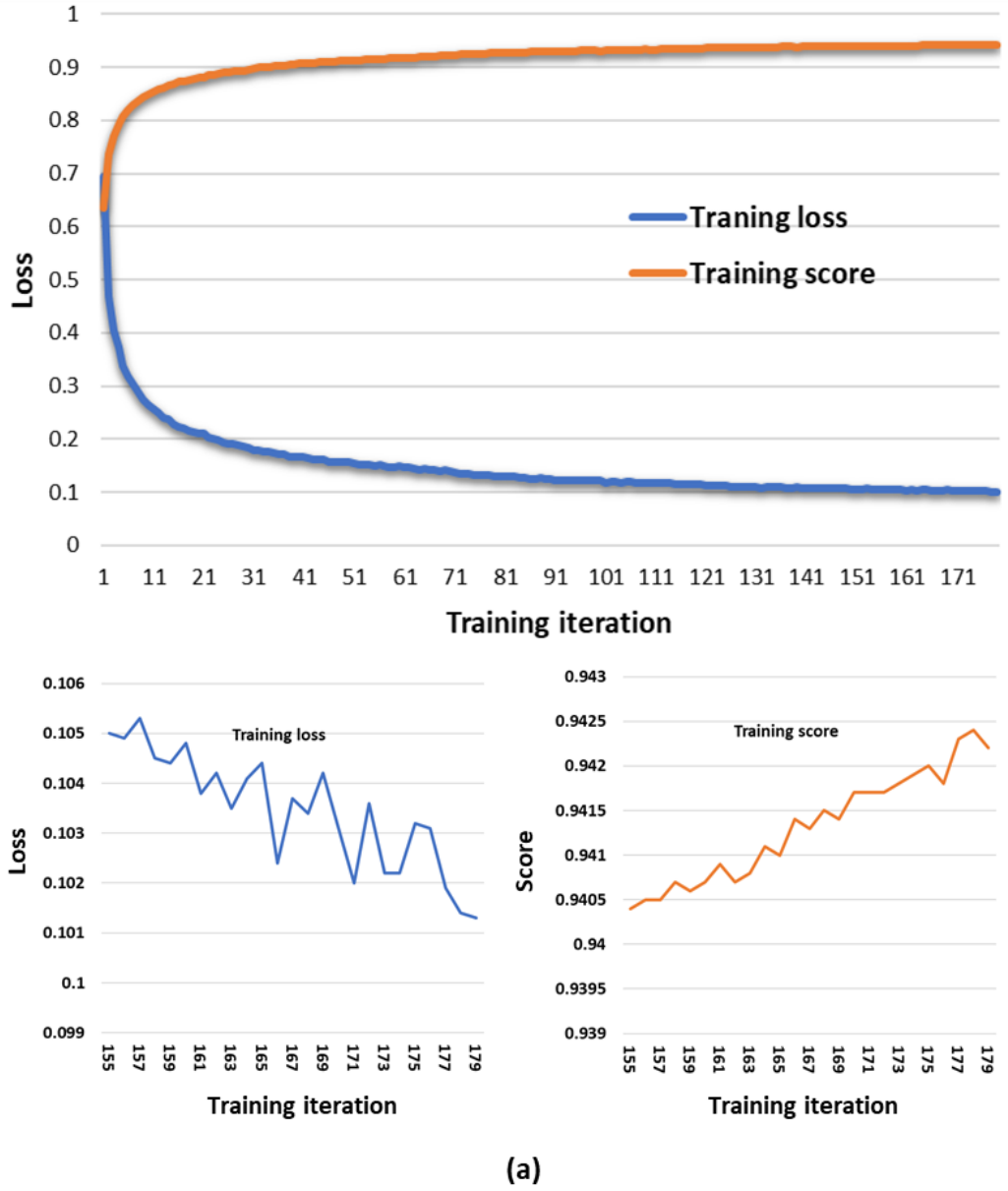
$$TL = \frac{TP + s}{TP + FP * a + FN * b + s}, \tag{5-14}$$

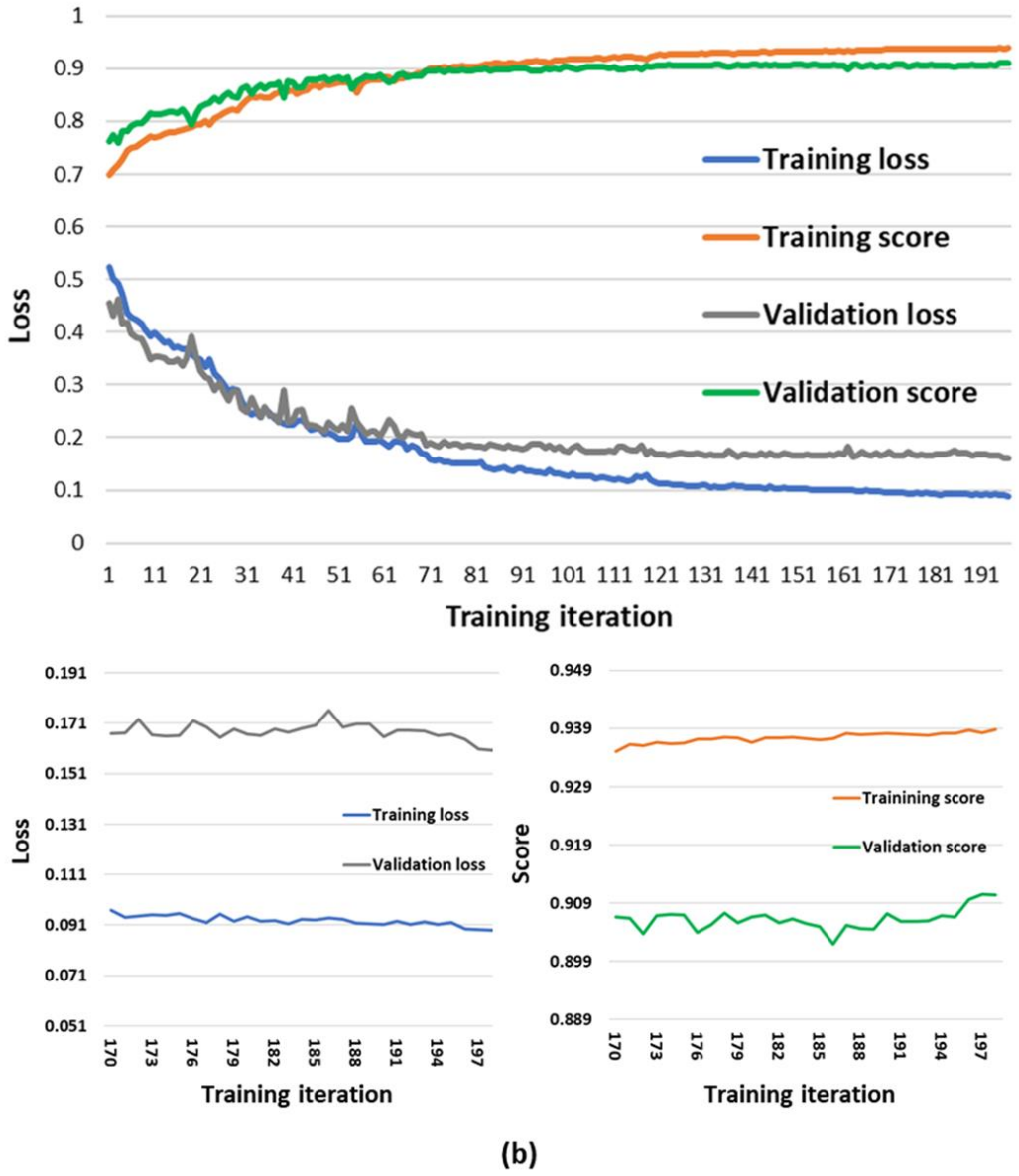
$$Focal - Tversky\ loss\ = (1 - TL)^c, \tag{5-15}$$

where TL is Tversky loss. TP, FP, and FN are true positive, false positive, and false negative, respectively. a, b, c, and s are all hyperparameters. Based on trial and error, a, b, c, and s are defined as 0.5, 0.5, 1.3, and 1.0, respectively. Abraham and Khan, (2019) investigated the performance of this focal-Tversky loss function in the segmentation problem and showed that it

outperformed to get balance between precision (FP) and recall (FN) compared to the dice loss function.

Figure 5-7. Focal Tverskey training loss and score via epoch iteration, (a) Hold-out validation, (b) T rain-valid-test splits validation (Kang & Cha, 2021)





(b)

In order to do backpropagation for the learnable parameter updating, the Adam optimizer was employed (Kingma & Ba 2014). The hyperparameters such as first moment, second moment, and dropout rate were defined as 0.9, 0.999 and 0.2, respectively. The initial learning rate was 0.005, and dropped by 20% when the number of epochs were 30, 70, and 120, to keep a stable training process. To reduce the training time, a DDP with batch size 8 was also used for four GPUs. The progress of the focal-Tversky loss through training epoch iteration is plotted in Figure 5-7. As clearly demonstrated in Figure 5-7, the focal-Tversky loss is successfully minimized, and training score also became almost 0.94. As shown in the figure, we conducted two types of training and

validation processes: hold-out validation and train-valid-test split validation. For the hold-out validation, we divided the total dataset into training and testing sets, as tabulated in Table 5-3, and conducted training and testing as the validation, which is plotted in Figure 5-7 (a). For the train-valid-test split validation, a 10% validation dataset (170 images) of the total images (1,748 images) was randomly selected from the training dataset (1,203 images), and training and validation losses and scores were plotted during the training iterations, as shown in Figure 5-7 (b). In these two validation processes, there was only a small discrepancy between the training score (93.8) and validation score (91.0), see Figure 5-7 (b). This means that the training set is not developed/determined to achieve a high performance from the specific testing and validation datasets, because the training score (93.8) is slightly higher than the va idation score (91.0) and the claimed testing score (92.6).

5.5 Case studies

The developed STRNet was extensively experimentally investigated. In section V-A, some parametric studies were carried out to find effective image synthesis technique, loss function, activation function, and effective decoder. In section 5.5, the eventual STRNet based on the parametric studies was tested on many complex scenes to segment concrete cracks. In section 5, extensive comparative studies were conducted in the same training and testing datasets with the same conditions of loss function for fair evaluation.

5.5.1 Parametric studies of STRNet

We conducted parametric studies to find the most effective parameters and architecture of STRNet. In order to train and test the developed network, the training and testing data presented in Table 5-3 were used. All data augmentation techniques described in Section 5.2.1 were also applied. The used evaluation metrics are:

$$Precision = \frac{TP}{TP + FP} \tag{5-16}$$

$$Recall = \frac{TP}{TP + FN} \tag{5-17}$$

$$F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5-18}$$

$$mIoU = mean(\frac{TP}{TP + FP + FN}) \tag{5-19}$$

The first study was for the method of image synthesis to overcome the limitation of prepared ground truth datasets. We compared two different image synthesis techniques described in Section 5.3.1 and the second image synthesis method showed better performances as presented in Table 5-5. This resulted in a 1.6% improvement. Two different loss functions for effective training of the STRNet were tested. The general IoU loss function, which is the most popular loss function in this field, and the focal-Tversky loss function were compared. The focal-Tversky loss function showed better performance, with a 6.7% improvement of mIoU.

Table 5-5. Parametric studies for STRNet

	Precision	Recall	F-1 score	mIoU
Without image synthesis	89.9%	90.8%	90.4%	91.0%
IoU loss function	81.0%	87.5%	84.1%	85.9%
$FT\ loss$ function	91.7%	92.7%	92.2%	92.6%
Without coarse upsampling	90.3%	92.0%	91.1%	91.6%
Without attention in decoder	89.9%	89.0%	89.5%	90.2%

At this experimental test, the image synthesis was applied for both cases. We used the coarse upsampling technique in STRNet and tested the effectiveness. The coarse upsampling method improved the mIoU by approximately 1%. Another unique technique in this STRNet was the attention decoder. The effectiveness of the attention decoder was also investigated, which showed that it improved the mIoU by approximately 2.4%. With these parametric studies, we decided the eventual network of the STRNet with training methods such as image augmentation and loss function.

In order to check any possibility of overfitting and underfitting problems, k -fold random validations for the fully trained network were conducted. In each validation, the 10% (170) of the validation sets were randomly selected from training, testing and total dataset, respectively, and conducted experiments to calculate mIoU using the fully trained network. All the mIoU from the total 30 number of validation datasets from the training, testing, and total datasets are expressed

as “Train”, “Test”, and “Total”, respectively in Table 5-6. The average mIoUs from the three different datatsets are 93.93%, 92.59%, and 93.15%, respectively. Our claimed maximum performance of the STRNet was 92.6% as shown in Table 5-5. Therefore, the obtained validation results are quite close to the final performance. Through these total 30-fold validation processes with total 5,100 images, we assume that our trained STRNet is not underfitted and overfitted.

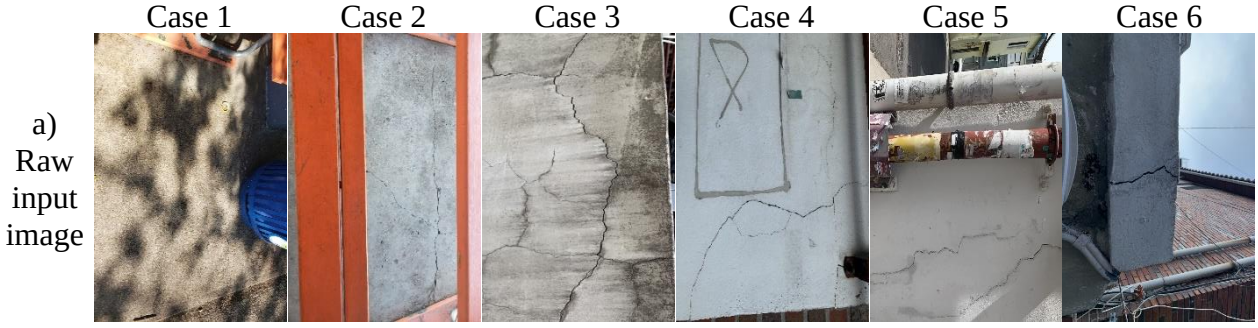
Table 5-6. Random validation through 10-fold random selection

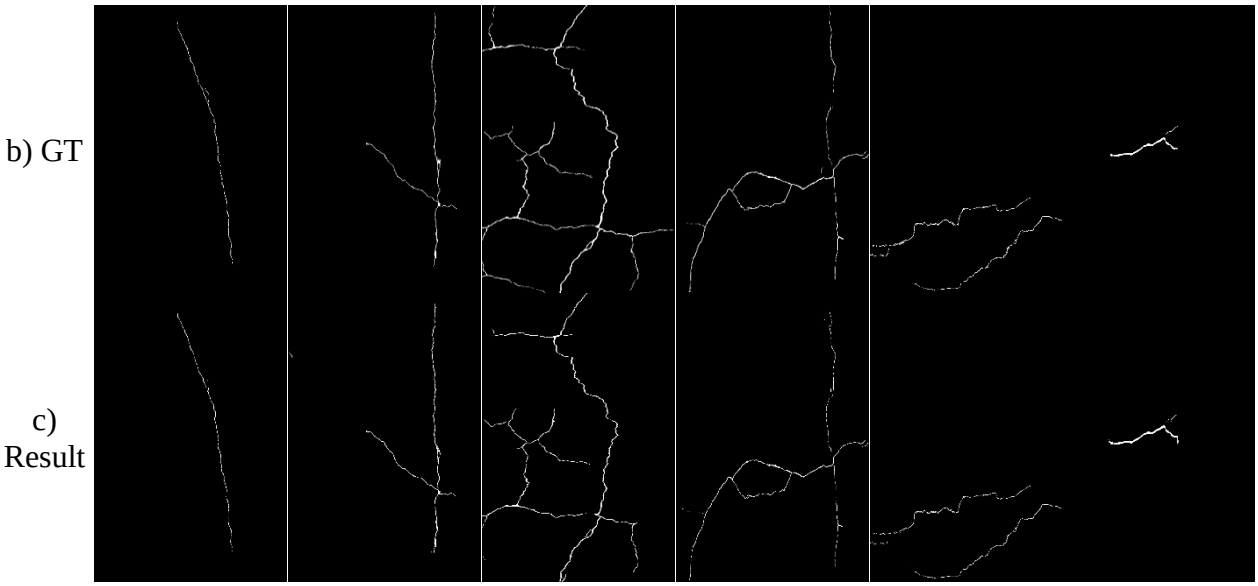
	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Avg(%)
Train	93.58	94.27	94.27	93.9	93.8	94.4	94.37	93.5	94.2	92.9	93.93
Test	92.4	92.41	92.0	92.6	92.6	92.5	93.04	92.7	92.51	93.14	92.59
Total	93.25	93.2	93.44	93.35	93.15	93.3	93.46	93.0	92.34	92.98	93.15

5.5.2 Experimental testing of STRNet

In this section, the eventual parameters and module from the experimental studies in Section 5.5.1 was selected as the final STRNet. This STRNet showed a maximum 92.6% mIoU on 545 images having complex scenes with 49.2 FPS using single V100 GPU for 1024×512 input images. This is much faster than required speed (i.e., 30 FPS) for real-time processing. It provides very stable performance without unbalance among false positives and false negatives based on 91.7% precision and 92.7% recall evaluation metrics including 92.2% F1 score. The reported mIoU 92.6% is considered to be a very high level of accuracy since all the ground truth (GT) data has a minimum level of annotation error because there are many unclear cases that a pixel is included in a crack or intact concrete surface. Therefore, it seems that a maximum of 5% error is unavoidable in ground truth data.

Figure 5-8. Examples of STRNet results on various complex scenes (Kang & Cha, 2021)





5.5.3 Comparative studies

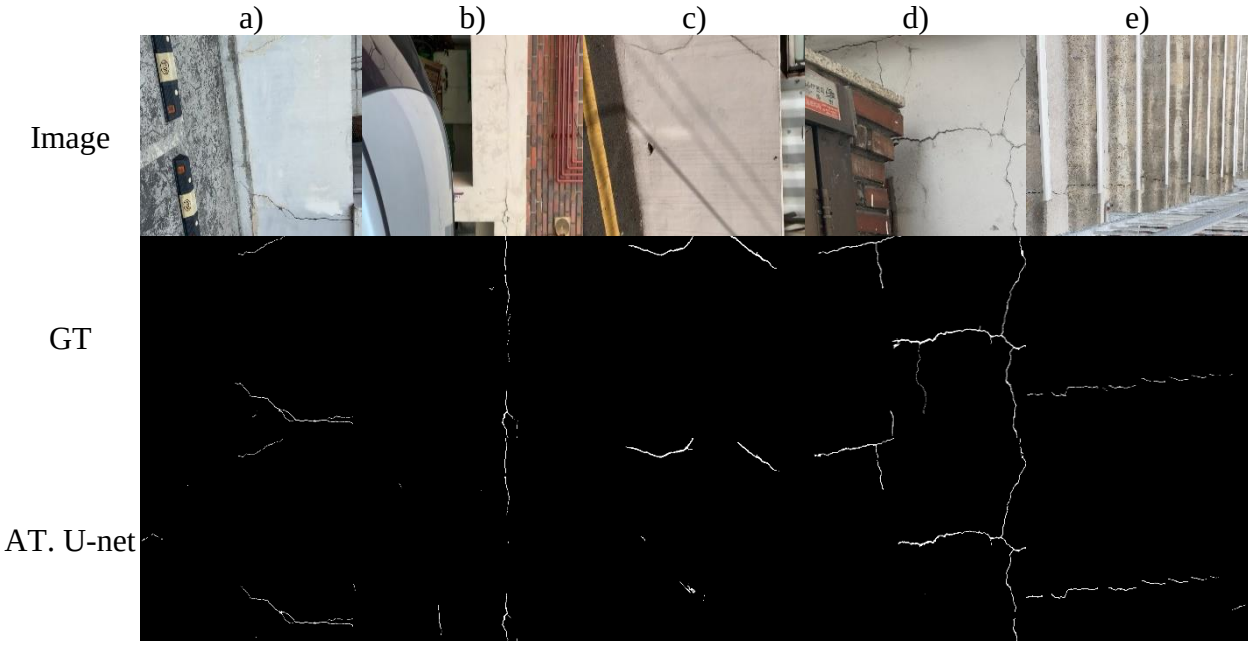
Extensive comparative studies were conducted to show the superior performances of the proposed STRNet compared to the traditional networks. The selected networks are attention U-net (König et al., 2019), Deeplab v3+ (Ji et al., 2020), Unet++ (Zhou et al., 2019), FPHBN (Rent et al., 2020) and CrackSegNet (Yang et al., 2019). All these advanced networks are recently developed and showed state of the art performances in this segmentation area and applied them to the crack segmentation problem.

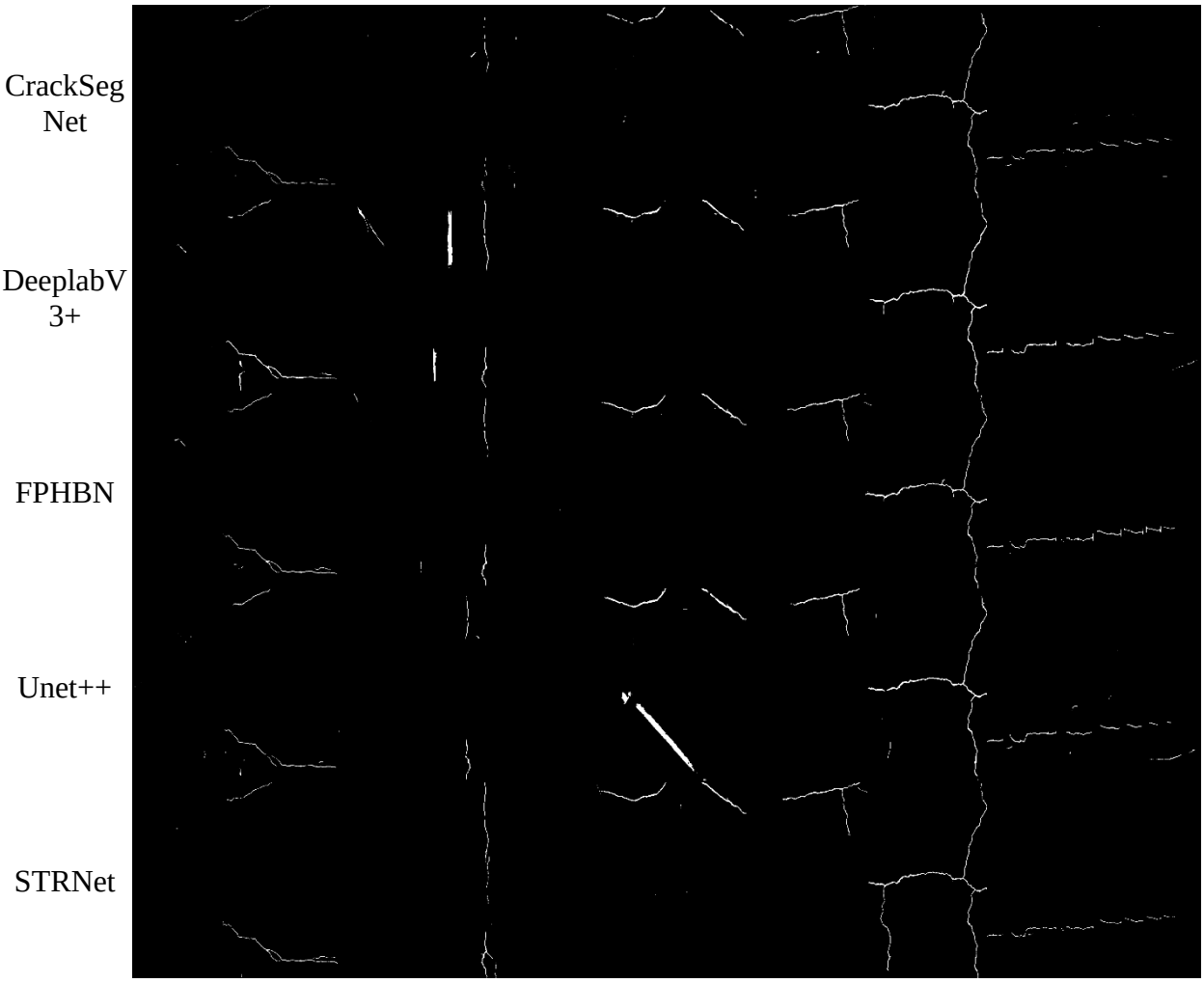
Table 5-7. Results of experimental comparative studies

Model	Precision (%)	Recall (%)	F1 Score (%)	mIoU (%)	V100 (FPS)	# of param
Attention U-net	85.63	91.22	88.33	89.1	17	34M
CrackSegNet	86.33	84.89	85.61	87.1	21.4	12.4M
DeeplabV3+	77.37	83.6	80.36	83.24	30.2	59M
FPHBN	86.33	84.89	85.61	87.1	34.0	5.9M
UNet++	82.9	85.4	84.13	85.9	30.6	26.9M
STRNet	91.7	92.7	92.2	**92.6**	49.2	2M

Each of these four selected networks were trained using the same training dataset, data augmentation techniques, and hyperparameters, including *FT* loss function for fair comparison. All of these well-trained networks were also tested by the same 545 testing images presented in Table 5-3. The experimental results are tabulated in Table 5-7. It showed that the proposed STRNet still demonstrated the best performances in terms of precision, recall, F1 score, and mIoU with the fastest processing with 49.2 FPS using single V100 GPU. The number of learnable parameters of the STRNet is smallest, which has the advantage of being operable in embedded microcomputing devices. This is beneficial for real structural applications and commercialization. The attention Unet, DeeplabV3+ and Unet++ showed unbalanced precision and recall scores, which means that these networks involve problems with false positive or false negative detections. In order to compare the performances, the complex scene images in different locations and structures with different lighting conditions are selected and processed by six networks as shown in Figure. 5-9. The proposed STRNet showed superior performance in the selected images. Deeplab V3+ showed the worst performance, with an approximately 9% lower mIoU than that of STRNet. Deeplab V3+ also showed very weak performance in negating dark areas to be detected as cracks. Attention Unet (i.e., AT. U-net) and Unet++ have problems negating shadowed areas. FPHBN achieved balanced false positive and false negative detections but still has issues with false positive and false negative detections, as shown in Figure. 5-9 (a) and (d).

Figure 5-9. Example results of the comparative studies (Kang & Cha, 2021)





5.6 Conclusion

In this paper, a novel STRNet, which is a deep convolutional neural network, is developed for concrete crack segmentation in pixel level. The developed network was trained using large training data set and tested on 545 images. The performances of the proposed network in terms of precision, recall, F1 score and mIoU are 91.7%, 92.7%, 92.2%, 92.6%, respectively, with 49.2 FPS using V100 GPU which is able to process relatively large input images (1280 × 720, 1024 × 512) with real-time manner. From the extensive comparative studies, this demonstrated the best performance in terms of the upper four evaluation criteria. New technical contributions of this paper are:

1) A new deep convolutional neural network was designed to be able to do real-time processing using relatively large input images (1280 × 720, 1024 × 512) with 49.2 FPS.

2) The proposed network showed state of the art performance in segmentation of cracks with 92.6% mIoU.
3) The STRNet has the lightest network size among the compared networks with a 2M memory size, which offers the great benefit of being applicable to real world problems using a microcomputer.
4) The network was able to segment cracks on highly complex scenes including different area, structures, and lighting conditions.
5) An image complexity evaluation method was proposed, and our training and testing datasets showed the highest level of complexity among the examined datasets.
6) The new encoder named as the STR module was developed to extract multi-level features effectively.
7) The new decoder with the attention module was developed to support the STR encoder by screening wrongly extracted features from the encoder to improve the segmentation accuracy (i.e., 2.4% mIoU).
8) Coarse upsampling was adopted for this crack segmentation problem. It improved the 1% mIoU.
9) The new loss function (Focal-Tversky loss function) was adopted to train the newly designed network to improve the crack segmentation performance (i.e., 6.7% mIoU).
10) Many training and testing data with large image sizes were established to conduct extensive evaluations (see Table 5-3).
11) The prepared ground truth data were drastically reduced in annotation errors compared to the publicly available crack segmentation data.
12) A new image synthesis technique was adopted to augment the ground truth training data to improve the network performance (i.e., 1% mIoU).
13) A learnable swish activation was adopted to improve the segmentation performance by keeping a concise network which enables faster than real-time processing. This may give us the possibility to increase the testing input size image.

The performance of STRNet accomplished outstanding performance on the given testing and training datasets, but a larger dataset will be required to monitor the many varying types of structures together using a single trained network. However, this problem can be resolved by grouping the structures, such as bridges, buildings, dams, etc. Then, depending on the specific

group, the user can collect data and train the network. The trained network can be installed beneath a reinforced concrete bridge deck or girders with a vision sensor and microcomputer as an example of a real structure application. The mixed precision training strategy must test for faster speed.

Chapter 6. Conclusion

This chapter summarizes the achievements of this research and its technical contributions and discusses possible future work.

6.1 Summary and conclusion

This thesis explored autonomous UAV applications and computer vision for SHM purposes. I initiated autonomous flights of the UAVs to detect structural damage using computer vision to overcome existing traditional approaches such as visual inspection and other physics-based models and data-driven approaches. The technical and non-technical contributions of this study are as follows:

In Chapter 3, I introduced a new autonomous flight method of UAVs for SHM in GPS-denied areas:

- Autonomous flight method of the UAVs was developed using PID controller.
- A new 3-D pseudo map was proposed using a USB system to replace GPS signals for autonomous flight of the UAV in GPS-denied areas.
- Concrete cracks were detected by deep CNN by processing the video collected by autonomous UAV system.
- The detected cracks were localized by the geo-tagging technique using the location of the UAV within the 3-D pseudo map.
- 96.6% accuracy, 91.9% sensitivity, and 97.9% specificity achieved by the algorithm.
- The integration of the UBS-based UAV and the CNN-based detection algorithm was suggested and tested in the SHM field for the first time.

In Chapter 4, I introduced a new hybrid pixel-level crack segmentation method by integrating image processing and deep learning in complex scenes:

- A hybrid pixel-level crack segmentation method was developed by the integration of modified TuFF and Faster R-CNN to consider complex background scenes.
- Original TuFF was modified to be applied to the crack segmentation problem.
- The issue of the data preparation cost was partially resolved by removing the requirement of pixel-level labelling of ground truth data, which is very time-consuming.

- Crack quantification method was applied. This can calculate the number of pixels to eventually calculate the length and width of segmented cracks by the hybrid method.
- 2.3 RMSE and 93% accuracy achieved with the help of the modified DTM and the skeletonization algorithm.

In Chapter 5, I introduced an advanced deep learning method to segment cracks on complex scenes in a real-time manner:

- STRNet was developed by carefully designing the entire network with advanced attend operator, STR module, attention decoder, and Focal-Tversky loss function.
- To increase the number of training data, a new data augmentation method was added to synthesize ground truth pixel-level labelled image with a complex-scene image.
- Stride is an important hyperparameter factor in small objects such as concrete cracks. An STR module was developed to effectively control this value between speed and accuracy.
- The STRNet demonstrated superior performance compared to other published algorithms with 92.6% mIoU on high resolution images (1024×512) with 49.2 FPS.

6.2 Future work

There are some unsolved problems in this topic, and these should be investigated in future works. The following are some recommendations for future works:

- The autonomous UAV algorithm should be implemented in large-scale infrastructures to consider actual environmental effects such as temperature, wind, and humidity.
- For the damage detection algorithm, various types of damages should be investigated.
- Obstacle avoidance method for autonomous flight of the UAV should be developed.
- The remote charging system and the developed autonomous damage inspection system can be applied for the remote SHM task.
- The automatic 3D damage map should be developed for better visualization and understanding of structure status.

References

[1]. American Society of Civil Engineers, 2021, Report card for America's infrastructure. https://infrastructurereportcard.org/ available.

[2]. Abdel-Qader, I., Abudayyeh, O. and Kelly, M.E., 2003. Analysis of edge-detection techniques for crack identification in bridges. Journal of Computing in Civil Engineering, 17(4), pp.255-263.

[3]. Abraham, N. and Khan, N.M., 2019, April. A novel focal tversky loss function with improved attention u-net for lesion segmentation. In 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019) (pp. 683-687). IEEE.

[4]. Adhikari, R.S., Moselhi, O. and Bagchi, A., 2014. Image-based retrieval of concrete crack properties for bridge inspection. Automation in construction, 39, pp.180-194.

[5]. Alam, S.Y., Loukili, A., Grondin, F. and Rozière, E., 2015. Use of the digital image correlation and acoustic emission technique to study the effect of structural size on cracking of reinforced concrete. Engineering Fracture Mechanics, 143, pp.17-31.

[6]. Ali, R. and Cha, Y.J., 2019. Subsurface damage detection of a steel bridge using deep learning and uncooled micro-bolometer. Construction and Building Materials, 226, pp.376-387.

[7]. Ardupilot (2017), Available at: https://github.com/ArduPilot/ardupilot.git, accessed March 2017.

[8]. Avenash, R. and Viswanath, P., 2019. Semantic Segmentation of Satellite Images using a Modified CNN with Hard-Swish Activation Function. In VISIGRAPP (4: VISAPP) (pp. 413-420).

[9]. Badrinarayanan, V., Kendall, A. and Cipolla, R., 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), pp.2481-2495.

[10]. Bang, S., Park, S., Kim, H. and Kim, H., 2019. Encoder–decoder network for pixel-level road crack detection in black-box images. Computer-Aided Civil and Infrastructure Engineering, 34(8), pp.713-727.

[11]. Beckman, G.H., Polyzois, D. and Cha, Y.J., 2019. Deep learning-based automatic volumetric damage quantification using depth camera. Automation in Construction, 99, pp.114-124.

[12]. Benz, C., Debus, P., Ha, H.K. and Rodehorst, V., 2019, December. Crack segmentation on UAS-based imagery using transfer learning. In 2019 International Conference on Image and Vision Computing New Zealand (IVCNZ) (pp. 1-6). IEEE.

[13]. Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M. and Kalinin, A.A., 2020. Albumentations: fast and flexible image augmentations. Information, 11(2), p.125.

[14]. Carvalho, J.P., Jucá, M.A., Menezes, A., Olivi, L.R., Marcato, A.L.M. and dos Santos, A.B., 2017. Autonomous UAV outdoor flight controlled by an embedded system using Odroid and ROS. In CONTROLO 2016 (pp. 423-437). Springer, Cham.

[15]. Cha, Y.J., Choi, W. and Büyüköztürk, O., 2017. Deep learning-based crack damage detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering, 32(5), pp.361-378.

[16]. Cha, Y.J., Choi, W., Suh, G., Mahmoudkhani, S. and Büyüköztürk, O., 2018. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. Computer-Aided Civil and Infrastructure Engineering, 33(9), pp.731-747.

[17]. Cha, Y.J., You, K. and Choi, W., 2016. Vision-based detection of loosened bolts using the Hough transform and support vector machines. Automation in Construction, 71, pp.181-188.

[18]. Chen, J.G., Wadhwa, N., Cha, Y.J., Durand, F., Freeman, W.T. and Buyukozturk, O., 2015. Modal identification of simple structures with high-speed video using motion magnification. Journal of Sound and Vibration, 345, pp.58-71.

[19]. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European conference on computer vision (ECCV) (pp. 801-818).

[20]. Cheng, J., Xiong, W., Chen, W., Gu, Y. and Li, Y., 2018, October. Pixel-level crack detection using U-Net. In TENCON 2018-2018 IEEE Region 10 Conference (pp. 0462-0466). IEEE.

[21]. Choi, W. and Cha, Y.J., 2019. SDDNet: Real-time crack segmentation. IEEE Transactions on Industrial Electronics, 67(9), pp.8016-8025.

[22]. Courbariaux, M., Bengio, Y. and David, J.P., 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. arXiv preprint arXiv:1511.00363.

[23]. Das, S., Saikia, J., Das, S. and Goni, N., 2015. A comparative study of different noise filtering Techniques in digital images. International Journal of Engineering Research and General Science, 3(5), pp.180-190.

[24]. Darby, P., Hollerman, W. and Miller, J., 2019. Exploring the Potential Utility of Unmanned Aerial Vehicles for Practical Bridge Inspection in Louisiana. In MATEC Web of Conferences (Vol. 271, p. 01001). EDP Sciences.

[25]. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L., 2009, June. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

[26]. Deutscher, J., Blake, A. and Reid, I., 2000, June. Articulated body motion capture by annealed particle filtering. In Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662) (Vol. 2, pp. 126-133). IEEE.

[27]. Díaz, E., Pérez, M.C., Gualda, D., Villadangos, J.M., Ureña, J. and García, J.J., 2017, June. Ultrasonic indoor positioning for smart environments: A mobile application. In 2017 4th Experiment@ International Conference (exp. at'17) (pp. 280-285). IEEE.

[28]. Dung, C.V., 2019. Autonomous concrete crack detection using deep fully convolutional neural network. Automation in Construction, 99, pp.52-58.

[29]. Dumoulin, V. and Visin, F., 2016. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285.

[30]. Escalona, U., Arce, F., Zamora, E. and Sossa, H., 2019. Fully convolutional networks for automatic pavement crack segmentation. Computación y Sistemas, 23(2), pp.451-460.

[31]. Eschmann, C., Kuo, C.M., Kuo, C.H. and Boller, C., 2012. Unmanned aircraft systems for remote building inspection and monitoring.

[32]. Eisenbach, M., Stricker, R., Seichter, D., Amende, K., Debes, K., Sesselmann, M., Ebersbach, D., Stoeckert, U. and Gross, H.M., 2017, May. How to get pavement distress

detection ready for deep learning? A systematic approach. In 2017 international joint conference on neural networks (IJCNN) (pp. 2039-2047). IEEE.

[33]. Felzenszwalb, P.F. and Huttenlocher, D.P., 2004. Efficient graph-based image segmentation. International journal of computer vision, 59(2), pp.167-181.

[34]. Feng, D., Feng, M.Q., Ozer, E. and Fukuda, Y., 2015. A vision-based sensor for noncontact structural displacement measurement. Sensors, 15(7), pp.16557-16575.

[35]. Frangi, A.F., Niessen, W.J., Vincken, K.L. and Viergever, M.A., 1998, October. Multiscale vessel enhancement filtering. In International conference on medical image computing and computer-assisted intervention (pp. 130-137). Springer, Berlin, Heidelberg.

[36]. Gillins, M.N., Gillins, D.T. and Parrish, C., 2016. Cost-effective bridge safety inspections using unmanned aircraft systems (UAS). In Geotechnical and structural engineering congress 2016 (pp. 1931-1940).

[37]. Girshick, R. 2015. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, 1440-1448.

[38]. Hallermann, N. and Morgenthal, G., 2013, September. Unmanned aerial vehicles (UAV) for the assessment of existing structures. In IABSE Symposium Report (Vol. 101, No. 14, pp. 1-8). International Association for Bridge and Structural Engineering.

[39]. Morgenthal, G. and Hallermann, N., 2014. Quality assessment of unmanned aerial vehicle (UAV) based visual inspection of structures. Advances in Structural Engineering, 17(3), pp.289-302.

[40]. Hao, M., Lu, C., Wang, G. and Wang, W., 2017. An improved neuron segmentation model for crack detection–image segmentation model. Cybernetics and Information Technologies, 17(2), pp.119-133.

[41]. He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[42]. He, K., Gkioxari, G., Dollár, P. and Girshick, R., 2017. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[43]. Hess, W., Kohler, D., Rapp, H. and Andor, D., 2016, May. Real-time loop closure in 2D LIDAR SLAM. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1271-1278). IEEE.

[44]. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. and Le, Q.V., 2019. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1314-1324).

[45]. Honegger, D., Meier, L., Tanskanen, P. and Pollefeys, M., 2013, May. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In 2013 IEEE International Conference on Robotics and Automation (pp. 1736-1741). IEEE.

[46]. Hu, J., Shen, L. and Sun, G., 2018. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[47]. Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).

[48]. Hutchinson, T.C. and Chen, Z., 2006. Improved image analysis for evaluating concrete damage. Journal of Computing in Civil Engineering, 20(3), pp.210-216.

[49]. Ioffe, S., & Szegedy, C. 2015, Batch normalization: Accelerating deep network training by reducing internal covariate shift, In International Conference on Machine Learning, 448-456.

[50]. Jahanshahi, M.R., Masri, S.F., Padgett, C.W. and Sukhatme, G.S., 2013. An innovative methodology for detection and quantification of cracks through incorporation of depth perception. Machine vision and applications, 24(2), pp.227-241.

[51]. Ji, A., Xue, X., Wang, Y., Luo, X. and Xue, W. 2020, An integrated approach to automatic pixel-level crack detection and quantification of asphalt pavement. Automation in Construction, 114, 103176.

[52]. Jiang, S. and Zhang, J., 2020. Real-time crack assessment using deep neural networks with wall-climbing unmanned aerial system. Computer-Aided Civil and Infrastructure Engineering, 35(6), pp.549-564.

[53]. Kang, D. and Cha, Y.J., 2018. Autonomous UAVs for structural health monitoring using deep learning and an ultrasonic beacon system with geo-tagging. Computer-Aided Civil and Infrastructure Engineering, 33(10), pp.885-902.

[54]. Kang, D., Benipal, S.S., Gopal, D.L. and Cha, Y.J., 2020. Hybrid pixel-level concrete crack segmentation and quantification across complex backgrounds using deep learning. Automation in Construction, 118, p.103291.

[55]. Kee, S.H. and Zhu, J., 2013. Using piezoelectric sensors for ultrasonic pulse velocity measurements in concrete. Smart Materials and Structures, 22(11), p.115016.

[56]. Keys, R., 1981. Cubic convolution interpolation for digital image processing. IEEE transactions on acoustics, speech, and signal processing, 29(6), pp.1153-1160.

[57]. Kingma, D.P. and Ba, J., 2014. Adam: a method for stochastic optimization. arXiv. arXiv preprint arXiv:1412.6980, 22.

[58]. König, J., Jenkins, M.D., Barrie, P., Mannion, M. and Morison, G., 2019, September. A convolutional neural network for pavement surface crack segmentation using residual connections and attention gating. In 2019 IEEE International Conference on Image Processing (ICIP) (pp. 1460-1464). IEEE.

[59]. Koziarski, M. and Cyganek, B., 2017. Image recognition with deep neural networks in presence of noise–dealing with and taking advantage of distortions. Integrated Computer-Aided Engineering, 24(4), pp.337-349.

[60]. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A. and Duerig, T., 2020. The open images dataset v4. International Journal of Computer Vision, pp.1-26.Lanczos, C., 1958. Linear Systems in Self-Ad Joint Form. The American Mathematical Monthly, 65(9), pp.665-679.

[61]. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278-2324.

[62]. Lee, J.J. and Shinozuka, M., 2006. A vision-based system for remote sensing of bridge displacement. Ndt & E International, 39(5), pp.425-431.

[63]. Lee, T.C., Kashyap, R.L. and Chu, C.N., 1994. Building skeleton models via 3-D medial surface axis thinning algorithms. CVGIP: Graphical Models and Image Processing, 56(6), pp.462-478.

[64]. Li, J., Deng, J. and Xie, W., 2015. Damage detection with streamlined structural health monitoring data. Sensors, 15(4), pp.8832-8851.

[65]. Li, S., Zhao, X. and Zhou, G., 2019. Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. Computer-Aided Civil and Infrastructure Engineering, 34(7), pp.616-634.

[66]. Lim, H., Park, J., Lee, D. and Kim, H.J., 2012. Build your own quadrotor: Open-source projects on unmanned aerial vehicles. IEEE Robotics & Automation Magazine, 19(3), pp.33-45.

[67]. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

[68]. Lin, Y.Z., Nie, Z.H. and Ma, H.W., 2017. Structural damage detection with automatic feature-extraction through deep learning. Computer-Aided Civil and Infrastructure Engineering, 32(12), pp.1025-1046.

[69]. Liu, J., Yang, X., Lau, S., Wang, X., Luo, S., Lee, V.C.S. and Ding, L., 2020. Automated avement crack detection and segmentation based on two-step convolutional neural network. Computer-Aided Civil and Infrastructure Engineering, 35(11), pp.1291-1305.

[70]. Liu, Z., Cao, Y., Wang, Y. and Wang, W. 2019a, Computer vision-based concrete crack detection using U-net fully convolutional networks. Automation in Construction, 104, 129-139.

[71]. Liu, Y., Yao, J., Lu, X., Xie, R. and Li, L. 2019b, DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. Neurocomputing, 338, 139-153.

[72]. Liu, C., Tang, L. and Liu, J., 2019c. A stacked autoencoder with sparse Bayesian regression for end-point prediction problems in steelmaking process. IEEE Transactions on Automation Science and Engineering, 17(2), pp.550-561.

[73]. Long, J., Shelhamer, E. and Darrell, T. 2015, Fully convolutional networks for semantic segmentation. IEEE conference on computer vision and pattern recognition, 3431-3440.

[74]. Lu, B., Yu, X.B., Lai, J.W., Huang, K.C., Chan, K.C. and Chu, H.K., 2019. A Learning Approach for Suture Thread Detection with Feature Enhancement and Segmentation for 3-D Shape Reconstruction. IEEE Transactions on Automation Science and Engineering, 17(2), pp.858-870.

[75]. MathWorks, MATLAB 2018a, available at: <https://www.mathworks.com/products/matlab.html/>, (Accessed 1 Feb 2019).

[76]. Marvelmind Robotics, 2017, Available at: https://marvelmind.com, accessed March 2017

[77]. Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T. and Omata, H., 2018. Road damage detection and classification using deep neural networks with smartphone images. Computer-Aided Civil and Infrastructure Engineering, 33(12), pp.1127-1141.

[78]. McCrea, A., Chamberlain, D. and Navon, R., 2002. Automated inspection and restoration of steel bridges—a critical review of methods and enabling technologies. Automation in Construction, 11(4), pp.351-373.

[79]. Mei, Q., Gül, M. and Azim, M.R. 2020, Densely connected deep neural network considering connectivity of pixels for automatic crack detection. Automation in Construction, 110, 103018.

[80]. Meier, L., Tanskanen, P., Fraundorfer, F., & Pollefeys, M. 2011, Pixhawk: A system for autonomous flight using onboard computer vision. In Robotics and automation (ICRA), 2011 IEEE international conference, 2992-2997.

[81]. Metni, N. and Hamel, T., 2007. A UAV for bridge inspection: Visual servoing control law with orientation limits. Automation in construction, 17(1), pp.3-10.

[82]. Meng, L., Li, L., & Veres, S. M. 2010, Aerodynamic parameter estimation of an unmanned aerial vehicle based on extended kalman filter and its higher order approach, In Advanced Computer Control (ICACC), 2010 2nd International Conference, 5, 526-531.

[83]. Minorsky, N., 1922. Directional stability of automatically steered bodies. Journal of the American Society for Naval Engineers, 34(2), pp.280-309.

[84]. Milletari, F., Navab, N., Ahmadi, S.A.V. and V-net, Fully convolutional neural networks for volumetric medical image segmentation. The 2016 Fourth International Conference on 3D Vision (3DV) (pp. 565-571).

[85]. Mukherjee, S., Condron, B., & Acton, S. T., 2015. Tubularity flow field—a technique for automatic neuron segmentation. IEEE Transactions on Image Processing, 24(1), 374-389.

[86]. Nagi, J., Ducatelle, F., Di Caro, G.A., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J. and Gambardella, L.M., 2011, November. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA) (pp. 342-347). IEEE.

[87]. Nair, V. and Hinton, G.E., 2010, January. Rectified linear units improve restricted boltzmann machines. In ICML.

[88]. Nayyeri, F., Hou, L., Zhou, J. and Guan, H., 2019. Foreground–background separation technique for crack detection. Computer-Aided Civil and Infrastructure Engineering, 34(6), pp.457-470.

[89]. NHTSA & SAE International (2014), Available at: https://www.sae.org/misc/pdfs/automated_driving.pdf, accessedMarch 2017.

[90]. Ni, F., Zhang, J. and Chen, Z., 2019. Zernike-moment measurement of thin-crack width in images enabled by dual-scale deep learning. Computer-Aided Civil and Infrastructure Engineering, 34(5), pp.367-384.

[91]. Nikon, Nikon d7200, Available from https://www.nikonusa.com/en/nikon-products/product/dslr-cameras/d7200.html> (Accessed 29 April 2019).

[92]. Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B. and Glocker, B., 2018. Attention u-net: Learning where to look for the pancreas. arXiv preprint arXiv:1804.03999.

[93]. Orsag, M., Korpela, C. and Oh, P., 2013. Modeling and control of MM-UAV: Mobile manipulating unmanned aerial vehicle. Journal of Intelligent & Robotic Systems, 69(1), pp.227-240.

[94]. Ortega-Zamorano, F., Jerez, J.M., Gómez, I. and Franco, L., 2017. Layer multiplexing FPGA implementation for deep back-propagation learning. Integrated Computer-Aided Engineering, 24(2), pp.171-185.

[95]. Owen, E., 2007. Minneapolis bridge collapse exposes inspection failures. New Civil Engineer.

[96]. Özgenel, Çağlar, Fırat., 2019, "Concrete Crack Segmentation Dataset", Mendeley Data, v1 http://dx.doi.org/10.17632/jwsn7tfbrp.1

[97]. Pacheco, J., Šavija, B., Schlangen, E. and Polder, R.B., 2014. Assessment of cracks in reinforced concrete by means of electrical resistance and image analysis. Construction and Building Materials, 65, pp.417-426.

[98]. Paglieroni, D.W., 1992. Distance transforms: Properties and machine vision applications. CVGIP: Graphical models and image processing, 54(1), pp.56-74.

[99]. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A., 2017, Automatic differentiation in pytorch, Neural Information Processing Systems (NIPS) Workshop 16.

[100]. Perez-Grau, F.J., Caballero, F., Merino, L. and Viguria, A., 2017, Multi-modal Mapping and Localization of Unmanned Aerial Robots based on Ultra-Wideband and RGB-D sensing

[101]. Proficnc, 2017, Pixhawk v2 Feature Overview, Available at: www.proficnc.com, accessed March 2017

[102]. Rafiei, M. H. & Adeli, H., 2017, A novel machine learning based algorithm to detect damage in highrise building structures, The Structural Design of Tall and Special Buildings, 26, 18, https://doi.org/10.1002/tal.1400.

[103]. Rahman, M.A. and Wang, Y., 2016, December. Optimizing intersection-over-union in deep neural networks for image segmentation. In International symposium on visual computing (pp. 234-244). Springer, Cham.

[104]. Ramachandran, P., Zoph, B. and Le, Q.V., 2017. Searching for activation functions. arXiv preprint arXiv:1710.05941.

[105]. Ramana, L., Choi, W. and Cha, Y.J., 2019. Fully automated vision-based loosened bolt detection using the Viola–Jones algorithm. Structural Health Monitoring, 18(2), pp.422-434.

[106]. Ren, S., He, K., Girshick, R., & Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, 91-99, https://doi.org/10.1109/TPAMI.2016.2577031.

[107]. Ren, Y., Huang, J., Hong, Z., Lu, W., Yin, J., Zou, L. and Shen, X., 2020. Image-based concrete crack detection in tunnels using deep fully convolutional networks. Construction and Building Materials, 234, 117367.

[108]. Reza, A.M., 2004. Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement. Journal of VLSI signal processing systems for signal, image and video technology, 38(1), 35-44, https://doi.org/10.1023/B:VLSI.0000028532.53893.82.

[109]. Ronneberger, O., Fischer, P. and Brox, T., 2015. October. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, 234-241.

[110]. Rukundo, O. and Cao, H., 2012. Nearest neighbor value interpolation. arXiv preprint arXiv:1211.1768.

[111]. Salehi, S.S.M., Erdogmus, D. and Gholipour, A., 2017, September. Tversky loss function for image segmentation using 3D fully convolutional deep networks. In International workshop on machine learning in medical imaging (pp. 379-387). Springer, Cham.

[112]. Sankarasrinivasan, S., Balasubramanian, E., Karthik, K., Chandrasekar, U. and Gupta, R., 2015. Health monitoring of civil structures with integrated UAV and image processing system. Procedia Computer Science, 54, pp.508-515.

[113]. Samsung, Galaxy s9. available at: <https://www.samsung.com/ca/smartphones/galaxy-s9/camera/>, (Accessed 2nd April 2020).

[114]. Scherer, D., M¨ uller, A. & Behnke, S., 2010. Evaluation of pooling operations in convolutional architectures for objectrecognition, in Artificial Neural Networks–ICANN 2010, Springer, 92–101.

[115]. Serif, Affinity Photo. Available at: <https://affinity.serif.com/en-gb/photo/>, (Accessed 7th April 2020).

[116]. Seki, H., Kobayashi, S., Kamiya, Y., Hikizu, M. and Nomura, H., 2000, April. Autonomous/semi-autonomous navigation system of a wheelchair by active ultrasonic beacons. In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065) (Vol. 2, pp. 1366-1371). IEEE.

[117]. Shi, Y., Cui, L., Qi, Z., Meng, F., & Chen, Z., 2016. Automatic road crack detection using random structured forests. IEEE Transactions on Intelligent Transportation Systems, 17(12), 3434-3445.

[118]. Silvagni, M., Tonoli, A., Zenerino, E. and Chiaberge, M., 2017. Multipurpose UAV for search and rescue operations in mountain avalanche events. Geomatics, Natural Hazards and Risk, 8(1), pp.18-33.

[119]. Simonyan, K., & Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv, arXiv:1409.1556v6, available at: <https://arxiv.org/abs/1409.1556> (Accessed 2nd April 2020).

[120]. Smith, G. L., Schmidt, S. F., & McGee, L. A., 1962. Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle, National Aeronautics and Space Administration.

[121]. Smith, P.R., 1981. Bilinear interpolation of digital images. Ultramicroscopy, 6(2), pp.201-204.

[122]. Son, H., Hwang, N., Kim, C. and Kim, C., 2014. Rapid and automated determination of rusted surface areas of a steel bridge for robotic maintenance systems. Automation in Construction, 42, pp.13-24.

[123]. Soukup, D. and Huber-Mörk, R., 2014. December. Convolutional neural networks for steel surface defect detection from photometric stereo images. In International Symposium on Visual Computing (pp. 668-677). Springer, Cham.

[124]. Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014), Dropout: a simple way to prevent neural networks from overfitting, Journal of machine learning research, 15(1), 1929-1958.

[125]. Stempfhuber, W. and Buchholz, M., 2011. A precise, low-cost RTK GNSS system for UAV applications. Proc. of Unmanned Aerial Vehicle in Geomatics, ISPRS.

[126]. Sung, Y., Kwak, J., Jeong, Y.S. and Park, J.H., 2016. Beacon distance measurement method in indoor ubiquitous computing environment. In Advances in Parallel and Distributed Computing and Ubiquitous Services (pp. 125-130). Springer, Singapore.

[127]. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D. Vanhoucke, V. & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, 1-9, https://doi.org/10.1109/CVPR.2015.7298594.

[128]. Teuliere, C., Marchand, E., & Eck, L., 2015, 3-D model-based tracking for UAV indoor localization. IEEE Transactions on cybernetics, 45(5), 869-879.

[129]. Tiemann, J., Schweikowski, F., & Wietfeld, C., 2015. Design of an UWB indoor-positioning system for UAV navigation in GNSS-denied environments, In Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference, 1-7.

[130]. Tong, Z., Yuan, D., Gao, J. and Wang, Z., 2020. Pavement defect detection with fully convolutional network and an uncertainty framework. Computer-Aided Civil and Infrastructure Engineering, 35(8), pp.832-849.

[131]. Tsai, Y. J., Kaul, V., & Yezzi, A., 2013. Automating the crack map detection process for machine operated crack sealer. Automation in Construction, 31, 10-18

[132]. Tzutalin, LabelImg Git code, 2015. <https://github.com/tzutalin/labelImg/>, (Accessed 7th April 2020).

[133]. Valença, J., Gonçalves, L.M.S. and Júlio, E.N.B.S., 2013. Damage assessment on concrete surfaces using multi-spectral image analysis. Construction and Building Materials, 40, pp.971-981.

[134]. Van Rossum, G., 2007, June. Python Programming Language. In USENIX annual technical conference (Vol. 41, p. 36).

[135]. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017, Attention is all you need. In Advances in neural information processing systems, 5998-6008.

[136]. Vossiek, M., Wiebking, L., Gulden, P., Weighardt, J. and Hoffmann, C., 2003, August. Wireless local positioning-concepts, solutions, applications. In Radio and Wireless Conference, 2003. RAWCON'03. Proceedings (pp. 219-224). IEEE.

[137]. Xia, Y., Chen, B., Weng, S., Ni, Y.Q. and Xu, Y.L., 2012. Temperature effect on vibration properties of civil structures: a literature review and case studies. Journal of civil structural health monitoring, 2(1), pp.29-46.

[138]. Xie, Q., Li, D., Xu, J., Yu, Z. and Wang, J., 2019. Automatic detection and classification of sewer defects via hierarchical deep learning. IEEE Transactions on Automation Science and Engineering, 16(4), pp.1836-1847.

[139]. Xue, Y. and Li, Y., 2018. A fast detection method via region-based fully convolutional neural networks for shield tunnel lining defects. Computer-Aided Civil and Infrastructure Engineering, 33(8), pp.638-654.

[140]. Yang, F., Zhang, L., Yu, S., Prokhorov, D., Mei, X. and Ling, H., 2019. Feature pyramid and hierarchical boosting network for pavement crack detection. IEEE Transactions on Intelligent Transportation Systems, 21(4), pp.1525-1535.

[141]. Yang, X., Li, H., Yu, Y., Luo, X., Huang, T. and Yang, X., 2018. Automatic pixel-level crack detection and measurement using fully convolutional network. Computer-Aided Civil and Infrastructure Engineering, 33(12), pp.1090-1109.

[142]. Yuan, Y. and Wang, J. (2018), Ocnet: Object context network for scene parsing. arXiv preprint, arXiv:1809.00916.

[143]. Zeng, Z., Xie, W., Zhang, Y. and Lu, Y., 2019. RIC-Unet: An improved neural network based on Unet for nuclei segmentation in histology images. IEEE Access, 7, 21420-21428.

[144]. Zhang, C. and Elaksher, A., 2012. An unmanned aerial vehicle-based imaging system for 3D measurement of unpaved road surface distresses 1. Computer-Aided Civil and Infrastructure Engineering, 27(2), pp.118-129.

[145]. Zhang, C., Kuhn, M., Merkl, B., Mahfouz, M. and Fathy, A.E., 2006, June. Development of an UWB indoor 3D positioning radar with millimeter accuracy. In 2006 IEEE MTT-S International Microwave Symposium Digest (pp. 106-109). IEEE.

[146]. Zhang, L., Yang, F., Zhang, Y.D. and Zhu, Y.J., 2016, September. Road crack detection using deep convolutional neural network. In 2016 IEEE international conference on image processing (ICIP) (pp. 3708-3712). IEEE.

[147]. Zhang, A., Wang, K.C., Li, B., Yang, E., Dai, X., Peng, Y., Fei, Y., Liu, Y., Li, J.Q. and Chen, C., 2017. Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network. Computer-Aided Civil and Infrastructure Engineering, 32(10), pp.805-819.

[148]. Zhang, J., Lu, C., Wang, J., Wang, L. and Yue, X.G., 2019a. Concrete cracks detection based on FCN with dilated convolution. Applied Sciences, 9(13), p.2686.

[149]. Zhang, A., Wang, K.C., Fei, Y., Liu, Y., Chen, C., Yang, G., Li, J.Q., Yang, E. and Qiu, S., 2019b. Automated pixel-level pavement crack detection on 3D asphalt surfaces with a recurrent neural network. Computer-Aided Civil and Infrastructure Engineering, 34(3), pp.213-229.

[150]. Zhang, X., Rajan, D. and Story, B., 2019c. Concrete crack detection using context-aware deep semantic segmentation network. Computer-Aided Civil and Infrastructure Engineering, 34(11), pp.951-971.

[151]. Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N. and Liang, J., 2019. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. IEEE transactions on medical imaging, 39(6), pp.1856-1867.

[152]. Zhu, Z., German, S. and Brilakis, I., 2011. Visual retrieval of concrete crack properties for automated post-earthquake structural safety evaluation. Automation in Construction, 20(7), pp.874-883.

[153]. Zoidis, N., Tatsis, E., Vlachopoulos, C., Gotzamanis, A., Clausen, J.S., Aggelis, D.G. and Matikas, T.E., 2013. Inspection, evaluation and repair monitoring of cracked concrete floor using NDT methods. Construction and Building Materials, 48, pp.1302-1308.

[154]. Zou, Q., Cao, Y., Li, Q., Mao, Q., & Wang, S., 2012. CrackTree: Automatic crack detection from pavement images. Pattern Recognition Letters, 33(3), 227-238, https://doi.org/10.1016/j.patrec.2011.11.004.

[155]. Zwirello, L., Schipper, T., Harter, M., & Zwick, T., 2012. UWB localization system for indoor applications: Concept, realization and analysis, Journal of Electrical and Computer Engineering, 4.