

AUTONOMOUS VISION-BASED INSPECTION OF RC RAILWAY BRIDGES
FOR RAPID POST-EARTHQUAKE RESPONSE AND RECOVERY

BY

YASUTAKA NARAZAKI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Civil Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Professor Billie Spencer, Chair and Director of Research
Associate Professor Mani Golparvar-Fard
Assistant Professor Girish Chowdhary
Professor Shirley Dyke, Purdue University

ABSTRACT

This research proposes a framework for autonomous vision-based inspection of reinforced concrete (RC) railway bridges to realize rapid post-earthquake response and recovery, which are crucial for minimizing the disruptions to people's lives and business. Currently, after an earthquake, coarse preliminary structural assessment is performed based on the data immediately available (e.g. ground acceleration data). Then, human-based visual inspection is performed to obtain detailed information about the structures with suspected damage. This procedure is often time-consuming and laborious, because the preliminary assessment tends to result in conservative decisions, leaving a significant workload for the subsequent detailed inspection. For example, after the 2011 Tohoku Earthquake, the challenge of the post-earthquake inspection lead to 16-18 hours of operational shutdown of railways in Tokyo metropolitan area, Japan, causing more than 5 million people who were unable to return home. The framework proposed in this research seeks to minimize such negative impact to people's lives through improved structural assessment during the initial response phase, allowing rapid resumption of the operation of critical transportation infrastructure if the affected bridges are not damaged.

Realizing the proposed framework requires completion of four main research tasks: (i) development of synthetic environment for bridge structures, (ii) recognition and localization of critical structural components, (iii) autonomous vision-based unmanned aerial vehicle (UAV) navigation for image collection, and (iv) structural assessment by processing the collected images.

To investigate and validate the proposed framework, appropriate testing environments that represent the target post-earthquake structural inspection scenarios are crucial. The first task, development of the synthetic environment, produces photo-realistic computer graphics models of

railway bridges that can be used to generate training data for the machine learning algorithm applications, and to distill and validate various algorithms.

The second task, bridge component recognition and localization, is realized by fully convolutional network (FCN)-based semantic segmentation and depth prediction algorithms. The investigation begins with the bridge component recognition using a single real-world image. Based on the lessons learned from the initial work, the recognition and localization results are improved by incorporating recognition results from previous frames, as well as large-scale synthetic training data with accurate ground truth annotations.

The bridge component recognition and localization results are combined with the sparse point cloud data in the third research task, autonomous vision-based UAV navigation, to plan navigation paths for the rapid structural inspection. This task is realized by developing an on-line image data processing strategy that builds three-dimensional (3D) maps of the environment, where target structural components (columns) are represented by rectangular prismatic shapes. Based on the 3D column detection results, waypoints for performing rapid structural inspection are determined.

The fourth task, structural assessment using the collected images, is based on the FCN for semantic segmentation. Compared to the semantic segmentation for structural components, semantic segmentation for structural damage is challenging, because the damage often presents subtle textures, making accurate recognition difficult when the distance to the target surface is large. To address the problem, this research introduces the regions of reliable damage recognition (RRDRs), which are estimated using the depth prediction results. The output of the structural assessment unit is parsed images that contain the information of structural damage, structural components where the damage is identified, and the RRDR.

The final product of this research is a prototype autonomous system for vision-based rapid post-earthquake bridge inspection. The developed system is a significant step toward the automation of the entire bridge inspection procedure in the field environment that minimizes the disruptions to people's lives and business.

ACKNOWLEDGMENTS

I would initially like to express my sincerest gratitude to my advisor, Professor Billie F. Spencer Jr. Until I decided to pursue the Ph.D. in the USA in late 2013, I had never considered living outside of Japan. Therefore, when I started my Ph.D., I was not fully prepared for the new life, including limited language skills. Moreover, I started working on new areas as my Ph.D. research, and I needed preparation period to initiate main body of the research. Getting through all these processes took some time – probably a little more time than that required by an average Ph.D. student. However, he has been always supportive, providing guidance at right timings, based on the clear understanding of what was needed for me to be a successful Ph.D. Thanks to his guidance, I was able to familiarize myself to, and moreover enjoyed, various aspects of life as a researcher, including performing research, teaching students, writing journal papers, and giving lectures and presentations. He will be my model when I work as a faculty in the future.

I would like to acknowledge the valuable input from my committee. I was attracted by the autonomous systems for structural inspection when I was an undergraduate student, but at that time I had no technical background to elaborate the idea and work on the topic in the form of research. The situation has changed dramatically since when I took their courses and had opportunities to discuss with them. Professor Shirley Dyke helped me elaborate my research ideas and expand my research networks through the annual Midwest Smart Structures Colloquium (MSSC). In addition to giving me feedback to my research individually, she motivated discussions with her research group members, which have always been fun and beneficial in the long term. I learned a lot from Professor Mani Golparvar-Fard about how emerging computer vision and artificial intelligence technologies have been applied to address challenges in civil and

construction engineering fields. Every time I discuss with him, I learn new real-world examples of such cases, motivating further learning of relevant topics and development of research ideas. Professor Girish Chowdhary has supported me with the topics for which I had no background initially, that is, robotics. From his courses and discussions with him, I learned how to apply robotics techniques to solve practical problems, including autonomous navigation of UAVs for structural inspection. All those insights from the committee were indispensable for me to organize and proceed with my research.

I would like to acknowledge two individuals from Japan, Professor Yozo Fujino and Professor Tomonori Nagayama, who, as my bachelor's and master's advisors, taught me the significance of the civil and structural engineering fields. They also opened my way to pursuing the Ph.D. in the United States through mentoring, as well as opportunities like Asia-Pacific Summer School on Smart Structures Technology (APSS, currently APSS). Even after I started my Ph.D., they gave me feedback to my research, and helped me further elaborate my research ideas through networking with Japanese professors and institutions working in the related areas.

I greatly appreciate the collaborative efforts and assistance of current and former Smart Structures Technology Laboratory (SSTL) members. Vedhus Hoskere and Tu Hoang were among the best friends and at the same time mentors throughout my Ph.D. During earlier years of my Ph.D., we took many courses about computer vision and machine learning together, helping with each other on assignments and projects. Those relationships continued to grow as we started working on more advanced topics, leading to multiple co-authored journal papers, as well as fun memories in my private life. Yuguang Fu had been the great neighbor in the office for the most part of my Ph.D. His hard-working style sometimes affected me, adding extra amount of motivation to work on my research. Brian Eick has been an important mentor for me regarding my

research as a research assistant for the U.S. Army Corps of Engineers (USACE). With his experience as an engineer working for the USACE, he always helped me identify and reach resources that are needed to make progress with the research. I would like to acknowledge colleagues with whom I have had chances of working in collaboration, Fernando Gomez, Koji Tsuchimoto, Quincy Alexander, for the valuable and enjoyable research and learning experiences.

I would like to express my gratitude to my family. My parents, Hiroshi and Taeko ensured that I had everything I would ever need, and raised me up to now obtaining a Ph.D. My sister, Hiroko, speaks a lot, preventing me from getting bored when I am in my hometown in Japan. I am also grateful for my grandparents for their love from my childhood. One of them, Kikuko Watakabe, is still alive and I am happy that I can show her my Ph.D. degree. I would like to acknowledge my girlfriend, Yui, for her unwavering love and support – we have been apart for 6 years between Japan and Germany, and then USA and Germany, and finally USA and Japan, but she has always been my best partner.

Finally, I would like to acknowledge the financial support from the USACE as a research assistant. The support enabled me to pursue my Ph.D. and enriched my academic life by broadening my area of research and providing additional opportunities for thinking about how academic research benefits life of the public. Based on what I learned from the project, I will always pay attention to the big picture of my research in the future that orients to the people's life and business.

.

TABLE OF CONTENTS

Chapter 1 : Introduction	1
Chapter 2 : Literature Review	12
Chapter 3 : Development of synthetic environment of Reinforced Concrete railway bridges	35
Chapter 4 : Automated bridge component recognition using real-world image data	59
Chapter 5 : Automated bridge component recognition using video data.....	86
Chapter 6 : Bridge component recognition and localization using large-scale synthetic dataset.	95
Chapter 7 : Vision-based UAV navigation strategies for automated image collection	106
Chapter 8 : Vision-based post-earthquake structural assessment of RC railway viaducts	141
Chapter 9 : Conclusions and Future Directions	155
References	165

CHAPTER 1: INTRODUCTION

Railway infrastructure plays a crucial role in every aspect of human's lives. In some countries like Japan, railways are the dominant transportation mode (Table 1), accounting for more than 75% of total passenger transport [1]. Railway infrastructure in urban areas supports millions of people commuting to and from work and school [2]. Large capacity and stable operations have allowed railway systems to become a foundation for social and economic activities.

Table 1 Passenger transportation modal share in Japan [%]

Rail	Road	Water	Air
78.9	20.5	0.3	0.3

Railway operations is dependent on their infrastructure, such as bridges and tunnels. The fraction of various types of railway structures in Japan are shown in Figure 1 (a) and (b) for the Shinkansen (Japanese bullet train) lines and the main railway lines in Tokyo metropolitan area, respectively. For Shinkansen lines, bridges typically account for roughly 50% of the lengths of the railway routes, and most of the bridges are classified into a specific category, that is, viaducts. Tunnels are another common type of structure for the Shinkansen lines, accounting for approximately 50% of the total route lengths for the Kyusyu, Sanyo, and Hokuriku Shinkansen. For normal railways, bridges and viaducts also comprise a significant portion of the railway routes. The number of viaducts on the railway lines shown in Figure 1 (b) adds up to 5,240. Tunnels are also one of the main structure types, although the fractions tend to be less than the fractions for the Shinkansen lines. To maintain the operation of railway systems, the safety of all the railway structures need to be guaranteed based on the up-to-date information obtained by structural inspections.

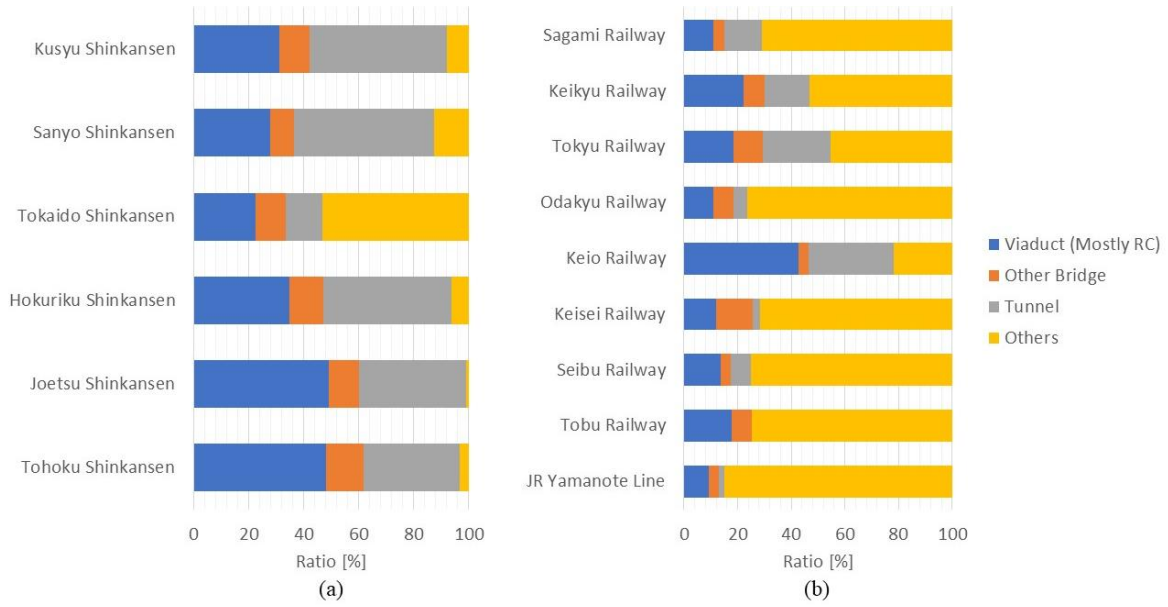


Figure 1 Ratio of railway structures
(a) Shinkansen (Japanese bullet train) lines (b) Main railway lines in the Tokyo metropolitan area [3][4]

Despite the strong demand for the uninterrupted railway operations, extreme situations sometimes undermine the safety of railway structures, limiting the performance of the railway systems. In Japan, earthquakes are one of the major sources of the risk to the railway structures. Japan has experienced severe earthquakes, such as the Kobe Earthquake in 1995 (M7.2), the Tohoku Earthquake in 2011 (M9.0), and the Kumamoto Earthquake in 2015 (M6.5 and M7.3) [5], which caused damage to railway structures in the affected areas. Even for smaller earthquakes (e.g., Shizuoka Earthquake in 2011, M6.5), railway operation can be halted for several hours [6]. To illustrate the need for rapid post-earthquake assessment of railway infrastructure, the aftermath of different levels of earthquakes in Japan are reviewed with respect to the effect on the railway operations in the next paragraphs.

Case 1: Kobe Earthquake (1/17/1995, M7.2) [5]. This earthquake is known to be an extremely severe earthquake with the epicenter depth of 16 km, located near the densely populated area in the city of Kobe. The earthquake caused the death of 6,425 people, injury to 43,772 people, and damage to 488,222 houses, including the collapse of 110,457 houses [7]. After the earthquake,

rapid initial response was crucially important for life-saving activities. The results of the search and rescue activities are shown in Figure 2 [8]. The number fraction of individuals found alive dropped rapidly as time progressed, and the search and rescue activities were rarely successful after three days (72 hours).

Despite the need for rapid reconnaissance after such a disaster, human resources available during the first 72 hours after the earthquakes were limited, because of the severe damage to homes and the city's transportation systems. For example [9], only 41% of the Kobe city's workforce was available on the day of the earthquake. Similarly, only 60% and 70% of the city's workforce was available during the second and the third day after the earthquake, respectively. Although identifying the damage to the transportation structures is necessary for effective response and recovery activities, collecting the information of structures from the entire affected area is not a straightforward task with the limited human resources.

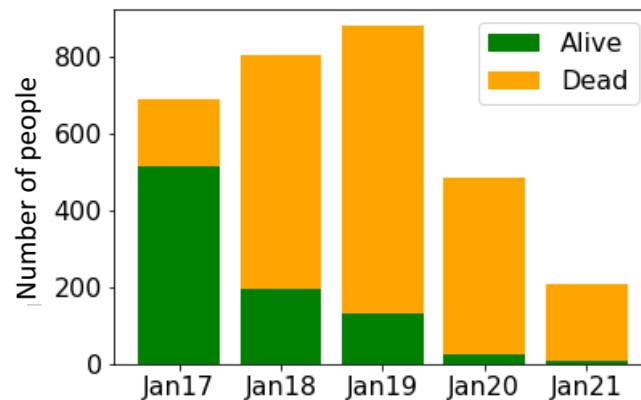


Figure 2 Number of people found during rescue activities [8]

Case 2: Shizuoka Earthquake (8/11/2009, M6.5) [5], [6]. The epicenter of the earthquake was located at the depth of 23 km of the bay in Shizuoka, Japan. The earthquake caused the death of one person, injury of 319 people, and partial collapse of five houses. Excluding minor non-

structural damage, the earthquake did not cause damage to any railway infrastructure. However, railway operations in the area were halted between 3-5 hours to carry out the safety inspections. In particular, 31 Shinkansen trains were cancelled, and 99 Shinkansen trains were delayed more than four hours on the day of the earthquake [10]. These cancellations and the delays hindered people's social and economic activities, and brought economic loss to the JR-Central company through the need to refund the cost of special express tickets [11].

A problem which needs to be highlighted regarding this earthquake is that railway operations were interrupted, even though no major damage to the railway structure was observed. Because the railway structures exist everywhere along the railway routes, visual inspection by human experts becomes an extremely time-consuming task. To reduce the social and economic damage to the railway users and the railway companies, rapid initial safety inspection of railway infrastructure is crucially important, even for earthquakes that cause little damage.

Case 3: Tohoku Earthquake (also called The Great East Japan Earthquake). Situation in Tokyo metropolitan area (3/11/2011, M9.0) [5]. The Tohoku earthquake was one of the most damaging earthquakes in the recent Japanese history. The earthquake and the following Tsunami waves caused the death of 19,533 people, 2,585 missing people, injury of 6,230 people, collapse of 121,768 houses, partial collapse of 280,160 houses, other damage to 744,396 houses [9]. In addition to the area near the epicenter, the impact of the earthquake reached the central, east, and north part of Japan, including the Tokyo metropolitan area (about 400km away from the epicenter).

In the Tokyo metropolitan area, the performance of the railway and other transportation systems decreased significantly because of suspected damage to various structures. For example, immediately after the earthquake, trains were stopped, and the passengers were evacuated to the nearest station guided by the railway staff. Concurrently, inspection and maintenance of railway

structures were carried out. Once the safety of the railway routes was ensured, the details of the railway operations were planned and executed. As a result, the railway operations were halted for 16-18 hours after the earthquake. The time spent in these initial response activities of three railway lines near Tokyo station are summarized in Figure 3 [12].

The progress of railway operation recovery after the earthquake is shown in Figure 4 for the areas within 30 km and 50 km from the Tokyo station. During the first 5-6 hours after the earthquake, the ratio of operation recovery was nearly 0%, showing operational downtime occurs regardless of the extent of the damage to the railway structures. The main problems reported during the downtime were traffic congestion and communication failure [12]. These problems pose difficulties in organizing inspectors, accessing to the sites, and communicating the evaluations by the inspectors. As depicted in Figure 5, more than 515 million people were stranded in the Tokyo metropolitan area and unable to return home and on the day of the earthquake because of the halted or limited operations of the railway and other transportation systems [9].

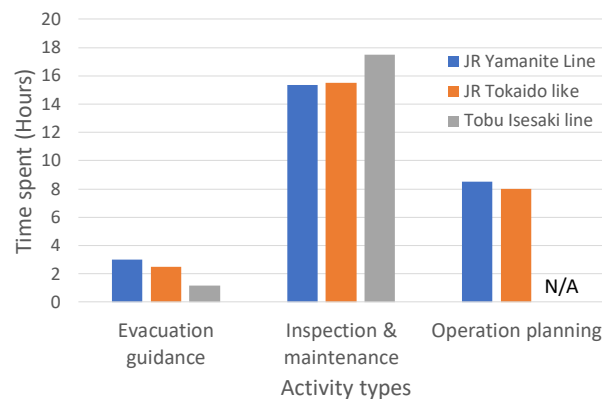


Figure 3 Initial response activities by the railways near Tokyo station (after the Tohoku Earthquake, 2011) [12]

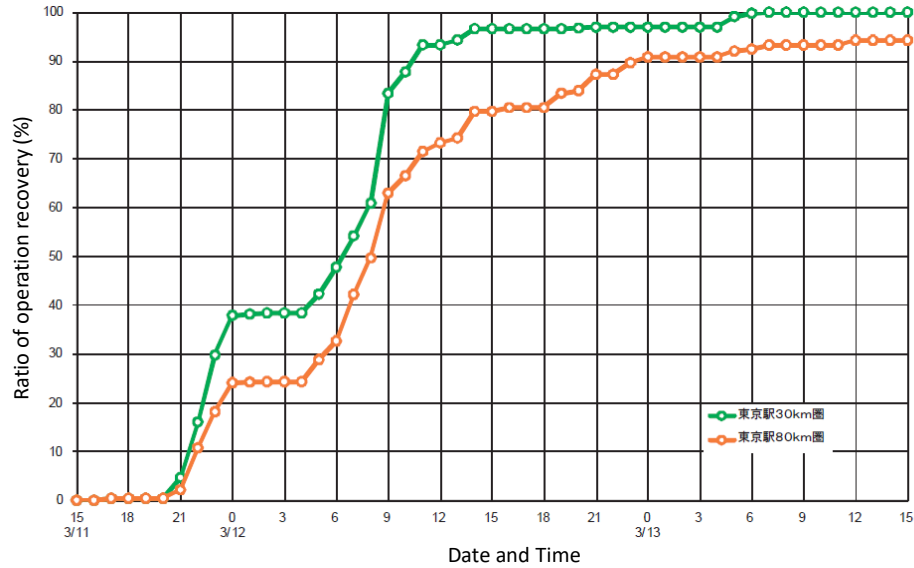


Figure 4 Ratio of operation recovery of railways near Tokyo station [12]
(Green: within 30 km from Tokyo station, Orange: within 80km from Tokyo station)



Figure 5 People stranded in Tokyo metropolitan area (photographed by UCK inc. [13])

These three cases show the importance and difficulty of rapid post-earthquake safety inspection of railway structures after all levels of earthquakes. The post-earthquake safety inspection of railway structures needs to (i) evaluate all bridges located in the affected area (ii) in a short time (iii) using limited human resources. To achieve this goal, the post-earthquake inspection is typically carried out in two phases. As shown in Figure 6 for bridge inspections, all bridge structures in the affected area are first evaluated based on the coarse information which is immediately available after earthquakes (“preliminary” inspection). If the existence of damage is

suspected, detailed inspection is carried out to get reliable information about the safety of the structures (“detailed” inspection). After the Tohoku earthquake, the maximum ground acceleration was used to make preliminary decisions about the railway structures [12]. If the maximum ground acceleration was between 40 gal ($\approx 0.04g$) and 100 gal ($\approx 0.1g$), speed limits were imposed, and human inspectors visually observed the railway structures from running trains. When the maximum ground acceleration exceeded 100 gal, railway operations were halted, and visual inspections were carried out from the ground. The preliminary inspection based on the ground acceleration is widely accepted by railways in Japan, because of the immediate availability of the data after earthquakes[14].

The problem of the preliminary inspection using the ground acceleration data is an uncertainty in the safety decisions. To measure the ground acceleration during earthquakes, accelerometers need to be pre-installed at every location of interest. Even when the ground acceleration at the exact location of a structure is available, the damage evaluations include significant uncertainty, because the approach neglects properties of the structures and environmental conditions. Therefore, preliminary inspection based only on the maximum ground acceleration need to be conservative, leading to many undamaged bridges being included in the list requiring detailed inspections [15].

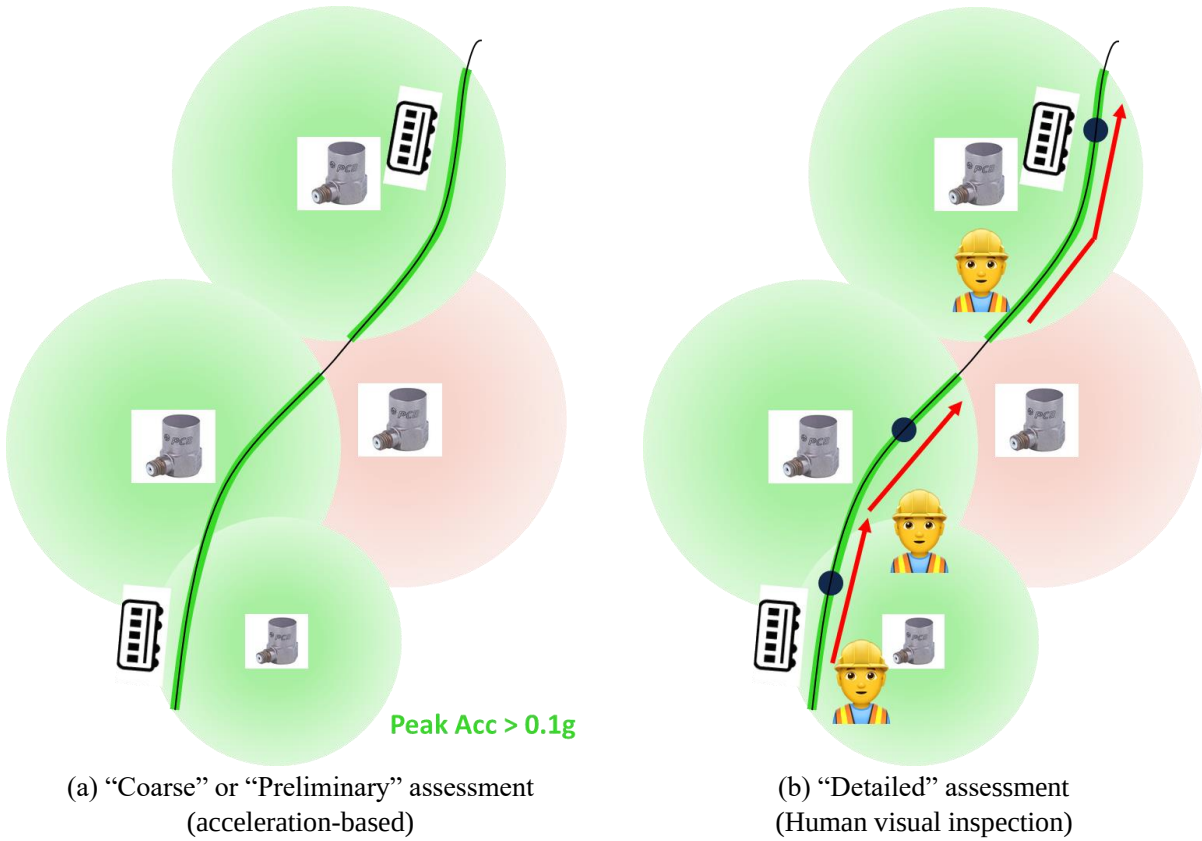


Figure 6 Typical flow of post-earthquake bridge inspection

The limitations of the current inspection method reveal the importance of developing an inspection approach that is based on the data collection from densely located points along the railway routes and has strong correlation with the damage evaluated by visual inspections. For the data collection from densely located points, a platform moving along the railway routes is effective. The representative platform of this class is an unmanned aerial vehicle (UAV, often called unmanned aerial system, UAS). UAVs are advantageous in accessing broad areas, even with significant traffic congestion. Besides, UAVs are typically equipped with imaging sensors (cameras). Image data can capture the visual information of structures, which is the mode of information evaluated during the post-earthquake visual inspection by human inspectors. Therefore, the visual data collected by UAVs is expected to have a strong correlation with the safety decisions derived by the visual inspections.

This research proposes a framework for autonomous vision-based inspection of RC railway bridges to realize rapid post-earthquake response and recovery, which are crucial for minimizing the disruptions to people's lives and business (Figure 7). Realizing the proposed framework requires completion of four main research tasks: (i) development of synthetic environment for bridge structures, (ii) recognition and localization of critical structural components, (iii) autonomous vision-based unmanned aerial vehicle (UAV) navigation for image collection, and (iv) structural assessment by processing the collected images.

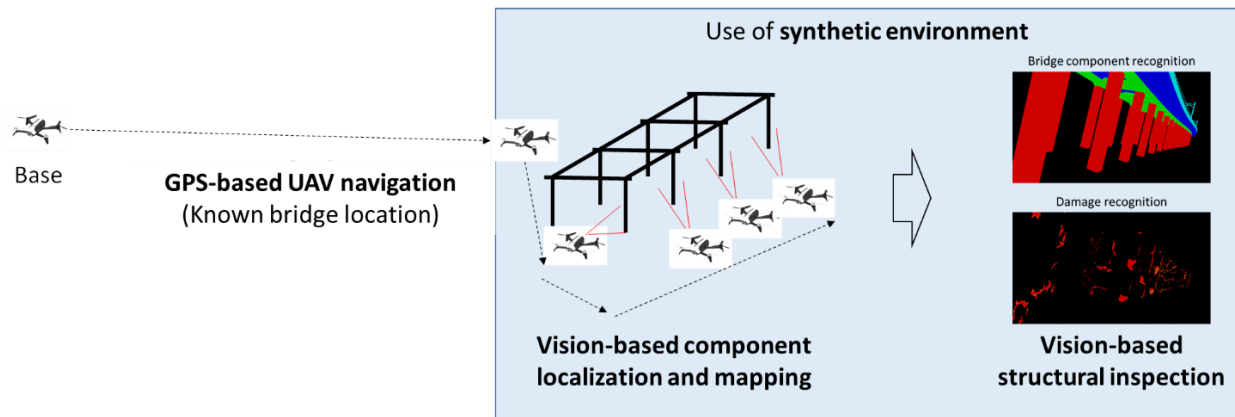


Figure 7 A framework of vision-based post-earthquake preliminary safety inspection investigated in this research

This dissertation is structured as follows. Chapter 2 reviews literature in the related areas. First, the current practice of vision-based inspections of civil infrastructure (bridges in particular) is discussed. Then, the key techniques investigated in this research, such as computer vision, machine learning, and techniques for autonomous navigation, are reviewed. Then, the applications of those techniques to civil infrastructure inspection tasks are discussed, including the use of UAVs for image data collection and image post-processing approaches to detect structural damage.

Chapter 3 discusses synthetic environments that can facilitate the investigation of vision-based component localization and mapping, as well as vision-based structural inspection. Two

synthetic environments are developed for this research: Unity game environment [16] to simulate video data during UAV flights, and a set of synthetic environments developed using Blender software [17] to enable in-depth investigation of component localization and mapping and structural inspection for a special type of RC rigid frame railway bridges, that is, viaducts.

To begin with the discussions of vision-based component localization and mapping, automated vision-based bridge component recognition using real-world image data is discussed in Chapter 4. Two different semantic segmentation algorithms (Fully Convolutional Networks, or FCNs [18] and SegNet [19]), as well as three different configurations of semantic segmentation networks (naïve, sequential, and parallel configurations) are investigated to realize a network that performs robustly even for complex urban scenes.

Chapter 5 extends the work presented in Chapter 4 to video data (image sequences) obtained during UAV flights. To get consistent results throughout the UAV flight, recurrent neural networks are applied, and the information from the previous frames is utilized as hidden states during subsequent frames. Large amount of video data generated effectively in the synthetic environment is used to train and test the proposed network with the recurrent neural networks.

Chapter 6 makes significant improvement to the image-based structural component recognition investigated in Chapter 4, and further estimate depth maps that can be used for bridge component localization by incorporating synthetic data generated using the Blender software. The improved dataset contains nearly 8,000 (8K) images, enabling the training of large semantic segmentation networks with reduced overfitting effects. The trained network performs accurately for RC viaducts of Japanese high-speed railway viaducts.

Chapter 7 discusses the use of the synthetic environment and the results of bridge component recognition and localization to the autonomous UAV navigation for image collection

of RC viaduct columns. The application of Simultaneous Localization and Mapping (SLAM) algorithm is investigated to integrate the information obtained in the previous sections and identify appropriate viewpoints to collect images of the target columns.

Chapter 8 discusses the automated vision-based structural assessment of RC railway viaducts after earthquakes, assuming images of target structural components have been collected successfully. To achieve the goal, deep semantic segmentation networks are trained and tested using both synthetic and real-world images. The resulting network can detect, localize, and quantify concrete damage (cracks and spalling), as well as exposed rebar.

Chapter 9 concludes this research and discusses future extensions.

CHAPTER 2: LITERATURE REVIEW

This chapter provides an overview of the literature related to the topics investigated in this research. Literature directly related to the investigations discussed subsequently will also be reviewed in the corresponding chapters.

2.1 Vision-based inspection of civil engineering structures

Inspections of civil engineering structures, such as buildings and bridges, rely on human vision as a primary source of information. In particular, most parts of the post-earthquake preliminary safety evaluation guidelines for buildings and bridges are spent in describing the correspondences between visual damage patterns of structures and severity classes [20]–[23]. Similarly, detailed post-earthquake inspections after the preliminary inspections [22]–[24] and regular inspections in normal times[25], [26] rely on visual information to decide appropriate actions to the structures. However, damage properties, such as damage dimensions and the ratio of damaged members among all members of the same type, are also evaluated during the detailed post-earthquake inspections and regular inspections. Furthermore, during regular inspections, physical methods typically using hammers are applied to evaluate the locations with potential damage identified by the visual inspections. In summary, the visual information is primarily used by all types of structural inspections, although different sources of information are combined depending on the level of details desired by each inspection. In this sense, post-earthquake preliminary safety inspection is an extreme case, where the visual information is almost the only source of information.

One of the technical difficulties of visual inspections is the limited accessibility to the structural members. For civil engineering structures, some of the structural members are located at the height beyond human's reach. Also, the area underneath bridges could be occupied by water, other traffic etc., which further limit the access to the structural members. Even when all critical members are visible by the human inspectors, walking around the structures to see all the critical structural members takes time because of the large size and number of the civil engineering structures.

Recently, unmanned aerial vehicles (UAVs) has gained popularity as a tool for the bridge inspection to overcome the difficulty of the limited access to the structural members. Use of UAVs for getting close-up images of bridge components with limited accessibilities are demonstrated by Minnesota Department of Transportation (MnDOT) [27], [28], Florida Department of Transportation (FDOT) [29], Michigan Department of Transportation (MDOT) [30], Gillins et al. [31] etc. The projects mentioned above demonstrates the use of UAVs for the regular inspections, and shows the example close-up images of bridge members at the locations with limited access.

Although the experiences from the demonstration projects for regular inspection are applicable to the post-earthquake preliminary safety inspections, use of UAVs need to be adjusted for the different requirement on the time and data quality. In contrast to the regular inspections, post-earthquake safety inspections typically have stronger requirement on time and less requirement on data quality (*presence* of critical damage is the primary interest). Also, inspections of bridges with simple and standardized structures tend to take more time than the inspections of large and complex structures, which are the frequent target for the UAV use in regular bridge inspections. Therefore, the investigation of automated system for the post-earthquake safety inspection of mass-produced railway viaducts with standardized design is important.

The automated post-earthquake preliminary safety inspection has two components – automated vision data collection using UAVs and automated vision data post-processing for the evaluation of structures. In section 2.2, key computer vision and machine learning techniques for the automated vision-based inspections are introduced. In section 2.3, further discussions are made regarding the recent advance of the autonomy of robotic platforms, including UAVs. In sections 2.4 and 2.5, the recent applications of computer vision, machine learning, and autonomous navigation methods to the civil engineering field are reviewed.

2.2 Computer vision and machine learning

The recent advances in the technologies for smart or automated visual inspections for civil engineering structures are supported by the advances in the computer vision and machine learning field. The computer vision methods can be used to extract features representing the important patterns for specific applications, and the extracted features are classified, regressed, or evaluated by machine learning algorithms. The feature extraction and evaluation are merged in modern methods, which allows for learning a complex nonlinear mapping from training data.

In classical methods, local image contexts are extracted using carefully tuned mapping operations, such as Canny edge detector[32], Harris corner detector [33], line detection by Hough transform [34], Line Segment Detector (LSD) [35], Histogram of Gradient (HOG) [36], Scale-Invariant Feature Transform (SIFT) [37], Affine-invariant version of the SIFT (A-SIFT) [38] etc. Feature description is often combined with classifiers, such as support vector machines [39] and neural networks[40], [41] to extract the high-level information (semantic labels) about the object in the image.

The application of the classical methods becomes increasingly difficult as the problem needs to encode the higher-level contexts, because of the complexity and the versatility of the

target object. To extract the high-level information from images without increasing the parameters prohibitively, convolutional neural networks (CNNs) were proposed and have been used extensively [42].

In CNNs, convolutional filters are applied, followed by nonlinear activations and spatial pooling operations, depending on the architecture of the network (Figure 8). As the input image goes through multiple layers of filtering and nonlinear activations, the input image is mapped to a high-level representation incorporating the contexts in a gradually enlarging field of view (FOV). The output at the final layer (**F** in Figure 8) is regarded as a feature vector that can be fed into the fully connected layers (FCLs) and a softmax layer for classification. In contrast to the classical feature extraction and classification methods, the convolutional filters are learned from training data, instead of using hand-made filters. This method has been successfully applied to wide range of image classification problems, ranging from handwritten digit classification problem (MNIST [43]), 10-class image classification problem (Cifar-10 [44]), and large-scale image dataset for image classification, object detection, and object localization (ImageNet [45]). In particular, the image classification challenge on the ImageNet dataset shows that the state-of-the-art CNN architectures can achieve less than 5% top-5 error for the 1000-class classification problem [46][47].

The CNN can be re-interpreted as a dense feature extractor [18]. As shown in Figure 8, the output of the last layer **F** is regarded as a ‘feature map’, which stores feature vectors at the corresponding locations of the image. Then, the feature vector at each location is classified into an appropriate category using FCLs and a softmax layer. During implementation, the FCLs applied at each location can be expressed by convolutional layers with filter size 1×1 . After converting the

FCL output to the softmax probabilities, the predicted probability scores for each category is upsampled using, for example, nearest neighbor interpolation or bilinear interpolation.

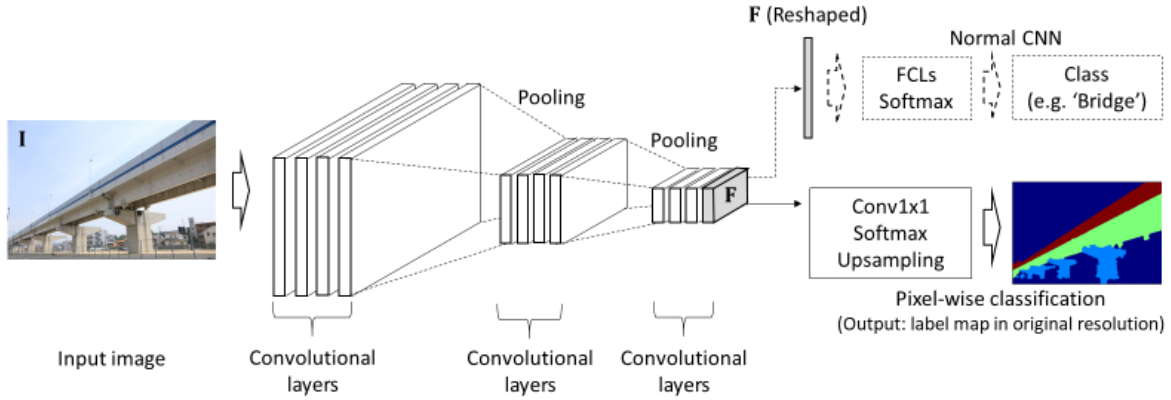


Figure 8 Convolutional neural networks (CNNs) and their extension to feature extractor

A problem of the simple extension of the normal CNN is the low resolution of the representation $\mathbf{F}$, which is caused by dropping spatial information at the pooling layers. Because the label map in original resolution is recovered by upsampling the lower resolution map, the resulting map may not represent the local context in the input image correctly.

Another difficulty of training the simple pixel-wise classifier for the recognition of objects in complex scenes is the versatility of scales in which the objects are photographed. For example, bridge components can appear in various sizes in images, depending on their distance from the camera and their actual sizes in 3-dimensional space. In the simplest approach shown in Figure 8, every object is represented at a fixed low resolution, based on the nonlinear convolutional filter responses from a fixed field of view (FOV).

To recognize objects with various sizes at an enhanced resolution, features are often extracted at multiple scales. Two main approaches for the multi-scale feature extraction are the approach using image pyramid [48] and the approach using multiple layers of a single CNN [18]. Following the terminologies in the referenced papers, the first approach is called a multi-scale

CNN, and the second approach is called a fully convolutional network (FCN) in this paper. Note that the exact definition of an FCN is a network which “takes input of arbitrary size and produces correspondingly-sized output with efficient inference and learning” [18], thus including both the “multi-scale CNN” and the “FCN” defined in this paper.

Multi-scale CNN [48] (Figure 9) is a compact and effective CNN architecture for the multi-scale feature extraction of complex scene images. The network extracts feature at multiple scales by applying convolutional and pooling layers shared among images that are downsampled to multiple resolutions (i.e., an image pyramid). After extracting the feature maps at multiple resolutions, the feature maps are upsampled and concatenated to form a dense multi-scale feature map, $\mathbf{F}$. Then, the feature vector at each location is classified into an appropriate category as described in the previous section.

FCN extracts the multi-scale feature representation without explicitly applying the same filter multiple times (Figure 10). Instead of downsampling the input image, the FCN uses ‘skips’, which combine the label prediction from multiple layers of a single CNN. In the example architecture shown in Figure 10, the feature vector at every location in the shaded layers, $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$, is classified into an appropriate category by convolutional layers with filter size 1×1 . Then, the predictions from the layer $\mathbf{f}_2$ is upsampled to the resolution of $\mathbf{f}_1$, and $\mathbf{f}_2$ after upsampling and $\mathbf{f}_1$ are added together. Similarly, the merged prediction from the layers $\mathbf{f}_1$ and $\mathbf{f}_2$ are upsampled and added to the prediction from $\mathbf{f}_0$. Finally, the merged predictions from the layers, $\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2$, are upsampled to the original resolution to get the final label estimation at each pixel location. The upsampling operations can be implemented using filters, either fixed (e.g., bilinear filter) or learned.

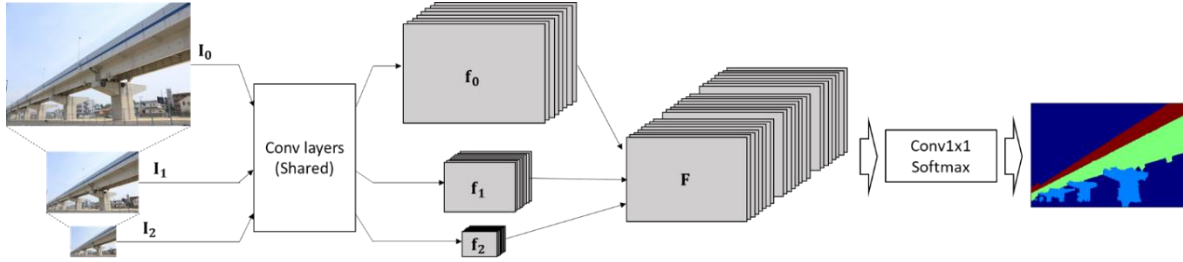


Figure 9 Multi-scale CNN architecture

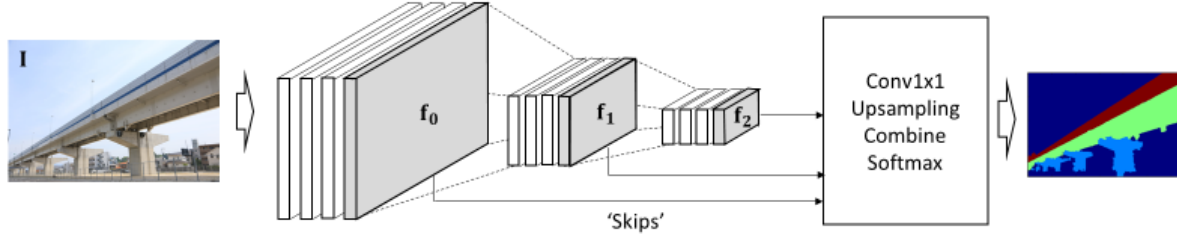


Figure 10 Fully convolutional neural networks (FCNs)

The parameters of the CNN as a dense feature extractor can be learned end-to-end using gradient-based methods (e.g., Adam optimizer [49]). However, the training of the parameters becomes difficult as the size of the network increases. To improve the classification results for the deep network architectures, various techniques have been proposed. When the size of the network increases, the training of the parameters becomes difficult. Various techniques have been proposed to improve the classification results for the deep network architectures. Weight decay [50] adds an L2 regularization term to the cross-entropy loss function to reduce the overfitting effect. Batch normalization [51] scales the input to the nonlinear activation function at each layer using the learned mean and standard deviation parameters to accelerate and improve the convergence of the parameter updating process. Median-frequency balancing [52] is a technique to weight the sum of cross-entropy loss for each class to compensate for the imbalance of the data distribution. Dropout [53] prunes outputs of designated layers during training time to improve the robustness of the trained network (reducing overfitting effect). For training deep networks, residual networks (ResNets) [46] are known to be effective for improving the training results. The residual networks

are characterized by interlayer connections, where the layers effectively learn residuals between the desired output and the input to the layers (Figure 11).

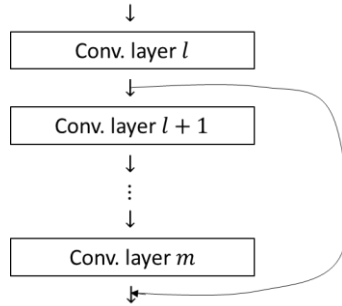


Figure 11 Example of residual network connections (From layer l to layer m)

The dense feature extractor can be used for different purposes. Main applications are listed below.

Semantic segmentation: Semantic segmentation problem assigns a semantic label to each pixels of an image. This category includes a multiscale neural network applied to the segmentation of natural scenes [48], modification of the VGG16 architecture [54] to the semantic segmentation for PASCAL VOC 2011 dataset [55][18], a SegNet [19] (a variant of CNN-based dense feature extractor with encoder-decoder architecture) applied to the segmentation of road scenes (CamVid dataset [56][57]), a U-net and its successor [58]–[60] (an encoder-decoder architecture with inter-layer connections to incorporate multi-scale features), application of FCNs to a large scale real and virtual datasets of urban scenes [61], [62], multiscale CNNs applied to RGB-D data of indoor scenes [63]. Sometimes, semantic segmentation from an image is combined with depth regression and instance-level segmentation performed in parallel to refine the labels [64].

Object detection: Object detection aims at finding target objects in an image typically by drawing a bounding box including each object. Generally, these methods transforms an input image into feature representations by CNN-based architecture, apply sliding windows to extract

the candidate bounding boxes in the feature space, and apply another small CNN to evaluate the object existence and refine the bounding box. Examples are spatial pyramid pooling [65], regions with CNN features (R-CNN) [66], Fast R-CNN [67], Faster R-CNN [68]. Computationally more efficient architectures have also been proposed, such as You only look once (YOLO) [69], [70].

Instance-level segmentation: Instance-level segmentation is similar in concept to semantic segmentation in the sense that an appropriate label is assigned to every pixel of an image. However, instance-level segmentation aims at separating different instances of objects by assigning different labels. Mask R-CNN [71] starts from the bounding boxes generated by the faster R-CNN, and further generate binary mask using another small FCN. Another approach to the instance-level segmentation is to use a discriminative loss function, which represents a distance metric between clusters in feature space [72]. Use of depth estimation is also investigated to achieve the instance-level segmentation [73].

Depth regression: Recently, depth regression has been investigated as a possible replacement or augmentation to the 3D information obtained from geometric computer vision methods. A new loss function for the depth regression using CNN is proposed by Eigen and Fergus [52]. Laina et al. [74] successfully applies reverse Huber loss function with the fully convolutional network architecture. The depth estimation by the CNN is used to refine semantic segmentation [64]. Data fusion between 3D points from SLAM and estimated depth by CNN has also been proposed to achieve a dense and robust 3D reconstruction of the natural scenes [75].

Spatio-temporal analysis: Recurrent unites have been combined with CNN feature extractor to model a sequence of images (i.e., video data). Simple recurrent neural networks (RNNs) can be implemented by regarding the past input data as additional channels of the current input data and optimizing the parameters by backpropagation through time [76]. MaskTrack

Convnet [77] is a CNN architecture similar in concept to the simple RNN, where an input image is augmented by the estimated object mask from the previous step to track and segment objects.

Despite the conceptual simplicity of such RNN architectures, learning from a sequence of data becomes difficult as the length of the data increases. Gradient-based parameter updating for such sequences is known to be inefficient, because backpropagation error vanishes or explodes rapidly (i.e., vanishing gradient problem). Long short-term memory (LSTM) units [78] have been proposed to circumvent the problem by explicitly implementing memory in the architecture (i.e., constant error carousel). The data and error flow from/into memory is controlled by “gates”, which are modeled by sigmoid functions. Gated recurrent units (GRU) have also been proposed as a simplified alternative to the LSTM unit [79].

The LSTM and GRU cells have been integrated into CNN architectures to model sequences of 2D maps, including images and grid measurement data. Convolutional Gated Recurrent Networks [80] have been proposed to improve the semantic segmentation and object tracking tasks for the video stream data of urban scenes. The Gated Recurrent Flow Propagation algorithm [81] combines the GRU with optical flow to achieve accurate segmentation of the urban scenes. Convolutional LSTM (ConvLSTM) network [82] has been proposed to perform weather forecasting from recent radar echo sequences represented by 2D maps.

2.3 Autonomous navigation techniques

Autonomous navigation has been investigated actively in robotics field, particularly for the development of autonomous cars. An example software architecture of the autonomous car, “Stanley”, which won the 2005 DARPA Grand Challenge, is shown in Figure 12 [83]. The Stanley was successful in autonomously driving a 175-mile route in Mojave Desert in 6 hours and 53

minutes using the waypoints defined by the GPS coordinate. The Stanley has five software components – Sensor interface, perception, planning and control, and user interface. After getting data from laser scanner, camera, radar, GPS, wheel encoders etc., the sensor data is processed in the perception unit to estimate the robot pose and model the external environment (finding road, finding obstacles). Then, appropriate local paths are planned by the path planner and executed by the control unit. Finally, interfacing units are placed to allow for setting and configuring the system. Based on the early success like the Stanley, each of the five components have made significant progress, leading to multiple autonomous vehicle platforms that have been tested in actual roads. For example, Waymo [84] uses cameras, lasers, and radar to collect and analyze information around the car in every direction, and plans actions that lead to safe navigation. Tesla has developed multiple commercial autonomous driving systems [85] that are equipped with cameras, ultrasonic sensors, and radar to perform basic cruising tasks automatically, and claims that the full automation will be achieved in the near future [86]. The remaining part of this section focuses on perception and path planning for autonomous navigation, which is relevant to this research.

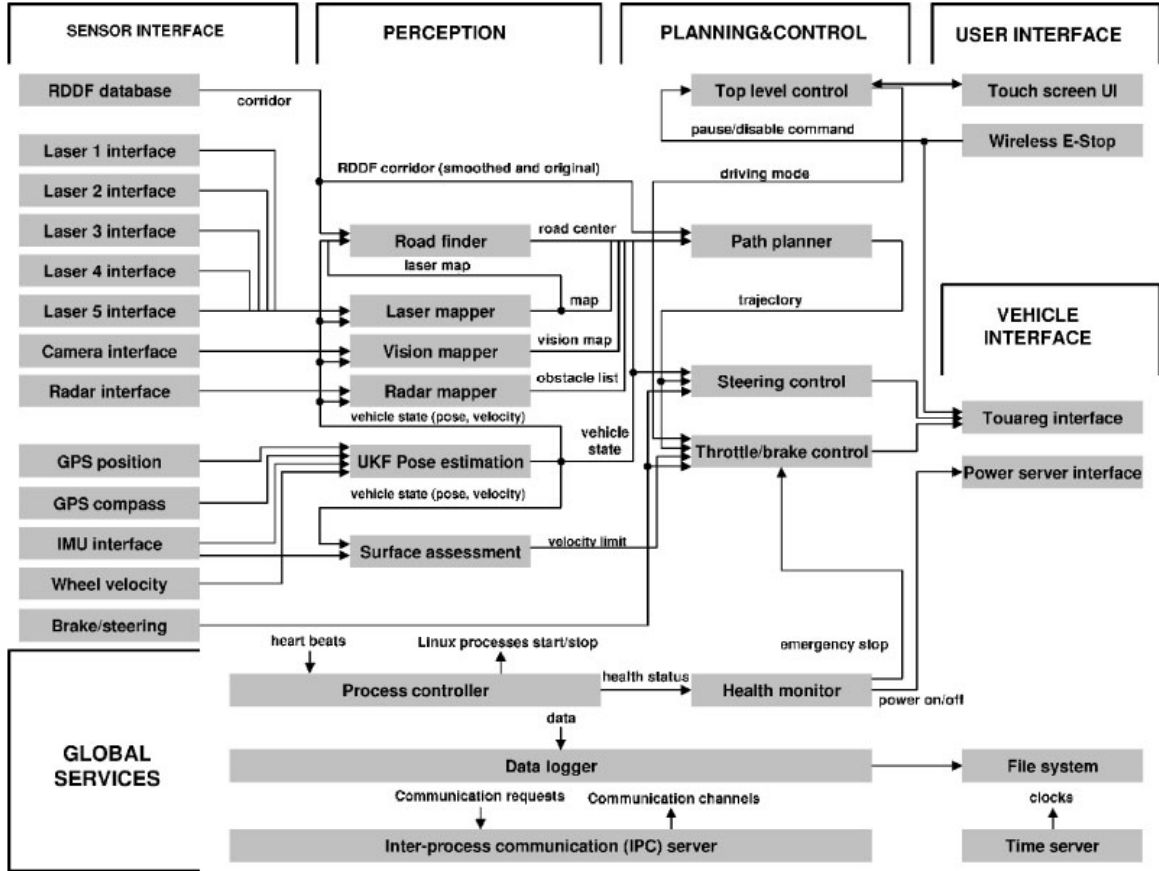


Figure 12 Software system of Stanley (From [83])

Perception unit essentially has two roles – estimation of the states of the robot and mapping of the external environment. Robot states typically includes position, orientation, and linear and angular velocities. To improve the state estimation accuracy by leveraging measurements by different sensors, Kalman filter (KF), Extended Kalman filter (EKF), or Unscented Kalman filter (UKF) are frequently used [87]–[89] (Stanley uses the UKF). In Stanley, the mapping component begins with modeling the external environment by creating a map of 3D points incrementally using data from the laser scanners. Then, the Stanley converts the 3D points into occupancy map, where every location on the ground is classified into one of the three classes: occupied, free, and unknown. Finally, drivable road surface is identified by removing occupied surface and extend the unoccupied surface to longer range with the help of a vision sensor.

The process of robot state estimation and mapping of the environment is integrated in the framework of simultaneous localization and mapping (SLAM, or Visual SLAM to be more explicit about the use of vision data). Mono SLAM [90] uses the EKF to estimate the locations of feature points and the pose of the robot. By limiting the search region for feature matching using the predicted position and covariance, the algorithm achieved processing at 30 Hz. Large Scale Direct SLAM (LSD-SLAM) [91] is a direct feature-less monocular SLAM, which incrementally expands and updates a pose-graph of the map, where the keyframes (images, inverse depth maps, variance maps of the inverse depth) form nodes and the 3D similarity transforms are assigned to the edges. ORB-SLAM [92] extends the feature-based SLAM approach by introducing a multi-thread implementation with an efficient loop-closure method. Data fusion between the SLAM and monocular depth estimation by CNN has been proposed to improve the dense 3D mapping of the external environment (CNN-SLAM) [75]. When additional depth information is available from either stereo cameras or depth cameras, the map point density and accuracy of both feature-based SLAM and direct SLAM methods can be improved [93]–[96].

After getting the perception of the robot states and external environment, robot paths are planned and sent to the control unit. Because the sequence of waypoints along the route was provided in the 2005 DARPA Grand Challenge, the path planning in the Stanley was defined as a local obstacle avoidance problem, which adjusts the lateral offset of the car from the centerline of the road. However, different types of path planning problems exist corresponding to the scale and the goal of the planning.

The path planning problem solved by the Stanley is referred to as a local path planning problem, where the robot need to find a feasible path to the goal without collision in a potentially changing environment [97], [98]. In contrast, in a global path planning problem, knowledge about

the environment is assumed to be known, and the complete path from the starting location and the goal location is determined before the robot starts the navigation. When the robot is navigating in an unknown environment for the first time, the problem is termed as an exploration, where the primary goal is to build a map (occupancy grid) of the environment as complete as possible, while avoiding obstacles during navigation.

For the local and global path planning in an environment with complete characterization of obstacles, approaches such as cell decomposition method, potential field method, and methods based on random sampling have been proposed. Cell decomposition method [99] divides the configuration space into cells and label each cell by one of the three labels: empty, full, and mixed. The cell representation of the environment is then used to create a piecewise linear path through the empty cells. Potential field method [100] models the environment using a potential field which has high potential at obstacle locations. The method then create a path by applying a “force” associated with the potential field. Sampling-based methods have been proposed as computationally efficient alternatives to the above two types of methods even when the robot navigates in high-dimensional state space [101]. Probabilistic roadmaps (PRMs) [102] randomly samples points in the configuration space, and connect the points to build a map (a 2D graph) of the environment. The path is then found by querying a feasible route with low cost on the graph. The PRM is improved computationally by the rapidly exploring random tree algorithm (RRT) [103] and RRT* [101]. In RRT, instead of connecting sampled points as in the PRM, the tree is expanded from the nearest existing node toward the sampled point by an amount achievable by the robot. By this method, the path can be planned incrementally during the navigation with decent amount of computation. RRT* is an improved version of the RRT method, where the optimality is ensured in an asymptotic sense.

Motion planning problems is a subset of path planning problem, where the robot dynamics need to be considered as differential constraints [104]. Sampling-based methods combined with the optimal control framework has been proposed by replacing the cost function defined by the Euclidean distance by the cost function used in the optimal control of the robot [105]–[107]. Also, to reduce the complexity of the problem, the search space is often discretized by defining finite set of actions and express the path by a set of those actions (the actions are also called motion primitives or trim trajectories, depending on the contexts) [108]–[110].

The common part of the local and global path planning problems, including the motion planning problem, is that the goal of all problems is the navigation from starting location (state) to the goal location (state). To achieve higher-level tasks, such as “park the car” or “inspect this bridge column”, sequence of waypoints which will be followed by any path planning algorithm needs to be defined.

One of the high-level tasks defined in an unknown environment is exploration. The primary goal of the exploration task is to build a map (occupancy grid) of the unknown environment by navigating the robot to the boundary between the known space and unknown space called frontier [111]. The frontier-based exploration algorithm is implemented on a UAV and autonomous indoor navigation is demonstrated in [112], [113]. Autonomous navigation following building wall is also demonstrated using Bug algorithm [113].

The significant improvement in the autonomous navigation techniques lead to the success of the demonstrations of navigation tasks, such as obstacle avoidance (local path planning), path planning in the environment when the map is available, and navigation in unknown environments to build a map. The focus of this research is the autonomous inspection of railway bridges using UAVs, where the robot (UAV) is expected to recognize the target structural components (e.g.

columns), rapidly take images of the close-up images of the components (e.g. four sides of the column, including bottom to top), and post-process the collected images for vision-based structural inspection. Because this problem setting is not identical to the general autonomous navigation problems, the problem needs to be formulated specifically for this research to apply the rich and powerful techniques developed to date.

2.4 Automated vision data collection for structural inspection using UAVs

Currently, the primary means of automation for the vision data collection is the GPS-aided waypoint navigation (or simply waypoint navigation), where the UAVs automatically follow predetermined paths defined by GPS coordinates. Paths can include UAV actions at each waypoint, such as capturing an image and recording a video. The waypoint navigation functionality is readily available in many commercial platforms (e.g., Litchi for DJI drones [114], FreeFlight for Parrot drones [115] etc.). By setting appropriate paths, the UAVs can fly to the fixed location near the target bridge without human's in-flight interference.

From February 2018, Kintetsu railway Co., Ltd, Canon Marketing Japan Inc., and KDDI corporation in Japan have launched a project, where a UAVs operated in 4G LTE network is demonstrated as a tool for inspecting facilities along 500 km of Kintetsu railway routes immediately after earthquakes [116].

Unlike aerial imagery and surveillance applications (e.g. [117]), the application of the waypoint navigation to the bridge inspection tasks causes two problems: (i) image data for the bridge inspection purposes frequently need to be taken underneath the bridge, where GPS signals are not reliable, (ii) the GPS data typically comes with an error of few meters [118], and the accuracy is lower in the environment near buildings and bridges because of the multi-path error.

Because of the difficulties mentioned above, manual navigation of UAVs for bridge inspections are necessary even when the waypoint navigation is integrated into the flight plan [119].

Investigation of autonomous robotic navigation strategies for the structural inspection is relatively sparse in the literature, primarily because of the need for the customized formulation to solve the problem. For example, autonomous creation of the complete map of an unstructured environment (exploration problem) has been investigated extensively [111], [113], [120]–[122]. On the other hand, the approaches are not directly applicable to this research, because the direction that should be given to the robots is “collect close-up images of critical structural components as quickly as possible”, rather than “build a complete map of the environment”. To investigate autonomous robotic navigation in the structural inspection and other related scenarios, different problem formulations have been proposed. One of the approaches for the problem formulation is to derive the navigation strategies using accurate pre-built maps of the environment and the paths defined using the maps. Sheng et al. [123] investigates aircraft rivet inspection using an autonomous crawler robot that visits all target rivets by following the path defined with respect to the computer-aided design (CAD) model. Ibrahim et al. [124] investigates indoor construction progress monitoring using a building information model (BIM)-driven autonomous unmanned ground vehicle (UGV). Another approach for the problem formulation is to encode the logic that characterizes the desired tasks. Stokkeland et al. [125] investigates autonomous navigation strategies for the UAV-based inspection of wind turbines, where a UAV is directed to recognize the shaft and blades using machine vision techniques (Hough transform and post-processing using prior knowledge about the angles of/between lines), and then tracking the blades by the Kalman filter. Hui et al. [126] proposes vision-based UAV navigation strategies for power line inspection, where the transmission tower is detected using the Faster R-CNN object detection algorithm [68],

and power lines are recognized as parallel lines. The UAV is then directed to point the camera to the tower center, and flies along a path parallel to the power lines toward their vanishing point. Higuti et al. [127] proposes light detection and ranging (LiDAR)-based autonomous navigation strategies for a small ground robot to perform phenotyping by following the rows of canopy-like spaces between crop lines in agricultural fields. To deal with the noisy measurement data caused by frequent occlusions by hanging leaves, the estimations of the crop lines are refined based on the prior knowledge about the line width, robot width etc. Those logic-based approaches do not need complete pre-built maps of the environment and tend to have simpler initialization steps (complete models are not aligned explicitly), while the perception and path planning steps are versatile and dependent on specific application scenarios. Consequently, new sets of logic need to be developed to extend those approaches to the problem of autonomous rapid post-earthquake structural inspection.

This research assumes a specific subtask during the bridge inspection, i.e., inspecting four sides of RC columns of railway viaducts, from the bottom to the top. This assumption simplifies the formulation of the navigation goals, and changing the problem setting from the navigation in completely unknown environments to the navigation in environments where prior knowledge about the structure is available.

2.5 Automated vision data evaluation for structural inspection

In the civil engineering society, automated vision-based methods have been frequently applied in three major stages of structural assessment: structural component recognition, local damage evaluation, and semi-global to global damage evaluation. For each stage, vision data is

either directly used by image processing algorithms or used to create 3D point clouds to represent 3D geometric information of the structures.

One of the early proposals for the automated vision-based structural inspection with structural component recognition was made by [128]. Within the framework, [129] proposed a method for reinforced concrete (RC) column recognition based on line segment detection. To distinguish line segments making up boundaries of RC columns from other irrelevant line segments, the method applies thresholds and picks up near-parallel line segment pairs with a predetermined range of the aspect ratio. The method was applied to 20 test images, and detected 38 the true positive (TP), 7 false positive (FP), and 13 false negative (FN) columns (84.4% average precision and 74.5 average recall). Despite the simplicity, the method strongly depends on the values of thresholds, and tends to fail to find partially occluded or relatively far columns. Also, the method recognizes any line segment pairs satisfying the thresholds as columns; without further contextual understanding, the method is not guaranteed to work robustly for more complex images.

More recently, approaches based on deep learning algorithms have been investigated to detect and localize structural components accurately and robustly. A framework for the automated structural damage assessment discussed in [130] has four steps with different levels: Metadata filtering, Scene classification, Object detection, and Damage evaluation. As a pilot study toward the framework, the study showed the results of building collapse classification using CNNs with the VGG architecture [54]. Yeum et al. [131] investigated a binary structural component recognition problem (Beams/Columns vs Walls) as part of their structural damage recognition framework. To identify the types of main structural components in the input image and localize the components, the study finetuned VGG-type CNNs [54] using 2000 images with image-wise labels. After getting image classification accuracy of 99% for beams/columns and 86% for walls,

the approximate localization of the components were identified using class activation map (CAM) [132], [133]. The visual observation of the results showed that the main structural components in images were highlighted as blob-like patterns in the CAM.

The structural component recognition problem has also been investigated using 3D point cloud data. For the semantic as-built representations of structures, 3D point cloud data are segmented using contextual and geometric features [134]–[136], and the material labels are assigned [137], sometimes with the help of prior knowledge available from as-planned models [138]. A top-down approach based on geometric preliminary knowledge of RC girder bridges has been proposed in [139].

For the local evaluation of structural damage, application of image processing methods has been actively investigated. In earlier research, detection of structural damage has been attempted using local contextual features, such as crack detection using edges [140]–[144] and an entropy-based approach for concrete spalling detection and quantification [145]. These damage detection approaches use local contextual information only, the approaches are prone to errors, often raising a significant number of false-positive alarms [143].

More recently, approaches using CNN-based methods have gained popularity because of their capability of learning complex and high-level patterns in an end-to-end fashion [138], [146], [147]. Many research efforts focus on images of structural surface viewed from close or controlled distances, or in other words, the decision is binary, i.e. damaged surface or intact surface. For example, Cha et al. [148] applies convolutional neural networks to sub-images to detect and localize concrete cracks. Xu et al. [149] investigates steel crack recognition problem by sub-image classification approach using the CNN improved to leverage multi-scale features (Fusion CNN). A publicly available dataset of annotated images of concrete cracks has been developed to support

the investigation of visual recognition approaches [150]. Following the promising results of such efforts, the problem has been generalized to more complex scenarios, where the input image is not necessarily the close-up view of the target structural surface. Yeum [151] has performed one of the pioneering work in this category, where an object detection algorithm (R-CNN [152]) is applied to find concrete spalling in images of complex scenes of structures. [153] uses the Faster R-CNN object detection algorithm [68] to recognize steel corrosion (two severity levels), steel delamination, bolt corrosion, and concrete cracks. [154] investigates a four-class image classification problem for the damage states of reinforced concrete columns, as well as the exposed rebar recognition using object detection algorithm (YOLOv2 [70]). Application of semantic segmentation algorithms has also been investigated to obtain the maps of structural damage at the resolution of the input image. Liang [155] applies the SegNet algorithm to identify concrete damage in the images of bridges. To obtain high accuracy even for the images of complex scenes, Hoskere et al. [156] combines two semantic segmentation tasks, segmentations based on material types and damage types, and applies a multi-task FCN with the homoscedastic loss function. Damage detection results obtained by the semantic segmentation algorithm have been mapped to 3D point cloud data collected by LiDAR for further quantification [157].

Application of 2D methods to the damage evaluation of entire structures typically takes the form of collapse classification. Yeum et al. [130] performed binary collapse classification (collapse vs non-collapse) using CNNs. However, the scope of the work was limited to the binary evaluation of whether the buildings are damaged or not. Xiong et al. [158] applies the similar image classification approach to image parts segmented as buildings. For the detection and quantification of damage on the concrete surface from the entire structures, a framework including 2D column recognition and local damage evaluation methods is discussed [128]. The framework

integrates local context-based methods for structural component recognition and damage recognition [129], [141], [145] to evaluate the conditions of the entire structure. Existing work about local damage recognition is well aware of the importance of the detailed evaluation of the entire structure, and therefore discusses how their contributions can be fit into the larger frameworks of structural condition assessment (e.g. [131], [146], [154], [155]).

Automated crack evaluation of the surface of concrete members using 3D data has also been proposed [159]. The method relies on dense point cloud to generate mesh and surface normal, which is then used to find and characterize the crack. For the 3D point cloud-based methods to be successful, the point cloud data need to be dense and accurate enough to represent the geometry of damage, such as cracks and spalling. Therefore, point cloud data from different sources are often used as an input to the damage detection algorithms (e.g., laser scanner [160], [161]).

The evaluation of the entire structure based on 3D point cloud data has also been investigated in a project called “the Aerial Robotic Infrastructure Analyst (ARIA)” [162]. The scenario drawn for the project begins with the use of a UAV equipped with a laser scanner and three video cameras to build a dense 3D point cloud of the target bridge. Then, the semantically rich 3D model of the structure is constructed (based on the method described in [134]). The 3D geometric information of the structure, together with the semantic labels of the structural components, can be used to assess structural damage based on the local method [161]. Finally, the semantically rich 3D model is transformed to the finite element model (FEM) for the further evaluation, although the automated conversion from semantic model to the FEM has not been solved fully. The framework demonstrated in the project is applicable to general types of bridges for the comprehensive assessment of structural damage, including damage detection, localization, and quantification.

Compared to the ARIA project that focuses on general types of civil infrastructure, the railroad bridges investigated in this study typically have simple structures. For such structures, the mode of structural damage potentially occurring after earthquakes can be predicted with high accuracy, i.e., cracks and spalling on the surface of concrete columns [163]. This assumption of the study implies that, if the UAVs can directly navigate to the bridge columns and take close-up images of the member surfaces, 2D local to semi-global methods are expected to be applied to make the preliminary safety evaluation of the entire structure, without constructing complete dense 3D models with the geometric characterization of every damage.

CHAPTER 3: DEVELOPMENT OF SYNTHETIC ENVIRONMENT OF REINFORCED CONCRETE RAILWAY BRIDGES

This chapter discusses the development of synthetic environments to enable efficient development, implementation, and testing of vision-based approaches for automated rapid inspection of RC railway bridges.

3.1 Roles of synthetic environment

The synthetic environments are important for the investigation of autonomous bridge inspection systems for the following three reasons:

Efficient experiments: Conducting UAV surveys in the field is challenging and time consuming; in contrast, surveys in the synthetic environment can be conducted at any time and under a wide variety of environmental and usage conditions.

Training data generation: A significant bottleneck of the application of computer vision and machine learning techniques to the automated inspection of civil infrastructure is obtaining sufficient labeled images for training CNN-based algorithms with large number of parameters. Typically, training data is generated manually, that is, a tedious and labor-intensive approach; in contrast, images from the synthetic environment can be labeled with significantly reduced efforts.

Ground truth measurement: The ground truth measurement data is available in the synthetic environment, such as ground truth depth map and ground truth damage state of the structure. Therefore, we can evaluate the performance of post-processing algorithms in a straightforward and quantitative manner.

The synthetic environments developed for this research is most inspired by the synthetic environment investigated to perform the recognition of scene objects for autonomous driving

purposes [62]. The study has shown that the semantic segmentation accuracy improves significantly by including 13.4K synthetic images into 200-300 real-world images. The accuracy improvement reaches more than 10% in some test cases. To bring the benefit of synthetic environment into the visual recognition tasks for civil infrastructure condition assessment, the author of this dissertation has investigated approaches to model civil infrastructure in various structural and damage conditions. In particular, Hoskere et al. [164], [165] have proposed a modeling approach, termed physics-based graphics models (PBGMs), in which damage hot spots identified by the finite element analysis are used to realize photo-realistic synthetic damage scenarios. The synthetic environment and the PBGMs have been first applied to the semantic segmentation of steel surface damage (e.g. corrosion) of navigation infrastructure on the U.S. inland waterways, and later extended to displacement and strain measurement of miter gates [166] and experimental bridge structure [167]. These studies have shown significant potential of facilitating civil infrastructure condition assessment by augmenting training data for visual recognition algorithms, as well as providing means of the quantitative evaluation of algorithm performance.

To take the full advantage of the capabilities of the synthetic environments discussed herein, this research has developed two environments: Unity game environment [16] to simulate video data during UAV flights, and a set of synthetic environments developed using Blender software [17] to enable in-depth investigation of component localization and mapping, as well as structural inspection for a special type of RC rigid frame railway bridges, that is, viaducts. The following sections discuss each of the two synthetic environments in detail.

3.2 Game environment to simulate UAV-based video data collection

A synthetic video dataset imitating random UAV flights around concrete girder bridges has been created using the Unity3D game environment [16][168]. The environment contains a series of photo-realistic RC girder bridges, as well as other objects typically encountered in urban scenes, such as cars, roads, buildings, fences, and poles. After creating photo-realistic models of the bridges and other objects, a simulated UAV is flown in the environment with random changes of the heading, pitch, and altitude. The UAV records video data throughout the flight to generate the dataset that contains a variety of perspectives and their changes.

The random navigation of the UAV is implemented in the following steps. First, a waypoint is randomly selected within predetermined bounding boxes in the two-dimensional ground plane. Two levels of bounding boxes are defined to imitate variety of viewpoint changes encountered during actual bridge inspections. The inner bounding box is a tight bound of the bridges, and the navigation within the bounding box corresponds to viewing different parts of the bridge closely. An outer bounding box is defined as well, which is sometimes used to imitate the inspectors' action of moving away from the bridge to view the global structure. During the UAV navigation, 2D waypoints are selected from the inner bounding box with 95% probability and from the outer bounding box from the 5% probability. After selecting the waypoint, a 2D collision-free path to the waypoint is created using the Unity3D built-in A* algorithm.

While the agent navigates in 2D plane, camera height and pitch relative to the 2D path in the ground plane are changed randomly and independently. For the camera height, a waypoint is randomly selected between 1.0 m and 4.0 m. Then, the camera height is updated using a simple proportional controller (velocity above 0.05m/frame is clipped). Similarly, pitch angle waypoints are randomly selected between -90° (looking up vertically) and 45° (looking down), and the

camera pose is updated using a simple proportional controller. When the camera height is within 0.3m from the waypoint for 200 consecutive frames, the height waypoint is updated. Similarly, the pitch waypoint is updated when the camera pitch is within 10° from the waypoint for 200 consecutive frames.

The data during the UAV flight is collected using three cameras with the same position and orientation. The first camera records the RGB image frames. The second camera records the corresponding ground truth bridge component labels. The third camera records the corresponding ground truth depth map. The example frames of the video are shown in Figure 13. The resolution of the video was set to 240×320 , and 37,081 training images and 2000 testing images are generated in this environment.

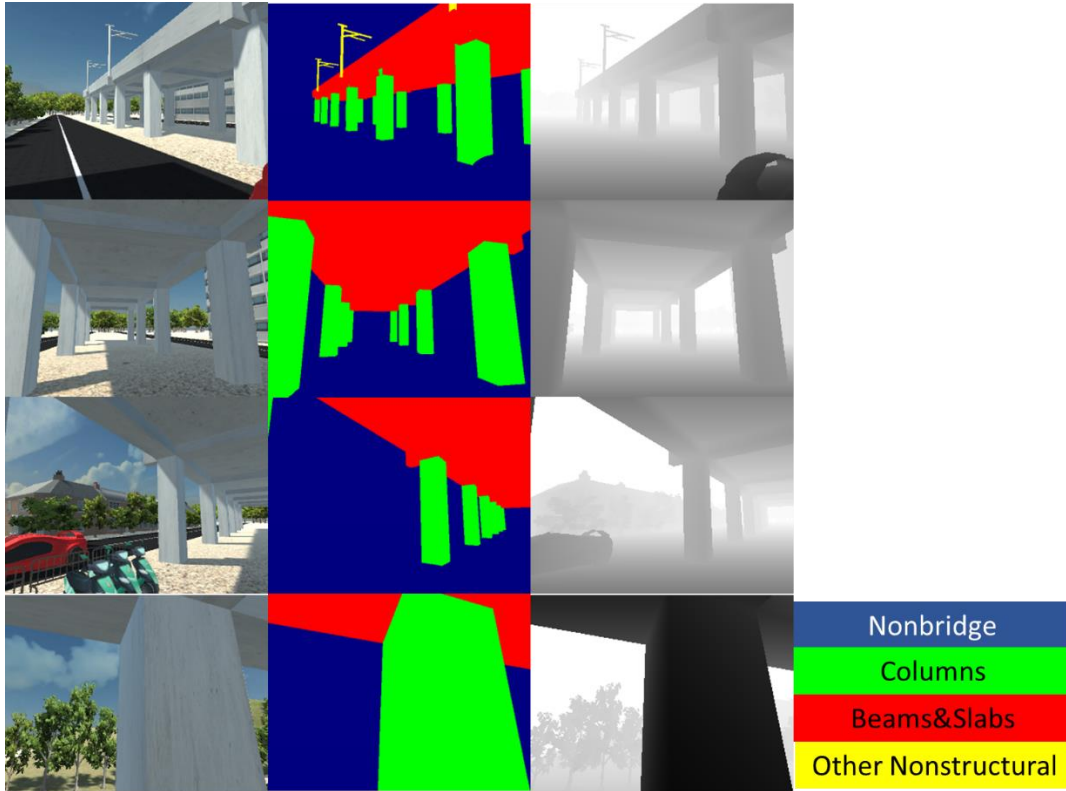


Figure 13 Example frames of the new video dataset with ground truth labels and ground truth depth maps

3.3 Synthetic environment of RC railway viaducts

This synthetic environment has been developed to extend the environment discussed in the section 3.2 to the highly generalizable one that can be used to train algorithms for structural component recognition, damage recognition, and depth map estimation. A special type of RC railway bridges is modeled in this environment: RC viaducts of the Tokaido Shinkansen (one of Japanese high-speed railway lines operated and maintained by the Central Japan Railway Company). The viaducts are designed by following the “standard design” procedure, which had been developed as simple and systematic steps to determine the geometric details using a few parameters, including slab height and track curvature [169]. When the Tokaido Shinkansen was constructed, such viaducts were mass-produced, comprising the vast majority of the 116 km of railway bridges of the 515 km-length Tokaido Shinkansen line [170]. Besides the generic geometry, this type of structures bring in an additional benefit for the investigation of damage recognition problems: major structural damage to the RC viaducts after earthquakes are cracks, spalling, and exposed rebar on the column surfaces [171], [172]. The synthetic environment developed in this research consists of 2,000 viaducts with random geometry realized by the standard design procedure, as well as random damage scenarios (concrete cracks, spalling, and exposed rebar) of the viaduct columns. The synthetic environment is used to produce a dataset of 8,715 images for structural component recognition and depth estimation, as well as 8,233 images for damage recognition. The image resolution is $1,920 \times 1,080$, and each image is associated with ground truth pixel-wise information of structural component types, damage types, the depth values (Figure 14).

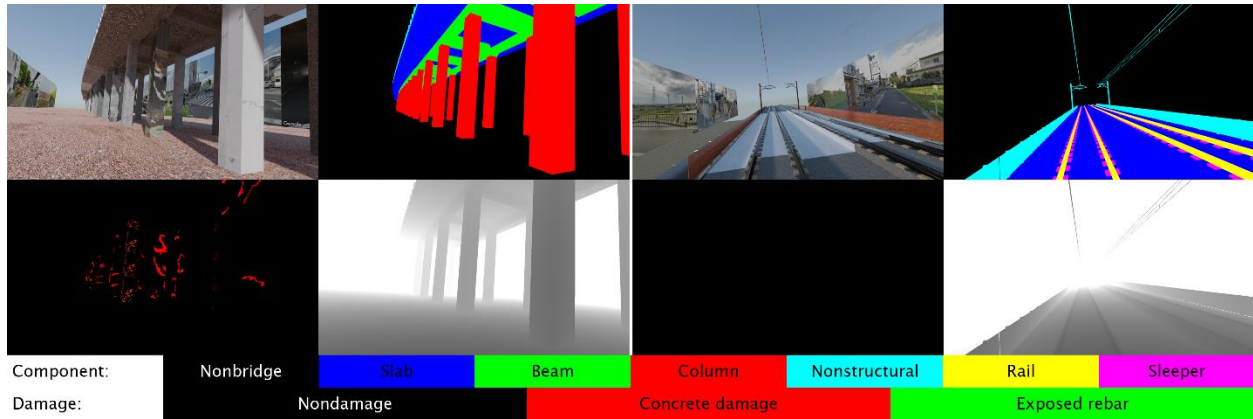


Figure 14 Example images (top left), structural component label maps (top right), damage maps (bottom left), and depth maps (bottom right) from the synthetic Tokaido dataset

The next section discusses the steps to develop the synthetic environment for civil infrastructure condition assessment. Then, the synthetic environment of the Japanese high-speed railway viaducts is developed, and the detail of the produced dataset (Tokaido dataset) is described.

3.3.1 Steps of developing the synthetic environment

This section describes the steps of developing synthetic environment that can produce training data for visual recognition tasks related to civil infrastructure condition assessment. The overview of the steps is depicted in Figure 15. First, meshes are created to represent the geometry of the target structure (Step 1 in Figure 15). Then, appropriate textures for each part of the meshes are determined (Step 2 in Figure 15). A simple method for this step is to specify images that contain information about mesh surface colors. Alternatively, multiple images can be specified to control the detail of the mesh surface, such as color, roughness, metallic properties, and shading effect caused by uneven surface normal fields. In step 3, the selected images are assigned to the model by defining mappings between the images and the mesh surface. After the model of the target structure is created, synthetic cameras are placed to render images (Step 4 in Figure 15). At the same time, one can obtain ground truth information for the visual recognition tasks, such as ground

truth maps of structural component labels, ground truth maps of damage labels, and ground truth depth maps. The remaining part of this section discusses key concept and implementation procedure of each step presented in Figure 15.

Blender 3D computer graphics modeling software is used to implement the modeling steps discussed herein. Besides, the Blender-Python API [173] is used to automate repeated Blender operations, so that many random viaducts and damage scenarios can be produced efficiently.

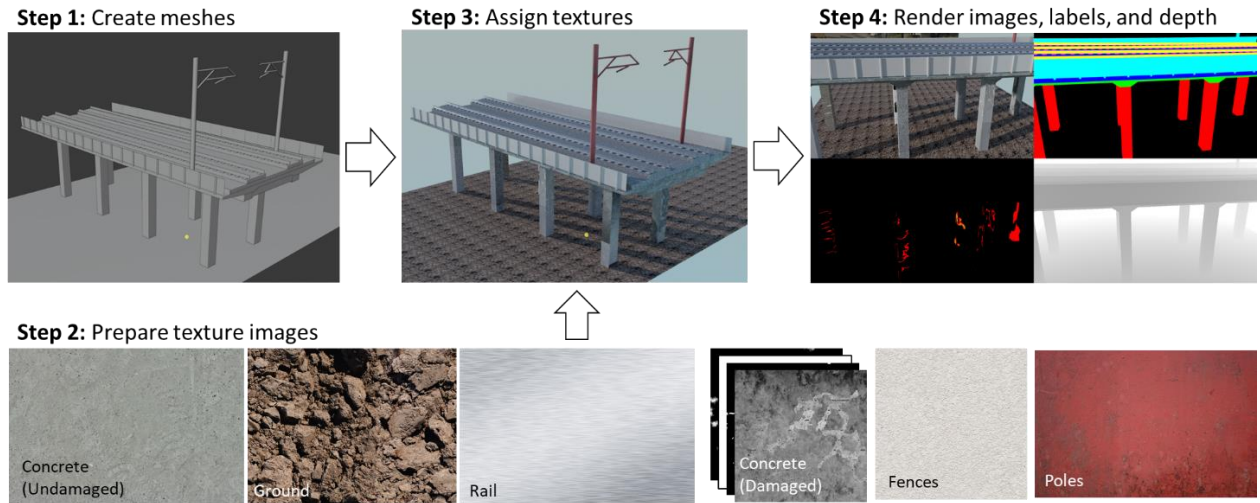


Figure 15 Overview of the computer graphics approach for civil infrastructure condition assessment

Mesh creation: The first step of developing the synthetic environment is to define mesh that specifies the geometry of structures. In Blender, mesh can be created either manually by combining basic operations, such as adding primitive shapes, scaling, moving, and extruding, or programmatically by directly specifying mesh vertex coordinates and faces. This research defines the mesh directly using Python to enable the random creation of railway viaduct models.

Texture image preparation: A simple method for defining a texture is to specify an image that represents the color of the mesh surface. To define realistic textures that are appropriate for the random creation of railway viaduct models, this research develops a repository of images of

concrete (575 images), steel (564 images), painted surface (283 images), and ground (491 images), downloaded from the Internet using the Bulk Bing Image Downloader (BBID) [174]. To create the image repository of surrounding scenes, the google street view API (“Google Street View API,” n.d.) is used to download 223 images along a route from Kyoto to Osaka, Japan. When a texture in each category is required by the model, an image is sampled randomly from the corresponding category of the repository.

A more advanced approach for texturing is physically-based rendering [175]. In this approach, various properties of the mesh surface, as well as RGB color, can be controlled by specifying multiple texture images corresponding to each of those properties. In particular, this research controls the following four properties to generate realistic textures of damage to the bridge columns: color, roughness, metallic properties, and surface normal field. The effect of controlling each of those properties is shown in Figure 16 for sample texture images available at textures.com. By using the RGB image and the metallic map, the proportion of reflection and refraction when the light hit the object surface can be specified: large metallic value (white color) assigns highly reflective surface (metal), while the light passes through the surface (refraction), scattered, and re-emitted outward when the small value is specified (dark color). The roughness map controls a statistical model of the micro-structure of the target surface termed microfacets [177]. The small value (dark color) specifies smooth surface, while the light is scattered more when the roughness value is high (white color). Finally, geometric structure of the target surface is modeled by defining a surface normal map, based on which the shading on the target surface is adjusted to add visual effects that represents the geometry. In Figure 16, the visual effect of the protrusions is attained by controlling the shading on the surface this way, rather than deforming the mesh, an approach that is significantly more expensive computationally.

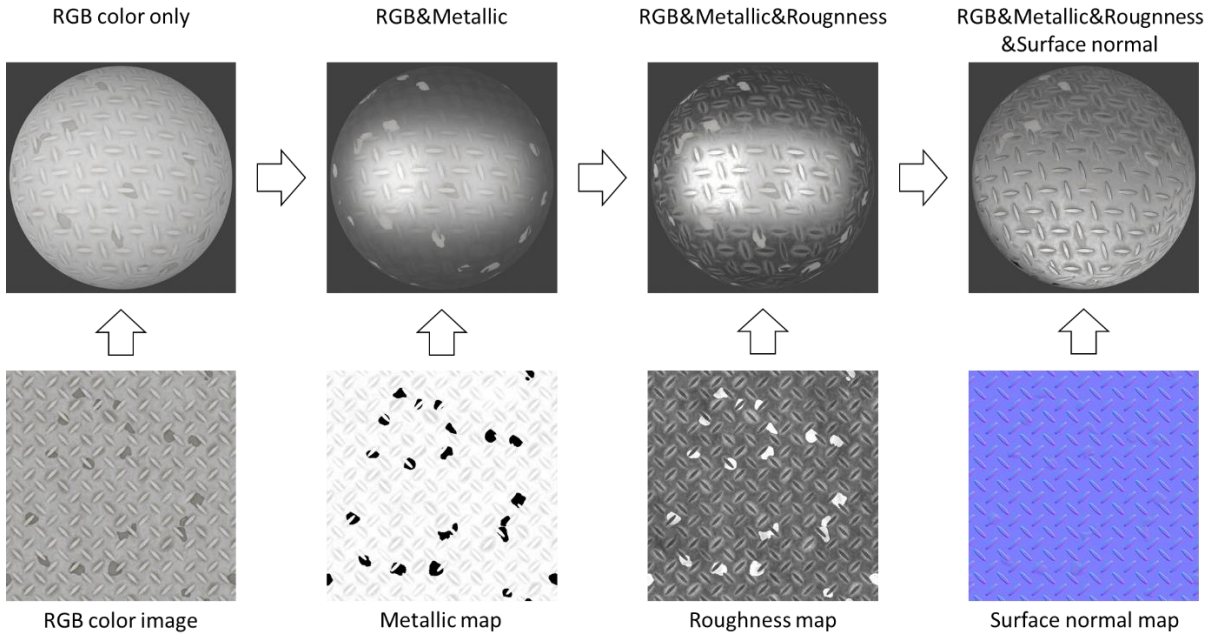


Figure 16 Steps of physically-based rendering

Texture assignment: The defined textures can be assigned to the surface of the mesh using the Blender Principled BSDF node (Figure 17) by either specifying a default value or connecting prepared images (read by the Image Texture node) to the appropriate entry of the node. Besides examples shown in Figure 16, an example of applying the Principled BSDF node to generate concrete crack model is presented in Figure 18, where steps discussed in a Blender tutorial [178] are implemented. The method generates noise texture, and successively applies thresholding and scaling to create a displacement map that can be converted to the surface normal map using the Blender Bump node. At the same time, the method defines different diffuse colors between the damaged and undamaged regions to enhance the reality. The shapes and the appearances of the cracks can be controlled by adjusting parameters of noise generation, thresholding, scaling, and the textures of damaged and undamaged regions.

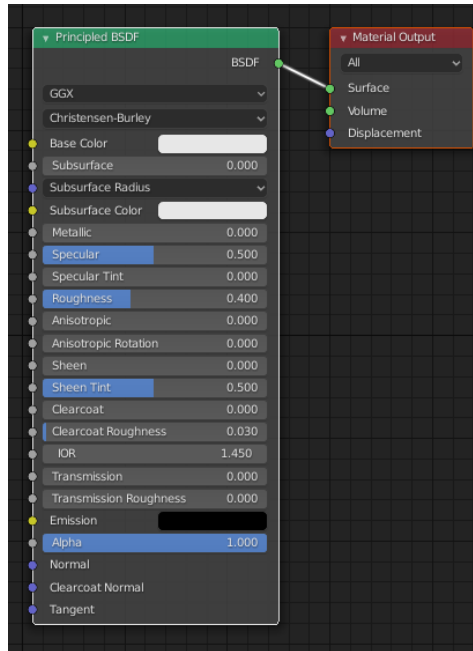


Figure 17 Blender principled BSDF node

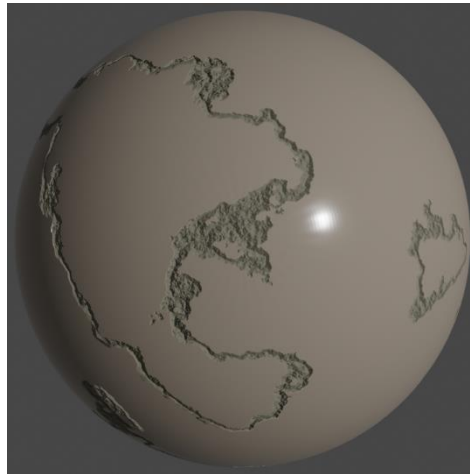


Figure 18 Concrete crack model created by steps presented in (“Blender Video Tutorials,” n.d.)

Rendering images, labels, and depth: Once the synthetic models of the structures are created, synthetic cameras are placed to render photo-realistic images. At the same time, ground truth label maps for structural components and damage can be rendered by assigning additional textures that contains such information (e.g. pure red color can be assigned to the mesh surface belonging to bridge columns). To eliminate unwanted shading effects in the label maps, Blender

emission shader is used, instead of the principled BSDF. The corresponding ground truth depth maps can be obtained using Blender compositing functionality.

3.3.2 Mesh generation based on the standard design

Typical RC viaducts with the standard design are shown in Figure 19. Each viaduct is 24 m, comprising of three central spans (6 m each) and cantilever-type end spans (3 m each). The detailed design of a viaduct is determined based on three parameters: loading conditions, viaduct height, and geotechnical conditions. The loading conditions of viaducts are classified based on their trajectories: straight trajectories when the radius of curvature is larger than 5,000 m, and curved trajectories when the radius of curvature is smaller than 5,000 m. Viaducts with curved trajectories are stiffer in the transverse directions and have 20% more rebars than those with straight trajectories. The dimensions of cross-sections of viaduct columns and intermediate beams are presented in Table 2 for different loading conditions and viaduct heights. Geotechnical conditions affect the design of footing underground, which is not modeled in this research. The readers are directed to the original Japanese literature for other dimensions and the further detail of the design concept [169].



Figure 19 Google street view image of typical RC viaducts with the standard design

Table 2 Dimensions of structural component cross sections [$cm \times cm$] [169]
 Straight: radius of curvature is larger than 5,000m. Curved: radius of curvature is smaller than 5,000m,

Viaduct height $H[m]$	Columns (Straight)	Intermediate beams (Straight)	Columns (Curved)	Intermediate beams (Curved)
$5.5 < H \leq 7$	60×60	None	60×70	None
$7 < H \leq 8.5$	70×70	None	70×80	None
$8.5 < H \leq 10$	80×80	None	80×90	None
$10 < H \leq 12$	70×70	80×60	70×85	80×60
$12 < H \leq 14$	80×80	80×60	80×95	80×60

This research first generates random viaduct trajectories, from which viaduct meshes are generated following the standard design procedure. The steps to generate a trajectory are described below:

- (i) Determine the number of viaducts N and the initial viaduct height. In this research, $N = 10$, and the initial height is sampled uniformly from the range [6.5m, 12m].
- (ii) Generate a random trajectory using the inverse Fast Fourier Transform (FFT) of rectangular spectra with uniform random phase (the cutoff frequency is $2/N$). The trajectory is defined using three noise sequences that represent ground height increment, viaduct height increment, and the curvature of the viaduct centerline (inverse of the radius of curvature). The amplitudes of the FFT spectra are set to 1.0 for the ground height increment and 0.2 for the viaduct height increment. The noise sequence for the viaduct curvature is scaled, so that the minimum radius of curvature along the trajectory is a value sampled from a uniform distribution in the range [500,5000].
- (iii) Apply constraints to the trajectory. Based on the typical values mentioned in the existing literature [169], [179], this research defines the constraints as shown in Table 3.

Table 3 Constraints used in this research

Viaduct Height $H[m]$	$5.0 \leq H \leq 14$
Viaduct slope S	$ S < 0.02$

Following the steps (i)-(iii), 200 trajectories of 10 viaducts are generated (2,000 viaducts in total). The histograms of viaduct height and curvature are shown in Figure 20. With the sampling scheme implemented in this research, the height and curvature of the generated viaducts span the range of those parameters of actual structures realized by the standard design.

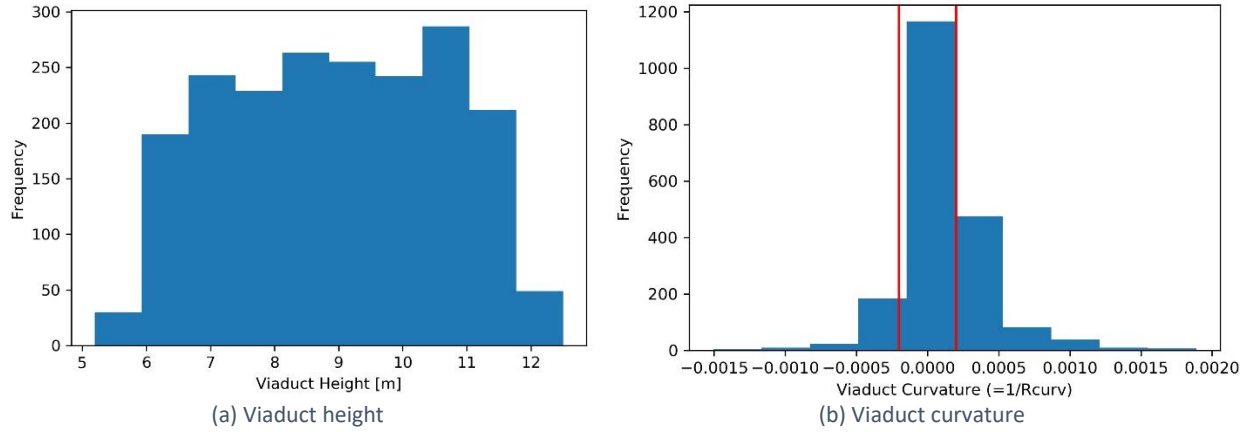


Figure 20 Histograms of viaduct height and curvature. Red lines separate straight and curved trajectories.

Information about the design of the superstructure of this type of viaducts, such as tracks, poles, and wires, is relatively sparse in the literature. To supplement the information, this research refers to literature about other railway lines in Japan as well [180]–[183]. The modeling accuracy need to be improved to enable further investigation of inspection and monitoring tasks that require precise information of those components, which is part of the future work.

3.3.3 Random texture generation and assignment

Once the meshes of the viaducts are generated, surface textures are defined using the images in the repository discussed in section 0. The texturing process for each component of the synthetic environment is as follows:

Slabs, beams, and undamaged columns. One concrete texture image is selected randomly for each slab, beam, or undamaged column, and the selected image is assigned using the Principled

BSDF node, where the “Metallic” and “Roughness” parameters are set to 0.0 and 1.0, respectively. Moreover, the scale of the texture is sampled uniformly from the range [0.1,1.0].

Sleepers. For each viaduct, sleeper material is sampled from concrete, steel, and wood materials with equal probabilities. Then, one image is selected randomly from the images of the selected material. For concrete and wood materials, the Metallic and Roughness parameters of the principled BSDF block are set to 0.0 and 1.0, respectively. For steel materials, the Metallic parameter is set to 1.0, and the Roughness parameter is sampled uniformly from the range [0, 1]. The scale of the texture is sampled uniformly from the range [0.1,1.0].

Rails and wires. One image is sampled randomly from the set of steel images. The other parameter setting follows the same steps as that of steel materials for sleepers.

Fences and poles. One image is sampled randomly from the set of painted surface images. Both the Metallic and the Roughness parameters are sampled uniformly from the range [0, 1]. The scale of the texture is sampled uniformly from the range [0.1,1.0].

Ground. One image is sampled randomly from the set of ground images. The Metallic and the Roughness parameters are set to 0.0 and 1.0, respectively, and the texture scale is sampled randomly from the range [0, 0.1].

Surrounding scene. The modeling of the surrounding environment of the viaducts is simplified in this research, considering that the focus of the research is on the viaducts: the surrounding environment is modeled using panels of outdoor scene images placed at the distance of 30 m from the viaduct centerline. An image of the surrounding scene is selected randomly from the repository for each panel. To avoid shading caused by the panels, emission shader is used with low intensity values (0.5).

3.3.4 Random damage texture generation and assignment

This research extends the method discussed in [178] to generate random damage to the RC viaduct columns. The idea of texture generation for concrete damage is shown in Figure 21. The cracks shown in Figure 18 can be characterized by the following three properties: (i) random geometry realized by thresholding and scaling noise textures, (ii) displacement texture that induces shading effects that are characteristic to concrete damage, and (iii) texture discontinuity that represents the texture difference between the exterior surface (typically subject to dirt, dust, paint etc.) and the fresh concrete surface exposed by the removal of the exterior surface concrete. This research *defines* the synthetic concrete damage as the surface regions with all the three properties, and randomly sample parameters of those properties to realize concrete damage textures. Note that the defined concrete damage is not always realistic, as shown in Figure 21. Instead of making the set of defined damage as close as possible to the set of realistic damage, the dataset is designed such that the defined damage covers the realistic damage, and therefore reliable detection of the defined damage leads to the reliable detection of the realistic damage.

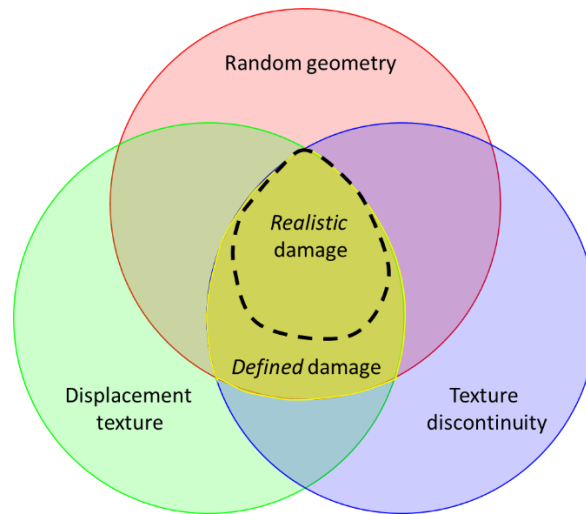


Figure 21 Concrete damage texture generation concept

The implementation procedure for concrete damage texture generation is shown in Figure 22. First, noise texture of the resolution $1,080 \times 1,080$ is generated using Perlin simplex noise [184], where the number of octaves is 8, and the frequency parameters are sampled uniformly between 20% and 200% of the texture dimensions. After taking the absolute values, the noise is clipped at the predetermined threshold value to get the synthetic crack profile. Mathematically, these operations can be written as

$$\mathbf{P}_c^- = \min \left(t, \text{abs} \left(\mathbf{N}(\text{freq}_x, \text{freq}_y, \text{oct}) \right) \right) \quad (1)$$

where $\mathbf{N}(\text{freq}_x, \text{freq}_y, \text{oct})$ is a $1,080 \times 1,080$ noise texture parameterized by frequency parameters in image X- and Y-directions, $\text{freq}_x, \text{freq}_y$, and the number of octaves, oct , and t is the threshold value. $\mathbf{P}_c^-$ is a map with resolution $1,080 \times 1,080$ that represents the geometric profile of the generated concrete damage. The profile consists of the intact surface specified as the value t , and the damaged regions where the geometry is expressed by the offset from the value t . The map $\mathbf{P}_c^-$ can be converted to the displacement map, $\mathbf{P}_c$, that are applied to mesh surface to represent concrete damage geometry (bottom of the second column of Figure 22) by

$$\mathbf{P}_c = d \cdot \text{normalize}(t - \mathbf{P}_c^-) \quad (2)$$

where d is the maximum depth of the concrete damage, and $\text{normalize}(\cdot)$ indicates the normalization operation to make the maximum value of the map one.

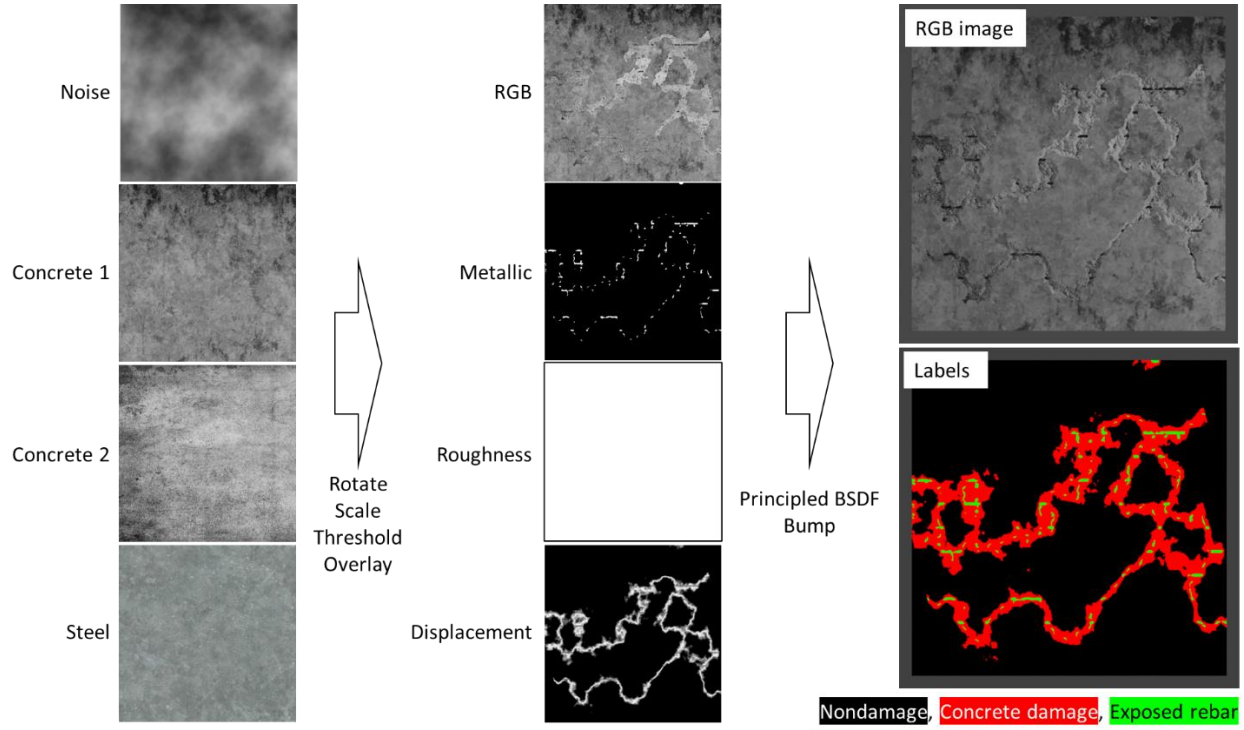


Figure 22 Random damage texture generation procedure

When the depth of the concrete damage is larger than the thickness of the cover concrete, rebar is exposed, affecting the structural assessment significantly. This research models the rebar following a concept similar to the one explained in Figure 21: the rebar is modeled such that the range of its design parameters cover the corresponding range of the actual rebar. With this concept, the geometry of the rebar is modeled using cylinders with sinusoidal surface ripples (ripple amplitude is 10% of the cylinder radius). The cylinder radii of the axial rebar and stirrups are 1/200 and 1/300 of the texture image dimension, respectively. To create rebar shape profiles, the thickness of the cover concrete for axial rebar, c_a , is first sampled uniformly from the range [2.5 cm, 7.5 cm], referring to the typical values for the actual design of concrete columns (ACI 318-14, 2014; Inaguma & Seki, 2004). Then, the cover thickness for the stirrups, c_s is determined by

$$c_s = c_a - 2 \text{ [cm]} \quad (3)$$

The intervals of the axial rebar and stirrups are sampled uniformly from the three and ten times the corresponding diameters. The shape profiles of the rebar are created based on these parameters and, when the concrete damage depth exceeds the depth of the rebar surface, the rebar shape profiles are used in place of the damaged concrete profile. The bottom row of the second column of Figure 22 shows the displacement profile after incorporating the rebar profile.

With the procedure discussed in this section, masks of the non-damage, concrete damage, and exposed rebar can be obtained. The RGB image and the metallic map used for the Blender principled BSDF block can be generated using the masks as follows:

(RGB-1) Two images are selected randomly from the set of concrete images, corresponding to the concrete texture in the damaged and undamaged regions.

(RGB-2) A single image is selected randomly from the set of steel images, corresponding to the exposed rebar.

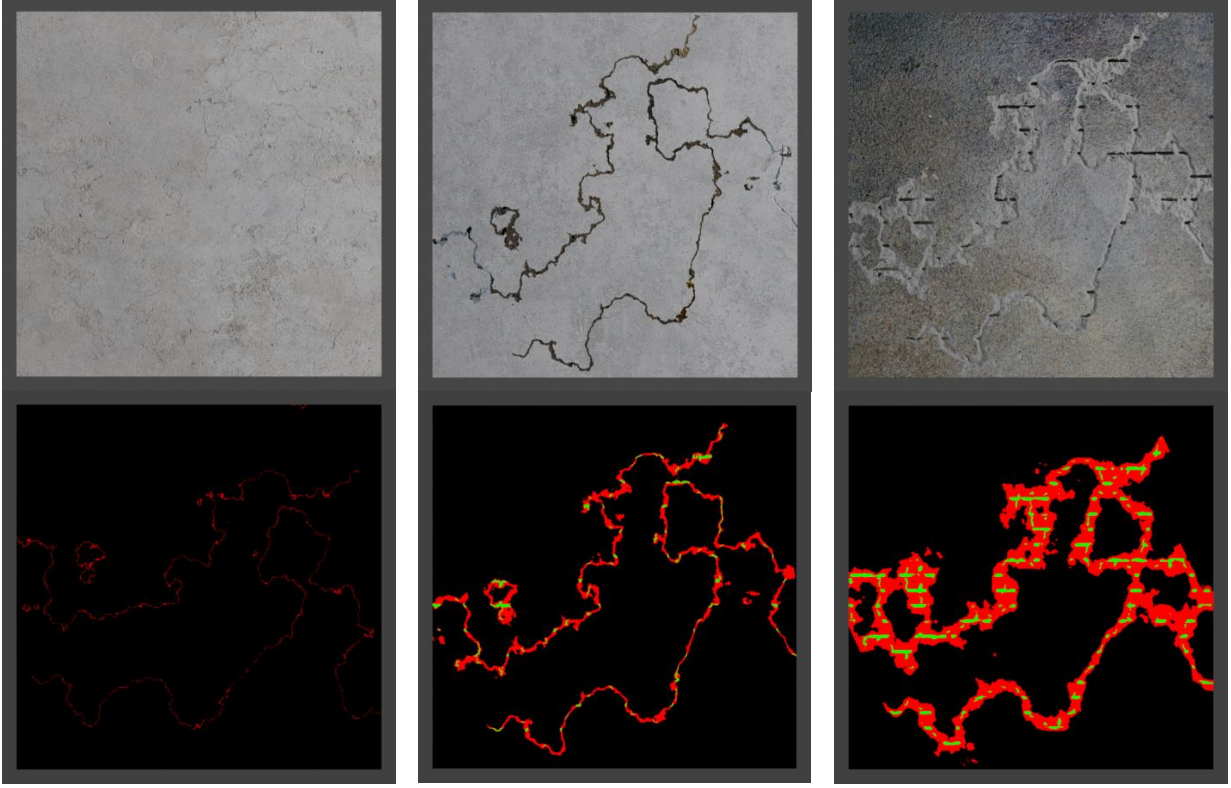
(RGB-3) The selected images are overlaid using the masks of nondamaged, concrete damage, and exposed rebar.

(Metallic) Set metallic to 1.0 for the regions of the exposed rebar, and 0.0 for other regions.

The first and the second images of the second column of Figure 22 show the RGB image and the metallic map generated for this example. The third image shows the roughness map, which is 1.0 everywhere. Once the RGB image, metallic map, roughness map, and displacement map are obtained, the damaged concrete texture can be defined and assigned following the steps discussed in Sections 0-0. An image rendered using a plate with the generated damaged concrete texture and the corresponding label map are shown in the right column of Figure 22.

The threshold t and the maximum depth parameter d control the severity of the damage. The area of concrete damage increases as t increases, and the depth of the concrete damage

increases as d increases, exposing more rebar. This research defines three damage severity categories (slight, medium, severe), and samples the values of those parameters using distributions defined for each category. Slight damage is designed such that the area of damage is between 0% and 1% of the total area (sampled uniformly). Once the ratio of the damaged area is determined, the corresponding value of t can be determined uniquely. The depth parameter d for the slight damage is sampled uniformly from the range [1cm, 2cm]. The range of the ratio of the damaged area for the medium damage is [90%, 99%], and the maximum depth is sampled from the range [1cm, 10cm]. For severe damage, the ratio of the damaged area is sampled from the range [70%, 90%], and the maximum depth is sampled from [2cm, 10cm]. Example damage textures generated using the same noise texture are shown in Figure 23 for each of the slight, medium, and severe categories.



(a) Damage area: 1%, d=2cm (b) Damage area: 5.5%, d=5.5cm (c) Damage area: 20%, d=6cm
Figure 23 Example images and labels of damage for different severity levels. (a) Slight, (b) Medium, and (c) Severe

Following the steps described in this section, 1,000 damage textures are generated for each of the slight, medium, and severe categories. The textures are assigned to the viaduct columns by the following rule: (i) a column is not damaged with probability 0.1 (texturing following the steps discussed in the section 3.3.3), (ii) a column is damaged with probability 0.9, (iii) for a damaged column, the damage category is chosen from the three categories with equal probability, (iv) once the damage category is determined, a damage texture is selected from the 1,000 textures generated by the steps discussed in this section, and (v) the selected texture is assigned to the column using the Blender Principled BSDF node and the Bump node, with texture scale sampled uniformly from the range [0.2, 1.0].

3.3.5 Synthetic data generation

A synthetic dataset for vision-based structural condition assessment of Japanese high-speed railway bridges, termed Tokaido dataset in this research, is created using 200 synthetic environments, each of which contains 10 viaducts generated randomly by the steps discussed in this section. For each environment, 100 synthetic cameras are defined by sampling focal length uniformly from the range [15mm, 55mm], and setting the sensor size and camera resolutions to 36mm and $1,920 \times 1,080$, respectively. The location of each camera is sampled uniformly from the space defined by the distance from the viaduct centerline, D , and height range $[h_0, h_1]$: $D = 10\text{m}$, $h_0 = 1\text{m}$, $h_1 = 10\text{m}$ for 50 cameras, and $D = 5\text{m}$, $h_0 = 1\text{m}$, $h_1 = 6.5\text{m}$ for the remaining 50 cameras. With this parameter setting, the first 50 cameras take images of the global structures as well as close-up images of the structural components (“regular images”), while the remaining 50 cameras take more close-up images of viaduct columns (“close-up images”). Camera rotations (in XYZ Euler angles used by default by the Blender software) are also sampled uniformly, such that the rotation about X and Y axes are in the ranges $[\pi/4, 3\pi/4]$ and $[-\pi/12, \pi/12]$, respectively. The rotation about Z axis is sampled from the range $[-\pi/2, \pi/2]$, measured from the direction that points to the viaduct. Once the RGB image is rendered, associated ground truth structural component label map, damage label map, and depth map are collected.

This process generates a dataset of 10,000 regular images and the same number of close-up images, as well as the associated ground truth maps. This raw dataset needs to be post-processed to remove unrealistic images: a camera may be placed too close to an object, or even inside the mesh of an object. This research applies the following criteria to filter out such unwanted images: (i) An image is removed when the average pixel intensity is less than 30, (ii) an image is removed if the image contains a single structural component label only, and (iii) an image is removed when the minimum depth from the camera is not larger than 50cm. After the image removal, the dataset

contains 8,715 regular images, 7,257 close-up images, and the associated ground truth maps. Examples of the RGB images, structural component label maps, damage label maps, and depth maps are shown in Figure 14 and Figure 24.

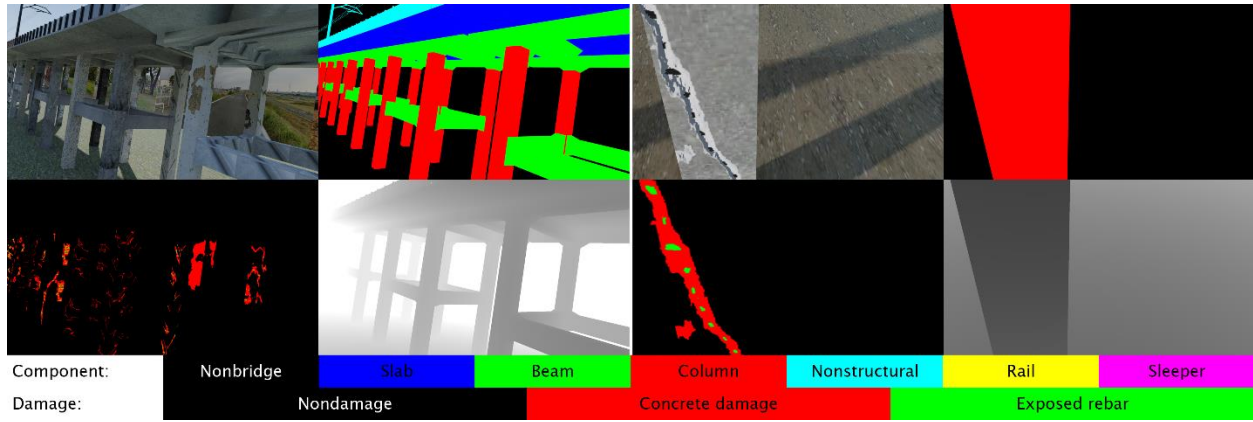


Figure 24 Additional example images (top left), structural component label maps (top right), damage label maps (bottom left), and depth maps (bottom right) from the Tokado dataset

In addition to the regular and close-up images, this research generates 3,000 images of damaged RC plates to further augment training data for damage recognition. The damage textures are obtained by the steps discussed in section 3.3.4, and the textures are applied to the plates without additional scaling. The plates are then rotated randomly, and photographed using synthetic cameras placed at random distances. The 3,000 images thus generated is termed “pure texture” image in this research. Examples of the pure texture images are shown in Figure 25. The composition of the Tokaido dataset is summarized in Table 4.

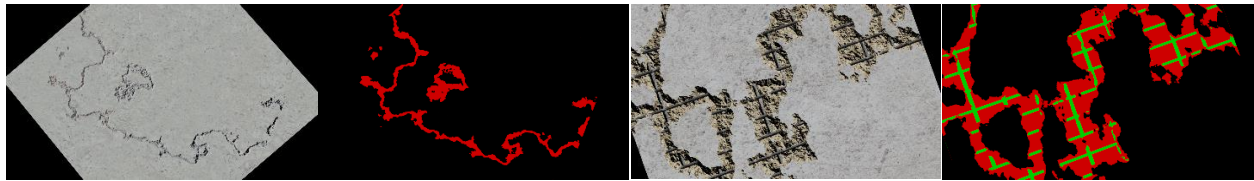


Figure 25 Example pure texture images and associated labels

Table 4 Composition of the Tokaido dataset

Number of images	Number of environments	Number of viaducts	Structural component labels	Damage labels	Depth range
8,715 (regular) 7,257 (close-up) 3,000 (pure texture)	200	2,000	Nonbridge, Columns, Beams, Slabs, Rails, Sleepers, Other nonstructural components	No Damage Concrete damage Exposed rebar	0.5m~30m

3.4 Summary and Discussions

In this chapter, two large-scale synthetic environments have been developed to investigate visual recognition tasks related to the structural condition assessment of RC railway bridges. The first environment is the game environment that simulates UAV-based video data collection. The second environment models a specific type of RC railway bridge, that is, RC railway viaducts with the standard design used by the Tokaido Shinkansen. The datasets produced in both environments contains images and the associated ground truth maps, such as structural component label maps, damage label maps, and depth maps. The synthetic environments are particularly useful to investigate vision-based structural condition assessment tasks, because obtaining such maps manually is labor-intensive and impractical in many application scenarios. The availability of the large-scale synthetic dataset facilitates the investigation of various state-of-the-art algorithms proposed for general semantic segmentation and depth recognition problems, such as SegNet [19], U-net [58], and its successors [59], [60], so that the structural engineering community can be fully benefitted from the recent advance of the computer vision and data science field.

While the Tokaido dataset is a powerful tool to improve the performance of visual recognition algorithms for civil infrastructure condition assessment, one should be careful when the dataset is applied to problems that are not modeled explicitly in this research. The networks trained using the Tokaido dataset alone are likely to be confused by component, damage, and depth patterns that do not exist in the dataset, e.g. components of bridges not designed by the standard design, deformed rebar, and debris. Depending on the individual application scenarios, the dataset

should be combined with additional data that is highly relevant to the problem at hand, or integrated into more advanced frameworks for correcting dataset bias, such as transfer learning and sym2real approaches [186].

CHAPTER 4: AUTOMATED BRIDGE COMPONENT RECOGNITION USING REAL-WORLD IMAGE DATA

This chapter investigates approaches to the vision-based automated bridge component recognition using images of complex scenes. Bridge component recognition in this chapter is based on semantic segmentation algorithms, which compute pixel-wise labels instead of image-wise labels (reviewed in section 2.2). Compared to general semantic segmentation problems or object detection problems (e.g. The PASCAL Visual Object Classes Homepage), the bridge component recognition problem has the following challenges:

- (i) For the bridge component recognition to benefit automated vision-based structural inspections, the algorithm should be able to recognize bridge components even when the components are not main objects in the input images (e.g. Both near and far bridge columns need to be recognized).
- (ii) Bridge components are defined by the high-level scene structure as well as the shapes and textures of the components themselves. For example, bridge columns and building columns are distinguished only by the structures to which the components belong.
- (iii) One of the approaches to solve the challenges (i) and (ii) is to increase the size of the model and annotated dataset. However, the bridge component recognition problem is a specialized and relatively novel application of visual recognition algorithms, and limited amount of data is available for training large and deep models.

To perform the bridge component recognition under these challenges, this chapter investigates approaches to impose high-level scene consistency to the bridge component recognition results. In the proposed approaches, the high-level scene structure is first recognized by performing semantic segmentation with 10-class general scene classes (building, greenery,

person, pavement, signs and poles, vehicles, bridges, water, sky, and others), i.e. scene classification. Then, the scene classification results are combined with another stage of semantic segmentation to get bridge component recognition results with high-level scene consistency, i.e. bridge component classification. Three configurations of classifiers are investigated in this study to combine scene and bridge component classifications: (i) naïve configuration, (ii) configuring a scene classifier and a component classifier in parallel (parallel configuration), and (iii) augment the input image to the bridge component classifier by the scene classification results (sequential configuration). These configurations are examined for accuracy, false-positives, and processing time. Furthermore, performance of the bridge component recognition methods under various scenario, where the components appear in different scales in images is investigated by examining the accuracy criteria computed for each image.

The next section discusses existing approaches to impose high-level consistency for visual recognition results. Then, the network architectures and configurations investigated in this chapter are discussed in detail. After discussing the performance evaluation criteria, the configured classifiers are trained and tested using both existing datasets and data labeled manually. The evaluation of bridge component recognition results for the three configurations shows that the scene-level consistencies play a key role in building a robust bridge component recognition platform with reduced false-positives, especially when the image is complex.

4.1 High-level consistency for visual recognition

Recognition of bridge components based on the information from high-level scene context can be categorized as a part detection problem in general. Earlier work regarding part detection took *bottom-up* approaches, where candidates of parts are detected first, and groups of the detected

parts are then evaluated for consistencies using models of their spatial relations [187], [188]. More recently, *top-down* approaches have also been investigated, where an area of attention is first determined based on the recognition of high-level contexts, and then the details of the selected area are analyzed to find parts which might not be remarkable in the original input image [189], [190]. CNNs are frequently used with the top-down approaches, because the algorithms learn the mapping to the high-level contexts in a data-driven manner. In contrast, implementation of top-down approaches is not straightforward, because the effectiveness of the approaches strongly depends on the quality and quantity of data for both high-level contexts and part detection. Although top-down approaches have been investigated in-depth for benchmark problems, such as body part segmentation [191] and (image-wise) fine-grained classification problems [192], [193], those problems are significantly different from the bridge component recognition problem.

This study investigates top-down approaches to integrate two different yet practically related classification problems to impose scene-level consistencies: scene classification and bridge component classification. The scene classification problem has 10 classes (building, greenery, person, pavement, signs and poles, vehicles, bridges, water, sky, and others) to recognize the high-level scene structure captured in the input image. The bridge component classification problem has 5 classes (non-bridge, columns, beams & slabs, other structural, and other nonstructural). As additional supportive information for the bridge component classification, the results of scene classification serve as a form of high-level scene understanding. More specifically, the scene classifier takes the role of explicitly recognizing non-bridge objects and removing them from the focus of the bridge component classifier, thus constraining the role of the bridge component classifier to the classification of component types given the bridge pixels. In terms of data, scene

classification is a well-investigated problem with many existing datasets, while bridge component recognition is a specific problem, where new data needs to be collected.

4.2 Network architectures

This chapter applies two semantic segmentation algorithms, FCN and SegNet, whose architectures are determined by extending the multi-scale CNN architecture used in the preliminary work [194], [195]. The preliminary work tested multi-scale CNN architectures ranging from 5 to 45 layers, and selected the architectures with 21 convolutional layers.

In this chapter, a network architecture with 45 convolutional layers shown in Table 1 and Figure 2 is developed for the feature extraction using FCNs. The first 21 convolutional layers are the same as the convolutional layers of the multi-scale CNN architecture used previously, which consist of 1, 8, and 12 convolutional layers spanning three different resolutions. This network then applies 12 convolutional layers (Conv21 – Conv32), a max-pooling layer (Maxpool3), and then 12 convolutional layers again (Conv33 – Conv44). The predictions using layer output from Conv20 and Conv32 are “skipped” and merged with the prediction from the last convolutional layer. This network is called *FCN45* in this research.

Table 5 FCN45 architecture

Name	Filt. Size - Stride	ResNet Connect.
Conv0	7x7x64-2	Conv0 – 2
Conv1-8	3x3x64-1	Conv2 – 4, 4 – 6, 6 – 8
Maxpool0	2x2-2	Maxpool0 – Conv10
Conv 9-20	3x3x128-1	Conv10 – 12, 12 – 14, 14 – 16, 16 – 18, 18 – 20
Maxpool1	2x2-2	Maxpool1 – Conv22
Conv21-32	3x3x128-1	Conv22 – 24, 24 – 26, 26 – 28, 28 – 30, 30 – 32
Maxpool2	2x2-2	Maxpool2 – Conv34
Conv33-44	3x3x128-1	Conv34 – 36, 36 – 38, 38 – 40, 40 – 42, 42 – 44
Pred. Layer Wt. decay Skips	Conv1x1 0.0001 From Conv20, Conv32	

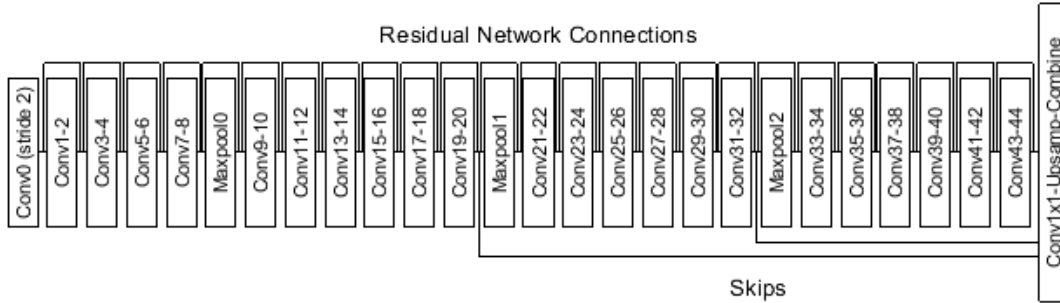


Figure 26 Schematic view of the FCN45 architecture with 45 convolutional layers

The *FCN45* architecture is extended to SegNet architectures by concatenating decoding layers after Conv44. Two architectures are defined, depending on the numbers and sizes of layers after up-sampling: *SegNet45-S* has one convolutional layer after each up-sampling layer with filter size 7x7. *SegNet45* architecture has three convolutional layers after each up-sampling layer with filter size 3x3.

4.3 Network configurations

This section describes three network configurations tested in this study to combine scene classification and bridge component classification effectively.

4.3.1 Naïve configuration

A naïve bridge component classifier, which recognizes bridge components directly from the input images without explicitly imposing scene-level consistencies, is trained and tested for comparison purposes. This classifier only depends on the training for the bridge component recognition task to integrate low-level contexts and high-level scene structures.

4.3.2 Parallel configuration

The parallel configuration refines the bridge component classification results using the pixel-level scene classification results, as shown in Figure 27(a). An example of this configuration is found in a study by [196], which refines structural damage recognition results using output of another binary classifier with “Damage” and “No damage” classes. This study places a linear classification layer after the softmax probability output of the scene classifier and the bridge component classifier to find appropriate weights for the combination without setting thresholds manually. The parallel configuration can integrate the bridge component recognition results at each pixel with the scene-level understanding at the corresponding location.

4.3.3 Sequential configuration

The sequential configuration first computes the softmax probability map corresponding to first nine categories of the scene classification (scaled to $[0,255]$), and then augments the input image (3 channels, i.e. RGB) with the 9-channel softmax probability map to form an input to the bridge component classifier (Figure 27(b)). The softmax probability map for the last scene class is not used explicitly, because the probability is equal to 1 minus the sum of the probabilities for the other nine classes. By feeding the augmented images to the bridge component classifier, the classifier can integrate the bridge component recognition results at each pixel with the scene-level understanding inside the FOV of the bridge component classifier.

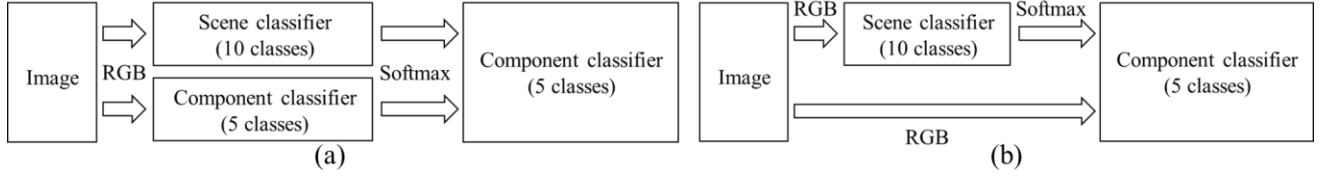


Figure 27 Network configurations to impose high-level scene consistency

4.4 Performance evaluation criteria

This section describes accuracy criteria used to evaluate the bridge component recognition results. The quantities listed below are accuracy criteria commonly used to evaluate classification and semantic segmentation results:

- Number of classes: N_{class}
- Number of pixels with ground truth label i and predicted label j : n_{ij}
- Total number of pixels of class i : $t_i = \sum_{j=1}^{N_{\text{class}}} n_{ij}$
- Total number of (labelled) pixels: $T = \sum_{i=1}^{N_{\text{class}}} t_i$
- The per-class accuracy criteria for class i are defined as:
- True positive (TP_i) = n_{ii} , false positive (FP_i) = $\sum_{k \neq i} n_{ki}$, false negative (FN_i) = $\sum_{k \neq i} n_{ik}$, true negative (TN_i) = $T - TP_i - FP_i - FN_i$
- Precision: $PR_i = TP_i / (TP_i + FP_i)$
- Pixel accuracy (Recall in the binary classification context): $PA_i = n_{ii} / t_i = TP_i / (TP_i + FN_i)$.
- Intersection over union: $IoU_i = TP_i / (TP_i + FN_i + FP_i)$
- False positive rate: $FPR_i = FP_i / (TN_i + FP_i)$
- These per-class criteria are averaged to get the following criteria for the entire problem:
- Mean precision: $MPR = (1/N_{\text{class}}) \cdot \sum_{i=1}^{N_{\text{class}}} PR_i$

- Mean pixel accuracy: $\text{MPA} = (1/N_{\text{class}}) \cdot \sum_{i=1}^{N_{\text{class}}} \text{PA}_i$
- Global pixel accuracy: $\text{GPA} = (1/T) \cdot \sum_{i=1}^{N_{\text{class}}} \text{TP}_i$
- Mean IoU: $\text{MIoU} = (1/N_{\text{class}}) \cdot \sum_{i=1}^{N_{\text{class}}} \text{IoU}_i$
- Frequency-weighted IoU: $\text{FWIoU} = (1/T) \cdot \sum_{i=1}^{N_{\text{class}}} (t_i \cdot \text{IoU}_i)$
- Mean false positive rate: $\text{MFPR} = (1/N_{\text{class}}) \cdot \sum_{i=1}^{N_{\text{class}}} \text{FPR}_i$
- Global false positive rate: $\text{GFPR} = (1/T) \cdot \sum_{i=1}^{N_{\text{class}}} \text{FP}_i$

Precision, recall, and false positive rate are commonly used in binary classification context [129], [197]. For the large-scale semantic segmentation and object detection problems with multiple classes, criteria such as PA, IoU, MPA, GPA, MIoU, and FWIoU are also commonly used to evaluate the ability of the algorithms to detect and localize the objects of interest [18], [48], [198], [199]. As discussed in the challenges (i)-(iii) in the introduction, bridge component recognition methods need to detect and localize components appearing in various scales in images of urban scenes. However, the criteria above are based on the number of pixels with correct and incorrect label predictions, which tend to overemphasize the performance of the methods on large (near) bridge components. To evaluate the bridge component recognition results in different scenarios, following image-wise criteria are also used in this study:

- Precision of image I and class i : Im. PR_i^I
- Pixel accuracy (recall) of image I and class i : Im. PA_i^I
- Intersection over union of image I and class i : Im. IoU_i^I
- False positive rate of image I and class i : Im. FPR_i^I

These image-wise criteria are averaged over testing images to derive mean image-wise PR/PA/IoU/FPR (denoted as Im. PR_i , Im. PA_i , Im. IoU_i , and Im. FPR_i , respectively). The image-

wise criteria interpret each testing image as a sampled scenario where bridge component recognition is performed, and estimate the expected value of PR, PA, IoU, and FPR over all possible scenarios.

4.5 Data collection and labeling

To train the bridge component classifiers integrated with scene-level consistency, training data needs to be collected for both the scene classification problem and the bridge component classification problem. In this study, the datasets are made by combining existing datasets and data manually labeled by the authors' research group. The manual pixel-wise labels are assigned using a MATLAB GUI developed by the authors' research group [196].

4.5.1 Scene classification dataset

The training dataset collected for scene classification is summarized in Table 6. The dataset is divided into three categories, general, urban, and bridge. The general images and labels are collected by combining the Stanford Background Dataset [200] and SIFT Flow dataset [201], both of which are fully-labelled image datasets of broad range of scenes. For the urban category, images are acquired from three sources. SYNTHIA dataset [198] provides fully-labelled images of a virtual city viewed from various perspectives and under various lighting conditions. CamVid dataset [56], [57] is a fully-labelled video image sequence of the city of Cambridge. Beside the existing datasets, images of various cities in the USA and Japan are downloaded from Google Street View using the Google Street View API. Bridge data consists of bridge images included in the general and the urban datasets, images taken by the authors, bridge and flyover images from the ImageNet dataset [204], Google Street View images, and other images provided by Mr. Shintaro Arai, an amateur photographer. The datasets marked in Table 6 are labeled manually by

the authors' research group into 10 categories (building, greenery, person, pavement, signs and poles, vehicles, bridges, water, sky, and others). A MATLAB GUI developed in the authors' research group was used to assign per-pixel labels to the images [156].

Table 6 Datasets used for scene classification
(Sources marked with asterisks indicate the images are labelled manually by the authors)

Category	Source	# of images	Notes
General	Stanford Background Dataset SIFT Flow dataset	3,403	92 bridge photos Night view excluded Fully labelled
Urban	SYNTHIA Dataset CamVid Dataset *Google Street View	6,842	69 bridge photos Night view excluded Fully labelled
Bridge	From General and Urban Datasets	161	Fully labelled
	*ImageNet (Bridge, Flyover) *Photos taken by the authors *Photographer images	1,257	Night view excluded Artistic images excluded Partly labelled
	*Google Street View * Photographer images	234	For testing Partly labelled

Before combining the datasets, all images are resized to have 320 pixels in the longer dimension. Then, the labels of the existing datasets are transferred to those used in this study according to the correspondences shown in Table 7. Finally, night view images and artistic images are excluded, because those images are out of scope of this study.

Table 7 Label correspondences among datasets used for scene classification
(Labels in existing datasets not listed here are neglected)

This study	Labels in existing datasets
Building	Building, Awning, Balcony, Building, Door, Staircase, Window, Wall, Archiway
Greenery	Tree, Grass, Mountain, Field, Plant, Vegetation, Terrain, VegetationMisc
Person	Person, Pedestrian, Rider, Bicyclist, Child
Pavement	Road, Crosswalk, Sidewalk, Parking-slot, Lanemarking, LaneMkgsDriv, LaneMkgsNonDriv, ParkingBlock, RoadShoulder
Sign&Poles	Pole, Sign, Streetlight, Traffic sign, Traffic light, Column_Pole, SignSymbol, TrafficLight
Vehicles	Bus, Car, Bicycle, Motorcycle, Truck, Bus, Train, MotorcycleScooter, OtherMoving, SUVPickupTruck, Truck_Bus
Bridges	Bridge
Water	Water, River, Sea
Sky	Sky, Moon, Sun
Others	Bird, Boat, Cow, Fence, Animal, CarLuggagePram, TrafficCone

The general and urban images are shuffled within each category, and saved as data blocks containing up to 250 images (Figure 28). For the general and the urban images, the first 90% of the images in each data block are used during training (3062 general images and 6,157 urban images), and the remaining 10% are used during testing (341 general images and 685 urban images).

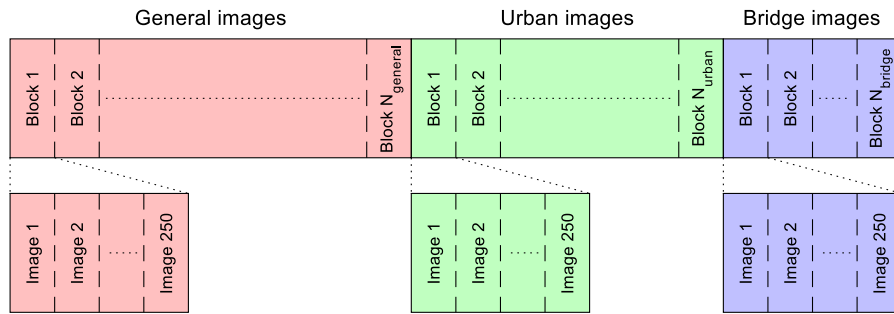


Figure 28 Data blocks of the scene classification dataset

For the bridge images, 234 testing images are separated from training images as shown in Table 6 to ensure independence between training and testing data (Google street view images from different cities, photographer images from different bridges). Then, the training images are saved

as data blocks containing up to 250 images (Figure 28). The bridge testing images are merged into a single file.

4.5.2 Bridge component classification dataset

The training data for the bridge component classification consists of “Bridge” and “Flyover” images from ImageNet, Google Street View images, and the images taken by the authors and the amateur photographer, as shown in Table 8. ImageNet images and photographer images tend to show bridges as main objects and contains many large bridges, including suspension, cable-stayed, and arch bridges. On the other hand, the Google Street View images are collected by downloading images in urban areas randomly and then manually selecting images including bridges. Therefore, smaller girder bridges are mainly included and sometimes bridges are far and/or partially photographed.

Table 8 Bridge component classification dataset
(Sources marked with asterisks indicate the images are hand-labelled by the authors)

Source	# of images	Notes
*ImageNet (Bridge, Flyover) *Photos taken by the authors *Others	1135	Bridge photos Night view excluded Artistic images excluded
Google Street View	194	Non-bridge photos
*Google Street View *Others	234	For testing

All the images are labeled manually by the authors’ research group into one of the five classes; Non-bridge, Columns (including piers), Beams & Slabs, Other Structural (trusses, arches, cables, abutments, remarkable braces, remarkable bearings etc.), and Other Nonstructural (Fences, poles etc.). The same 234 testing images for the scene classification (Google Street View images from different cities and photographer images from different bridges) are used again for the testing

of the bridge component classification. Each of the training and testing datasets are resized to have 320 pixels in the longer dimension saved as a single data block.

4.6 Bridge Component Recognition Results

This section describes training and testing of the bridge component classifiers integrated with high-level scene consistency using the collected datasets. The training begins with loading image data blocks (see Figure 28). Then, the training images in each data block are shuffled, augmented, and used to update the network parameters. Following [48], the image augmentation includes random resizing (between 75% and 125% for the general and the urban images, between 50% and 150% for the bridge images), random rotation up to $\pm 15^\circ$, random flipping, jitter (zero-mean normal distribution with $\sigma=2$), and random cropping. The network weights are initialized by zero-mean normal distribution with $\sigma = 0.001$ and then updated using Adam optimizer [205]. All the operations are implemented using Python and TensorFlow [206].

4.6.1 Scene classification results

During training of the *FCN45* for scene classification, an image data block is loaded randomly from each of the general, urban, and bridge categories. After shuffling and augmenting the training images in the loaded data blocks, a mini-batch of 10 images with the cropped dimension 180x180 is made by choosing 4 images from the general category, 4 images from the urban category, and 2 images from the bridge category. The creation of a mini-batch and subsequent network parameter updates are repeated until all the training images in at least one of the loaded data blocks are seen by the network. This study considers the iterations until the network sees all data blocks one cycle of training.

The *FCN45* network is trained for 200 cycles in total, where the learning rate is gradually decreased from 1.0×10^{-4} (first 100 cycles) to 1.0×10^{-5} (the next 50 cycles), and 1.0×10^{-6} (last 50 cycles). For the *FCN45* architecture, the up-sampling filters are fixed to the bilinear filters for the first 100 cycles, and updated during the later cycles. No dropout is applied during training.

After the training of the *FCN45* network, the *SegNet45-S* and *SegNet45* networks are trained by initializing the filters of Conv0 – Conv44 by those of the trained *FCN45* network. Using a mini-batch of 5 images (2 from general, 2 from urban, and 1 from bridge category) with the cropped dimension 256x256, the SegNet decoders are trained for 30 cycles with learning rate 1.0×10^{-4} first. Then, the entire networks (encoders and decoders) are trained for 10 cycles with learning rate 1.0×10^{-5} , and 5 cycles with learning rate 1.0×10^{-6} . The dropout (keeping 80% of the layer outputs) is applied to the decoding layers during the last 15 cycles to avoid overfitting.

The MPA, GPA, MIoU, and FWIoU discussed in section 4.4 are shown in Table 9. The *FCN45* and the *SegNet45* architectures achieve comparative accuracy, and both architectures performed better than the *SegNet45-S* architecture. Because of the simplicity of the problem (the number of classes were reduced to 10, see Table 7), state-of-the-art accuracy was obtained for major scene classes (e.g. building) without intensive training for recognizing less frequent classes accurately. The confusion matrix of the trained *FCN45* network evaluated for the testing set is shown in Figure 29. The network can classify every class except for the “Others” class with the per-class pixel accuracy (PA) higher than 80%, including 95% for the bridge class.

Table 9 Performance evaluation criteria of the scene classification

	MPA	GPA	MIoU	FWIoU
<i>FCN45</i>	88.77	91.12	70.00	85.72
<i>SegNet45-S</i>	86.07	85.57	64.15	79.54
<i>SegNet45</i>	89.02	91.08	69.99	85.82

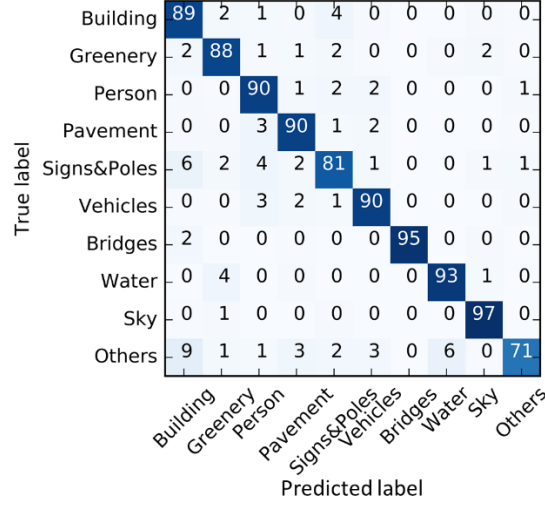


Figure 29 Confusion matrix of the FCN45 scene classifier. Each diagonal element shows PA

The results are obtained by a Windows PC with an i7 6700 3.40GHz processor, NVIDIA GTX 1070 8Gb graphics card and 16GB RAM. For a single 320×320 image, the processing time of the *FCN45*, *SegNet45*, and *SegNet45-S* architectures are 0.0472s, 0.0562s, and 0.0519s, respectively.

4.6.2 Bridge component recognition results with high-level scene consistencies

In this section, bridge component classifiers with scene-level consistencies are trained using three configurations (naïve configuration, parallel configuration, and sequential configuration), and the effectiveness of each configuration is discussed.

Naïve configuration: First, the naïve bridge component classifier without scene classification is trained using the *FCN45*, *SegNet45*, and *SegNet45-S* architectures. After experimenting with different training parameters, the training scheme for the *FCN45* architecture was determined as summarized in Table 10. The training of the *FCN45* is performed for 2,250 cycles (about 0.13k iterations per cycle) in total. The cropped image size is gradually increased to perform faster iterations in the earlier cycles and to incorporate context from larger field of view

in the later cycles. Learning rate is gradually decreased from 1.0×10^{-4} to 1.0×10^{-5} and 1.0×10^{-6} . The dropout is applied at the prediction layers of the FCN from the 1,001st cycle to reduce overfitting effect.

Similar to the training of the scene classifiers, the *SegNet45* and *SegNet45-S* architectures are trained by initializing the encoding layers by those of the trained *FCN45* network. Then, the decoding layers are trained for 100 cycles with learning rate 1.0×10^{-4} , followed by the training of the entire network for 50 and 10 cycles with learning rates 1.0×10^{-5} and 1.0×10^{-6} , respectively. The dropout (keeping 80% of the layer outputs) is applied to the decoding layers during the last 60 cycles to avoid overfitting.

The performance evaluation criteria discussed in section 4.4 are shown in Table 11 for the *FNC45*, *SegNet45*, and *SegNet45-S* architectures. Confusion matrix of the *FCN45* network evaluated for the testing set is shown in Figure 30(a). Bridge component recognition results of example input images are shown in Figure 31 for the *FCN45* network. Figure 31(a) shows the input image, Figure 31(b) shows the corresponding ground truth labels, and Figure 31(c) shows the labels predicted by the naïve configuration with the corresponding MIoU and FWIoU values. In the figure, the first seven images are from the testing set of the bridge component classification dataset, and the last two images are from the non-bridge testing images from the scene classification dataset.

Parallel configuration: The scene and the naïve bridge component classifiers with *FCN45*, *SegNet45*, and *SegNet45-S* architectures are configured in parallel to refine the bridge component recognition results. As mentioned previously, the refined classification scores are computed by putting a linear output layer after the combined softmax probability output (10-class

scene classification and 5-class bridge component classification). The linear output layer is trained for 100 cycles with the learning rate 1.0×10^{-3} and the batch size one.

The performance evaluation criteria presented in Table 11 indicate that the performances of the naïve and the parallel configurations do not differ significantly (e.g. difference of MIoU and FWIoU are less than 1%). The confusion matrix evaluated for the testing set in Figure 30(b) for the *FCN45* network. Because of the high correlation of the naïve classifier output to the desired bridge component recognition results, the information from the scene classifier trained on a different dataset tends to be disregarded. The example results of the *FCN45* are shown in Figure 31(d), which again shows similarity to the results of the naïve classifier shown in Figure 31(c).

Sequential configuration: In this configuration, the scene classifiers with the *FCN45*, *SegNet45*, and *SegNet45-S* architectures are used to augment the input image, and additional classifiers with the same architectures are trained to recognize the bridge components from the augmented image. The bridge component classifier with the *FCN45* architecture is trained for 2,250 cycles in total, with the training scheme summarized in Table 10. Then, the *SegNet45* and *SegNet45-S* networks are trained by the steps described in the previous section.

The performance evaluation criteria are presented in Table 11 and the confusion matrix of the *FCN45* network evaluated for the testing set is shown in Figure 30(c). The sequential configuration achieves performance comparable to that of naïve/parallel configurations (e.g. differences of MIoU and FWIoU are less than 1%). Some of the example *FCN45* results are shown in Figure 31(e). The visual comparison of the Figure 31(c)-(e) show that the bridge component recognition results are similar when the bridge is photographed as a main object (first, second, fourth, and fifth images). The bridge component recognition results of the third and the seventh images are less accurate, which implies that the training data for the corresponding viewpoints

and/or bridge shapes are not sufficient. The results of the sixth image are better with the sequential configuration in terms of consistency, because the predicted beam/slab exists along the bottom of the arch, not protruding into the water area.

The most remarkable difference between the three configurations can be observed for non-bridge images (the last two images). Because these images do not include any bridge, a perfect classifier should label every pixel into the non-bridge class. However, the naïve classifier shows significant false-positives. As discussed previously, the effect of the high-level scene consistency is not significant for the parallel configuration. In contrast, the sequential configuration shows no false-positive detection. The visual observation of Figure 31 indicate that the information of the scene classifier can be combined more effectively with the sequential configuration because the configuration optimizes the network parameters based on both the input images and the scene classification results, rather than training the bridge component classifier independently.

Table 10 Training schemes for the naïve and sequential configuration

Cycles	Crop size	Upsamp.Filter	Learn. rate	Dropout (Pred)*
1-500	180x180	Bilinear	10^{-4}	None
501-1000	180x180	Learned	10^{-4}	None
1001-1500	184x184	Learned	10^{-4}	0.8
1501-2000	184x184	Learned	10^{-5}	0.8
2001-2150	256x256	Learned	10^{-5}	0.8
2151-2250	256x256	Learned	10^{-6}	0.8
Batch size	10			

*Dropout applied to the prediction layers. The values in the table shows the probability of keeping layer outputs.

Table 11 Performance evaluation of bridge component recognition [%]

	Precision (PR)					MPR	GPR	Per-Class Pixel Accuracy (PA)					MPA	GPA
	NB	Col	BS	OS	ONS			NB	Col	BS	OS	ONS		
Narazaki et al.(2017)	92.5	48.7	72.1	70.0	39.8	64.6	80.3	87.6	70.3	82.3	59.2	32.8	66.4	80.3
Naive(FCN45)	91.4	63.0	76.4	79.5	53.4	72.7	84.1	91.8	63.6	86.7	63.9	42.6	69.7	84.1
Parallel(FCN45)	91.7	62.5	75.7	79.2	54.1	72.6	84.1	91.6	64.2	86.9	64.0	42.1	69.7	84.1
Sequential(FCN45)	91.6	62.1	75.1	82.4	51.6	72.5	83.9	92.0	65.6	86.1	60.5	42.9	69.4	83.9
Naive(SegNet45)	91.4	59.0	74.6	73.9	46.7	69.1	82.3	89.5	62.5	85.1	62.7	43.8	68.7	82.3
Parallel(SegNet45)	91.5	65.0	73.6	74.4	48.8	70.7	82.9	90.2	60.1	87.0	63.4	41.6	68.5	82.9
Sequential(SegNet45)	92.5	57.3	74.3	76.4	47.1	69.5	82.9	89.7	68.8	85.4	64.7	38.9	69.5	82.9
Naive(SegNet45-S)	91.8	61.0	74.4	76.8	50.6	70.9	83.1	90.1	65.1	86.9	63.2	42.1	69.5	83.1
Parallel(SegNet45-S)	91.8	61.9	73.7	77.7	52.4	71.5	83.3	90.4	64.7	87.4	62.8	40.3	69.1	83.3
Sequential(SegNet45-S)	92.5	59.0	73.1	77.3	46.9	69.7	82.7	89.1	66.7	86.6	64.3	42.4	69.8	82.7
	IoU					MIoU	FWIoU	False Positive Rate (FPR)					MFPR	GFPR
	NB	Col	BS	OS	ONS			NB	Col	BS	OS	ONS		
Narazaki et al.(2017)	81.9	40.3	62.4	47.2	21.9	50.8	69.5	10.8	5.0	7.4	2.8	2.2	5.7	4.9
Naive(FCN45)	84.5	46.3	68.4	54.8	31.1	57.0	73.8	13.1	2.5	6.2	1.8	1.7	5.1	4.0
Parallel(FCN45)	84.6	46.4	67.9	54.8	31.0	56.9	73.7	12.6	2.6	6.5	1.9	1.6	5.0	4.0
Sequential(FCN45)	84.8	46.9	67.0	53.6	30.6	56.6	73.5	12.9	2.7	6.7	1.4	1.8	5.1	4.0
Naive(SegNet45)	82.6	43.6	66.0	51.3	29.2	54.5	71.5	12.9	3.0	6.7	2.5	2.3	5.5	4.4
Parallel(SegNet45)	83.2	45.4	66.3	52.0	28.9	55.2	72.2	12.8	2.2	7.3	2.4	2.0	5.3	4.3
Sequential(SegNet45)	83.6	45.5	65.9	53.9	27.1	55.2	72.4	11.1	3.5	6.9	2.2	2.0	5.1	4.3
Naive(SegNet45-S)	83.3	46.0	66.8	53.1	29.8	55.8	72.5	12.3	2.8	7.0	2.1	1.9	5.2	4.2
Parallel(SegNet45-S)	83.7	46.3	66.6	53.2	29.5	55.9	72.7	12.4	2.7	7.3	2.0	1.7	5.2	4.2
Sequential(SegNet45-S)	83.1	45.6	65.6	54.1	28.7	55.4	72.1	11.0	3.2	7.4	2.1	2.2	5.2	4.3

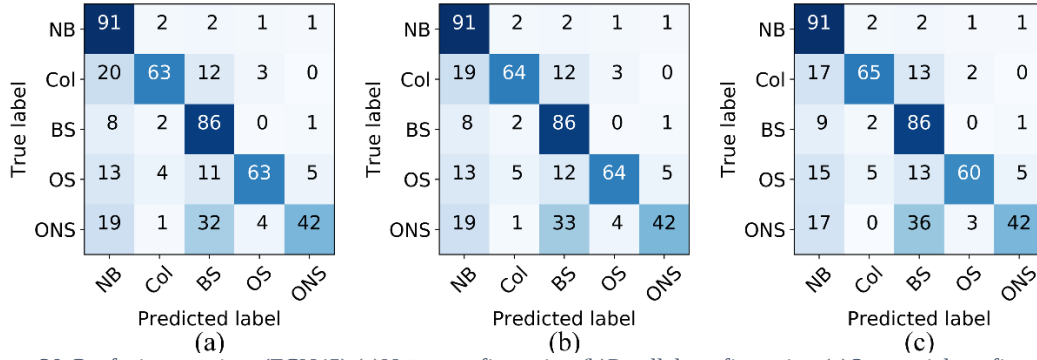


Figure 30 Confusion matrices (FCN45) (a)Naive configuration (b)Parallel configuration (c)Sequential configuration (NB: Nonbridge, Col: Columns, BS: Beams&Slabs, OS: Others(Structural), ONS: Others(Nonstructural))

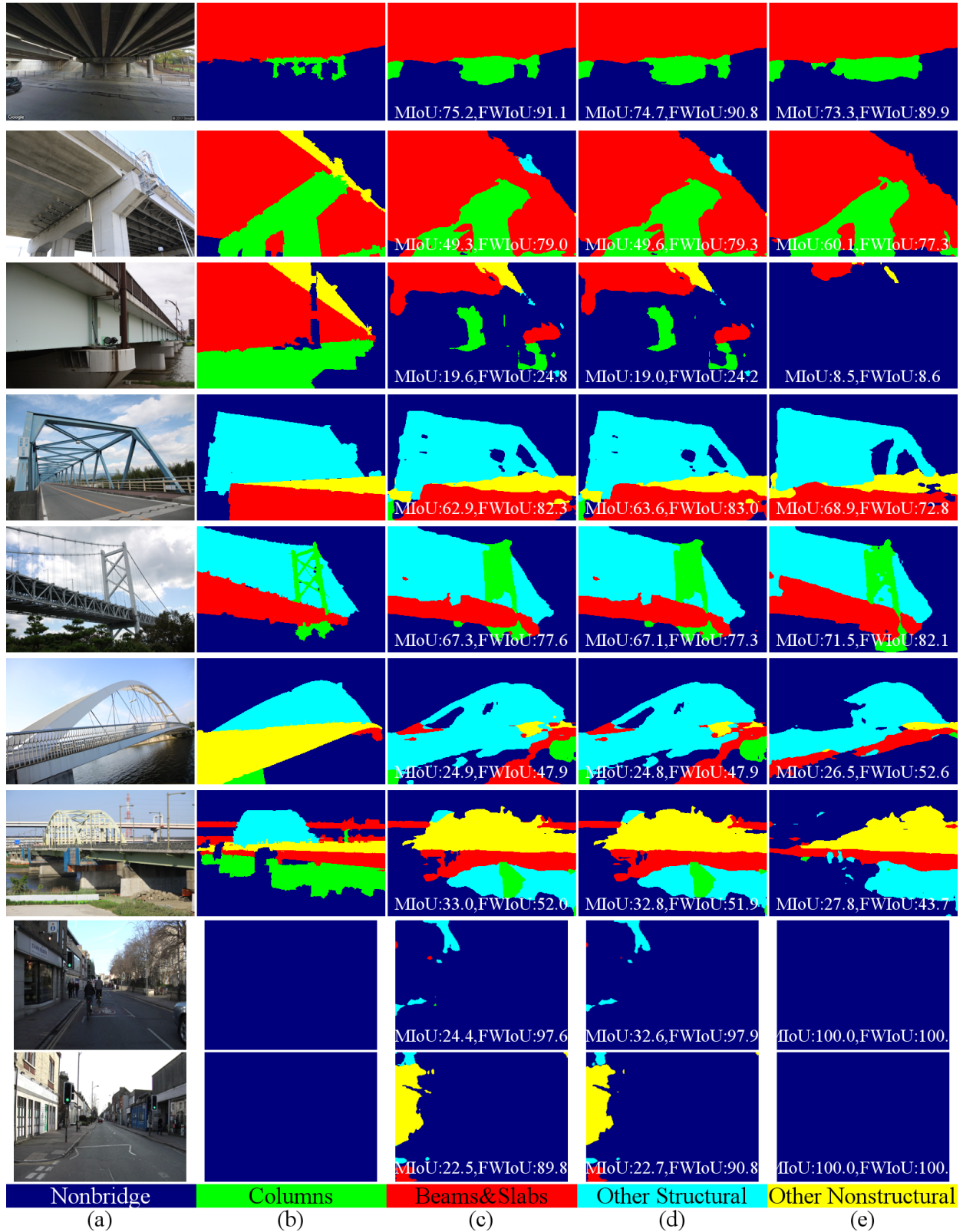


Figure 31 Example results (a)input (b)ground truth (c) naïve configuration (d)parallel configuration (e)sequential configuration (Shown with Mean IoU and Frequency-Weighted IoU)

4.7 Discussions

The performance evaluation criteria in Table 11, the confusion matrices in Figure 30, and the example results in Figure 31 show that the performance of the three configurations are similar when the bridge components are photographed as main objects (The differences of MIoU and FWIoU are less than 1%). In Table 11, the performance evaluation criteria of all configurations in this study are higher than those of the previous study [194] because of the significantly larger model and extensive training. By comparing the results in Figure 31 with example results from [194] in Figure 32, following points can be observed: (i) the FCN results in Figure 31 show smoother boundary owing to the learned upsampling layer, and (ii) the FCN results are closer to human's perception in the sense that closer and more distinct structural components are recognized instance-wise, while far and less distinct components are recognized as areas where the collection of components exist. On the other hand, the filters of multiscale CNNs used in [194] are constrained to be the same among scales, showing less localization capability for close distinct objects. Note that, the level of detail between predicted and ground truth labels (created manually) do not always match (e.g. 4th image in Figure 31). Although removal of this type of error is challenging, the dataset was created to label all bridge components at all scales as consistently as possible by sharing the labeling manual and labeling examples among labellers.



Figure 32 Example results from [194]

The visual observation of the example results in Figure 31 implies that the sequential configuration works better in a scenario where the bridge components may not necessarily exist in the input image. To further investigate the performance of the three configurations in terms of false-positives, non-bridge images of the scene classification testing dataset are fed into the bridge component classifiers. Again, the input images do not contain bridges, and thus every pixel should be classified into non-bridge class by the perfect bridge component classifier. The ratio of pixels classified into one of the four bridge component classes are shown in Table 12 for the nine non-bridge scene classes. The table shows that the performance of the bridge component classifier significantly improves by integrating the scene-level understanding using the sequential configuration. Although the parallel configuration is also effective in reducing the false-positive detections, the improvement of performance is less apparent than the sequential configuration. Therefore, the sequential configuration is the best approach to reduce false-positives in complex urban images.

Table 12 False positive rates for the 9 scene classes [%]
 BLDG: Building, GRN: Green, PSN: Person, PVMT: Pavement, S&P: Signs&Poles
 VHCL: Vehicles, WTR: Water, SKY: Sky, OTHR: Others

Scene label	BLDG	GRN	PSN	PVMT	S&P	VHCL	WTR	SKY	OTHR
Narazaki et al. [194]	1.85	0.58	0.09	0.59	0.32	0.22	2.25	0.13	3.34
Naïve	18.62	5.85	3.65	3.11	6.84	4.64	9.92	0.93	13.20
Parallel	16.93	5.17	3.14	2.70	5.79	3.91	8.90	0.79	11.80
Sequential	0.48	0.08	0.03	0.05	0.21	0.18	0.81	0.03	0.50

To investigate the performance of the bridge component recognition under scenarios where bridge components appear in various scales, image-wise criteria discussed in section 4.4 are computed. The image-wise PR, PA, IoU, and FPR are plotted against the ratio of pixels of each class (excluding nonbridge class) within test images in Figure 33. The bar charts in the figure are the histograms of the ratio of pixels, and the image-wise criteria are averaged over each bin.

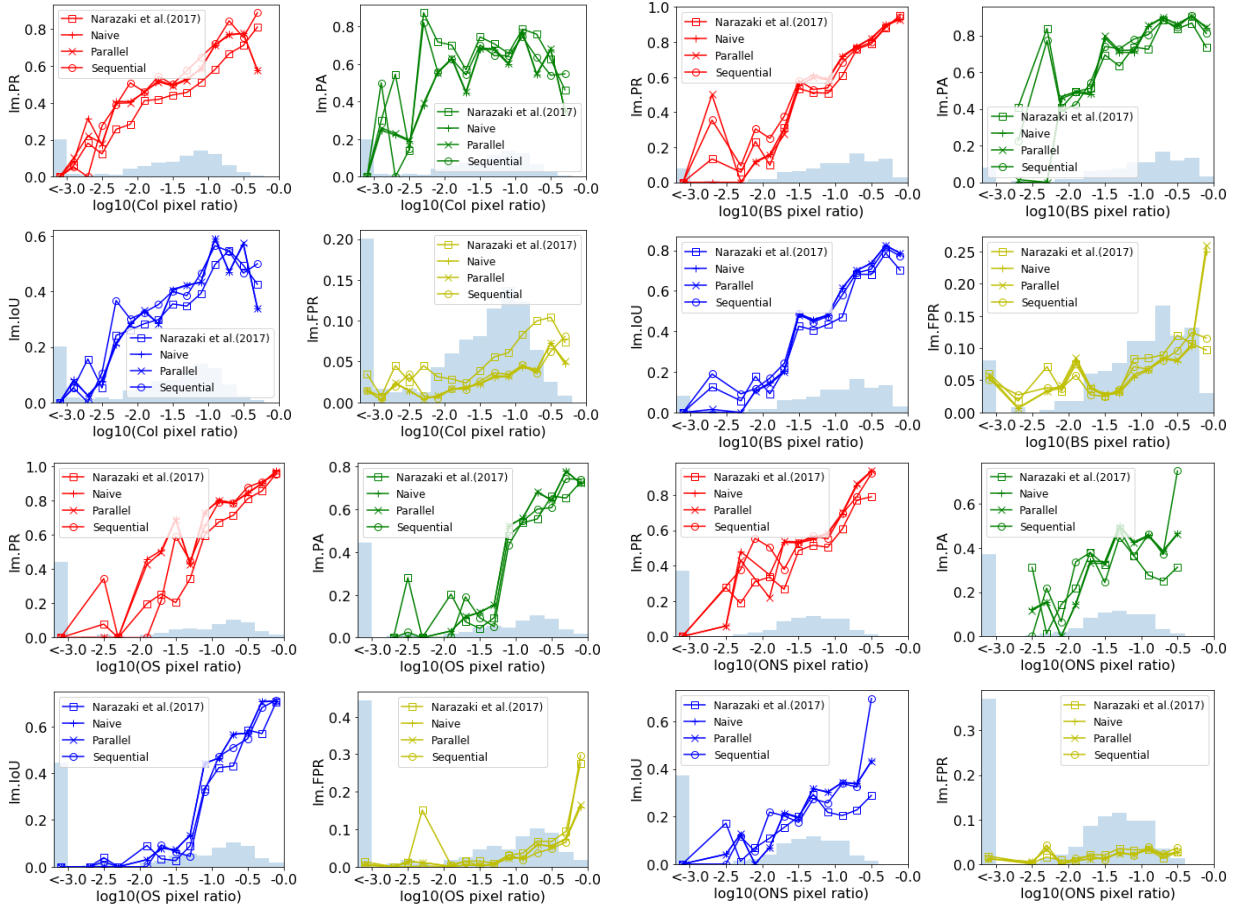


Figure 33 Image-wise accuracy criteria plotted against the ratio of pixels of each class within images (FCN45)

Bar chart indicate the histogram of the ratio of pixels of the corresponding class

(NB: Nonbridge, Col: Columns, BS: Beams&Slabs, OS: Others(Structural), ONS: Others(Nonstructural))

The histograms in Figure 9 show that the test data contains images of bridge components photographed at different scales, ranging from 1% to 100% of the total image area. The figures of image-wise PA for **Col** and **BS** show that the accuracy does not decrease significantly as the ratio of pixels goes below 1% for the column class and 10% for the beams & slabs class. Meanwhile, the PR decreases, and the IoU decreases accordingly. This observation indicates that even small bridge components tend to be detected, although the localization accuracy decreases gradually (causing false-positives). This tendency is less apparent for **OS** and **ONS** component, which could

be explained by more variation of the objects included in those categories and less samples of such objects included in the training data.

Image-wise performance evaluation criteria averaged over all test images are shown in Table 13. The values of the image-wise criteria tend to be lower than the corresponding values in Table 11, because scenarios where bridge components are not main objects are weighted more to compute the values. However, as discussed previously, the decrease of accuracy is less apparent for the PA of the column class (65.6% $\rightarrow$ 62.1% for sequential configuration, 63.6% $\rightarrow$ 59.5% for the naïve configuration, and 64.2% $\rightarrow$ 60.2% for the parallel configuration) and the PA of the beams & slabs class (86.1% $\rightarrow$ 78.9% for sequential configuration, 86.7% $\rightarrow$ 78.4% for the naïve configuration, and 86.9% $\rightarrow$ 79.0% for the parallel configuration). Also, the Im. PR, Im. PA, and Im. IoU for the column class with sequential configuration are 1.1–2.8% higher than the corresponding image-wise accuracy criteria with the naïve/parallel configuration, which indicates that the effectiveness of the sequential configuration to detect and localize columns in various scales. The discussions on the beams and slabs are less straightforward. Im. PR and Im. IoU with the sequential configuration are not necessarily higher than those of the naïve/parallel configurations. Further observation of the image-wise PR for the beams & slabs class in Figure 33 (first row, third column) leads to the interpretation as follows: while naïve, parallel, and sequential configurations behaves similarly for the frequent scenarios (at bins with large frequency), naïve/parallel configurations work slightly better than the sequential configuration because the input images are expected to have strong correlation with the training data. On the other hand, additional input used in the sequential configuration makes the classifier robust to images from different scenarios (1% or less pixel ratio in this case).

Table 13 Image-wise performance evaluation criteria of bridge component recognition [%]

	Im. PR				Im. PA				Im. IoU				Im. FPR			
	Col	BS	OS	ONS	Col	BS	OS	ONS	Col	BS	OS	ONS	Col	BS	OS	ONS
Narazaki et al.(2017)	39.0	59.1	42.2	36.0	67.6	75.5	42.8	33.7	30.3	50.0	24.2	15.4	5.2	7.9	3.5	2.2
Naive	46.9	63.4	55.0	44.7	59.5	78.4	46.2	37.8	33.6	54.5	30.9	20.7	2.6	7.1	2.3	1.7
Parallel	47.0	62.8	55.5	44.6	60.2	79.0	46.3	37.1	33.8	54.3	31.3	20.6	2.7	7.3	2.4	1.6
Sequential	49.7	61.9	60.0	44.1	62.1	78.9	42.3	36.7	34.9	53.4	30.8	20.4	2.8	7.2	2.1	1.8

Times spent in processing a 320×320 image with the naïve configuration are 0.0419s for the *FCN45*, 0.0511s for the *SegNet45*, and 0.0468s for the *SegNet45-S*. Because the parallel and the sequential configurations consist of two classifiers, the computation time increases accordingly (Parallel: 0.0956s (*FCN45*), 0.1247s (*SegNet45*), 0.1055s (*SegNet45-S*); Sequential: 0.0865s (*FCN45*), 0.1054s (*SegNet45*), 0.0962s (*SegNet45-S*)). However, the computation time for the parallel configuration could be reduced by sharing part of the convolutional layers for the two different tasks. This multi-task approach is often used when the images need to be processed under strong time requirement [64], [68].

Based on the discussion, the sequential configuration of the classifiers is effective in automatically recognizing bridge components in images of complex scenes under challenges (i)-(iii) mentioned in the introduction. The configuration shows equal level of accuracy to the naïve and parallel configurations, and at the same time significantly reduces false-positive detections in the scenario where bridges may not necessarily exist in the input image. The sequential architecture is also effective for detecting bridge components (mainly columns, beams, and slabs) robustly under scenarios where the components are photographed at wide range of scales.

This study also demonstrated appropriate configurations for different scenarios. When large amount of straining data is available (challenge (iii) is relaxed) or processing time is the priority, the naïve or parallel architecture is the most appropriate. Also, if the bridge component recognition problem assumes the images where bridges are main objects (Challenge (i) is relaxed), parallel architecture will yield the best accuracy, although the sequential architecture can also

achieve equal level of accuracy. The bridge component recognition integrated with the high-level scene consistency investigated in this study will be the critical building block of the automated post-earthquake bridge inspection systems.

4.8 Summary

This study proposed an approach for automated bridge component recognition using semantic segmentation algorithms, where per-pixel labels are predicted instead of per-image labels. To get the bridge component recognition results consistent with the high-level scene structure even when the bridge may not necessarily be existent nor a main object in the image, a scene classifier was trained and integrated with the bridge component classifier. In this study, the scene classifier takes the role of explicitly recognizing non-bridge objects and removing them from the focus of the bridge component classifier. Two semantic segmentation algorithms (FCN and SegNet) and the three network configurations (naïve, parallel, and sequential configurations) were investigated to examine the most effective approach to combine the two different yet practically related problems containing images up to 320×320 .

The FCN with 45 convolutional layers (*FCN45*) and the SegNets extended from the FCN45 architecture (*SegNet45*, *SegNet45-S*) were first compared using the scene classification dataset. The *FCN45* and the *SegNet45* achieved high accuracy (including 70% mean intersection over union and more than 85% frequency-weighted intersection over union) and outperformed the *SegNet45-S*.

Then, the bridge component classifier was trained using three configurations to impose high-level scene consistency, i.e., the naïve configuration, the parallel configuration, and the sequential configuration. The accuracy criteria such as mean precision, mean/global pixel

accuracy, mean/frequency-weighted intersection over union, and mean/global false positive rate showed equal level of performance for the three configurations (differences of these criteria for different configurations are limited within 1%). However, when the input image is from a complex scene not necessarily including bridges, the performance of the bridge component classifier was improved by explicitly imposing the scene-level consistencies, showing fewer false-positive detections. The most significant improvement was observed for the sequential configuration, which showed false positive rates less than 1% for all non-bridge scene categories.

Further investigation of image-wise accuracy criteria showed that the sequential configuration is effective for recognizing bridge columns appearing in versatile scales. For the recognition of beams and slabs, the scene classifier in the sequential configuration had an effect of making the classifier robust at less frequent scales, maintaining equal level of accuracy at more frequent scales. The parallel configuration was less effective in reducing the false-positive detections, but can be used effectively when the input images are assumed to include bridges as main objects and/or when there is a limitation on processing time by sharing part of the convolutional layers. The bridge component recognition investigated in this study is an important building block for automated post-earthquake bridge inspection systems, including autonomous navigation of robots for image collection, damage detection consistent with the context of the structure, and the derivation of global safety based on the observed damage.

CHAPTER 5: AUTOMATED BRIDGE COMPONENT RECOGNITION USING VIDEO DATA

In Chapter 4, promising results have been obtained for the automated recognition of critical bridge components. At the same time, research efforts to date have demonstrated significant potential of vision-based approaches for the recognition of damage on the component surfaces (Chapter 2). In such methods, damage recognition is most likely to be successful when the image is a close-up view of the component surfaces, while the bridge component recognition needs global cues from the entire structure. During bridge inspection by humans, this trade-off is easily resolved by first examining the entire structure, and then moving close to the structural components of interest. However, the implementation of the visual recognition task during the bridge inspection process is not straightforward, because the naïve application of convolutional neural networks (CNNs) and their variants processes each frame of the video independently without leveraging information from previous frames.

Inspired by the significant progress of video processing techniques, this chapter investigates automated bridge component recognition using video data, where the information from the past frames is used to augment the understanding of the current frame. The synthetic dataset generated in the Unity game environment (discussed in section 3.2) is used to train and test machine learning algorithms that combines fully convolutional networks (FCNs) and recurrent neural networks (RNNs). The proposed approach outperforms the single image-based FCN both quantitatively and qualitatively.

5.1 Recurrent Neural Networks (RNNs)

Recurrent neural networks refer to neural networks with feedbacks. As shown in the left side of Figure 34 [207], a recurrent unit takes both the input data x and the output of the unit at previous time step, and apply nonlinear operations to compute the output at the current time step. The recurrent architecture can be “unfolded” to create an equivalent graph which can be regarded as a deep architecture with shared parameters W (Figure 34 right).

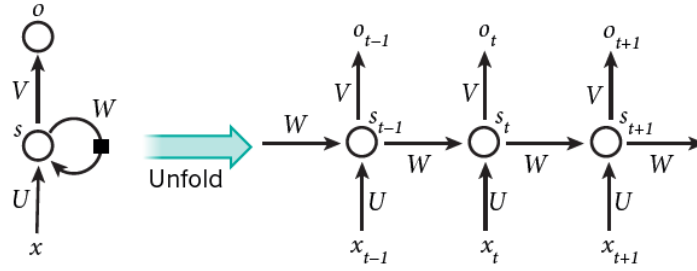


Figure 34. Illustration of Recurrent Neural Networks (RNNs). Quote from [207].

The simple recurrent unit can be created by implementing normal matrix multiplication or convolution, followed by a nonlinear activation function, i.e.

$$o_t = f(U \cdot x + W \cdot s_{t-1}) \quad \text{or} \quad o_t = f(U * x + W * s_{t-1}) \quad (4)$$

where $\cdot$ denotes matrix product and $*$ denotes convolution. The recurrent networks thus created can be trained by gradient descent algorithms (backpropagation through time [76]).

A problem of the simple RNN is the difficulty of learning patterns of long sequences, known as the vanishing gradient problem [78], [207]. As the error gradient propagates backward in time domain (see Figure 34 right), the gradient explodes or vanishes rapidly, which makes the learning of long-term patterns impractical. The vanishing gradient problem is particularly problematic in this study, because the understanding of the structure obtained at a certain global to semi-global view needs to affect the later structural component recognition while the viewer takes a close look at the structural components.

The Long Short-Term Memory (LSTM) is a recurrent unit designed to circumvent the vanishing gradient problem [78]. The structure of the LSTM cell is illustrated in Figure 35. Following Figure 34, the input to the cell at time t is linearly transformed by appropriate weights and passed through a nonlinear activation function g (e.g., the tanh function). Then, the output of g is multiplied by the value of an “input gate”. The input gate is modelled by a sigmoid function of the input x_t and the previous output o_{t-1} . When the sigmoid function takes the value close to one, the input signal flows into the cell and added to the hidden state s . The hidden state s is kept constant in time except for the addition of input signal (this part is called “constant error carousel”). The hidden state is passed through a nonlinear function h (e.g., tanh function) and again multiplied by the value of an “output gate” to control if the hidden state can affect the output of the network. The advantage of explicitly implementing the memory by the constant error carousel is that the gradient does not vanish or explodes during training (mathematical proof is provided in [78]).

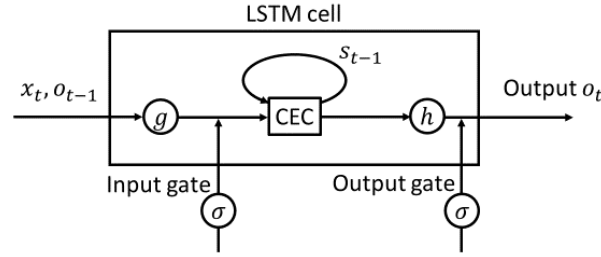


Figure 35. Illustration of the Long Short-TermMemory (LSTM) cell.

Recurrent neural networks including LSTM have been applied to the modelling of video data (sequences of images). Perazzi, et al. [77] shows that stacking current input image and estimated mask at the previous time step and feeding the augmented input into convolutional layers are effective steps to track an object in the video. Shi, et al. [82] developed a convolutional LSTM (ConvLSTM) architecture, where the equations for the units are expressed as follows:

$$s_{t+1} = \sigma(W_{xf} * x_{t+1} + W_{of} * o_t + W_{sf} \circ s_t + b_f) \circ s_t + \sigma(W_{xi} * x_{t+1} + W_{oi} * o_t + W_{si} \circ s_t + b_i) \circ \tanh(W_{xs} * x_{t+1} + W_{os} * o_t + b_s) \quad (5)$$

$$o_t^j = \sigma(W_{xo} * x_{t+1} + W_{oo} * o_t + W_{so} \circ s_t + b_o) \circ \tanh(s_t) \quad (6)$$

where $*$ denotes convolution and $\circ$ denotes element-wise product. The first equation shows the update of the hidden states, where the input to the cell is gated by a sigmoid function. Also, an additional “forget gate” is implemented in this model by multiplying a sigmoid function to the previous state. The second equation shows the output of the ConvLSTM cell expressed by a product of the output from the CEC and the output gate. Siam, et al. [80] used similar recurrent unit (Gated Recurrent Unit [79]) with FCNs to get improved semantic segmentation of video data.

5.2 Bridge component recognition using recurrent neural networks

The network architecture used in this study is illustrated in Figure 36. First, a deep single image-based FCN is applied to extract a map of label predictions (before scaling by softmax). The RNN layers are then concatenated to the lowest resolution prediction layer (prediction from $\mathbf{f}_2$ in the example of Figure 36). Finally, the output from the RNN layers and other skipped layers with higher resolutions are combined to generate the final estimated label maps.

Compared with the fully integrated CNN-RNN architectures [79], [82], the FCN and RNN are trained separately in this study using different datasets. This feature is advantageous, because the dataset for single image-based FCN includes a variety of bridge types, while the video dataset includes simulated video records during random navigation around a single bridge. If the FCN and the RNN are trained simultaneously in an end-to-end fashion using the synthetic video dataset, the resulting network is expected to show overfitting effect. The RNN units are inserted only after the lowest resolution prediction layer, because the RNN units in this study are used to memorize where the video is focused, rather than improving the level of details of the estimated map. In addition to

the reduction in the size of the problem, this architecture is advantageous because predictions of skipped layers can be pre-computed, and the RNN units can be trained without repeating the FCN computation.

Two types of RNN units are tested in this chapter – simple RNN and ConvLSTM units. For the simple RNN units, the input to the unit is augmented by the output of the unit at the previous time step, and the convolution with ReLU activation function is applied in the unit. Alternatively, ConvLSTM units are inserted into the RNN of the architecture and the effectiveness for modelling long-term patterns are evaluated.

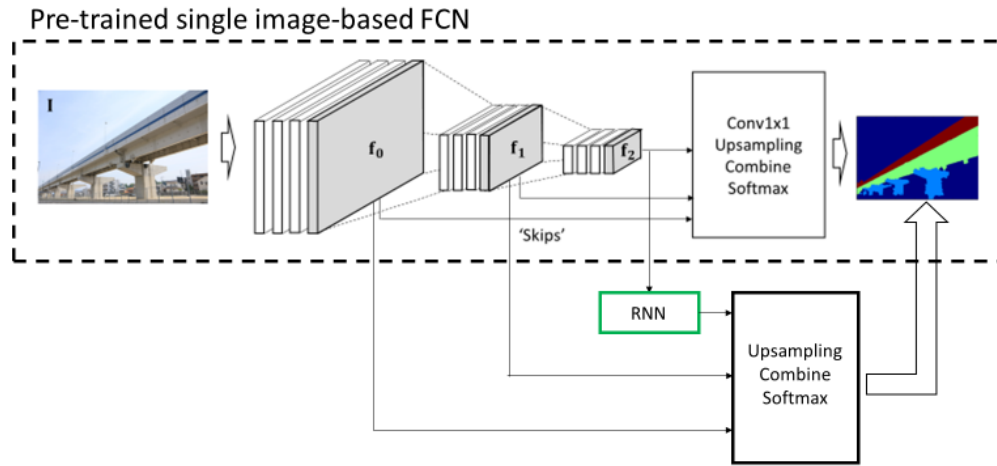


Figure 36 Illustration of the network architecture for bridge component recognition using video data

5.3 Results of bridge component recognition using video data

5.3.1 Dataset

The synthetic video dataset discussed in section 3.2 is used to train and test RNN units. The dataset contains 37,081 training images and 2000 testing images of the resolution 240×320 . The pre-trained single image-based FCN is trained using a dataset discussed in Chapter 4.

5.3.2 Fully convolutional networks

For single image-based FCN, the *FCN45* network with the sequential configuration discussed in Chapter 4 is used herein. This network is trained using real-world images of general bridges, and therefore performs more robustly compared to networks trained using the synthetic video dataset: if the network is trained using the synthetic video dataset, very high accuracy can be obtained for the modeled bridge. However, the trained network is hardly generalizable because of the overfitting effect.

The performance of the FCN is first evaluated on the test set of the video dataset. The confusion matrix is shown in Figure 37. Comparing to the confusion matrix presented in Chapter 4, the results herein show low accuracy for Beams & Slabs class and Other (Nonstructural) classes. Although the accuracy for the column class appears to be improved at a first glance, the comparison is not straightforward, because the bridge component classification dataset contains a variety of bridge types, while the video dataset includes images of a single concrete girder bridge only. The total pixel-wise accuracy for the video dataset is 65.0%, which shows the difficulty in recognizing bridge components from a single frame of the video when the video captures close-up views as well as global views. Examples of estimated labels are shown in Figure 39 (second column).

True label	Nonbridge	89	5	2	1	2
	Columns	14	84	0	0	0
	Beams&Slabs	62	24	10	3	0
	Others(Structural)	0	0	0	0	0
	Others(Nonstructural)	36	52	7	2	0
		Predicted label				
		Nonbridge	Columns	Beams&Slabs	Others(Structural)	Others(Nonstructural)

Figure 37 FCN test results for the video dataset

5.3.3 Recurrent Neural Networks

As discussed in section 5.1, recurrence is introduced to the network by adding two types of recurrent units to the pre-trained single image-based FCN – simple RNN and ConvLSTM. For both cases, 3 layers of the recurrent units with the filter size of 5x5 and the depth of 15 are placed after the lowest resolution prediction map (5x5x5x15 – 15x5x5x15 – 15x5x5x15), followed by a normal convolutional layer (15x1x1x5) to compute the updated prediction. Tensorflow implementation of the ConvLSTM [206] is used, and the RNN parameters are tuned by the Adam optimizer [205]. During training, batch size is set to 1 for both cases, and the ConvLSTM units are unrolled up to 5 time steps. The learning rate is set to 1.0×10^{-4} for the first 10 epochs, 1.0×10^{-5} for the next 5 epochs, and 1.0×10^{-6} for the last 1 epoch (an epoch refers to a set of iterations from the beginning to the end of the training data set). To prevent overfitting of the temporal characteristics of the image sequences, frames are randomly sampled as follows: (i) divide the training data into blocks of size 1,000, (ii) randomly sample 1,000 integers between 0

and 999, and (iii) only use frames of the data block indexed by the integers appeared at least once during the second step.

The confusion matrices for the two recurrent architectures are shown in Figure 38. Total pixel-wise accuracy for the simple RNN and the ConvLSTM units are 74.9% and 80.5%, respectively. Compared to the confusion matrix of the FCN, the effectiveness of the recurrent unit is clearly observed. Moreover, the ConvLSTM outperforms the simple RNN, except for the accuracy for the Beams & Slabs class. Example results in Figure 39 shows the effectiveness of the recurrent units when the FCN fails to recognize the bridge components correctly. Based on the results and discussion in this section, the ConvLSTM units combined with the pre-trained FCN is an effective approach to perform automated bridge component recognition, even if the visual cues of the global structures are temporally unavailable.

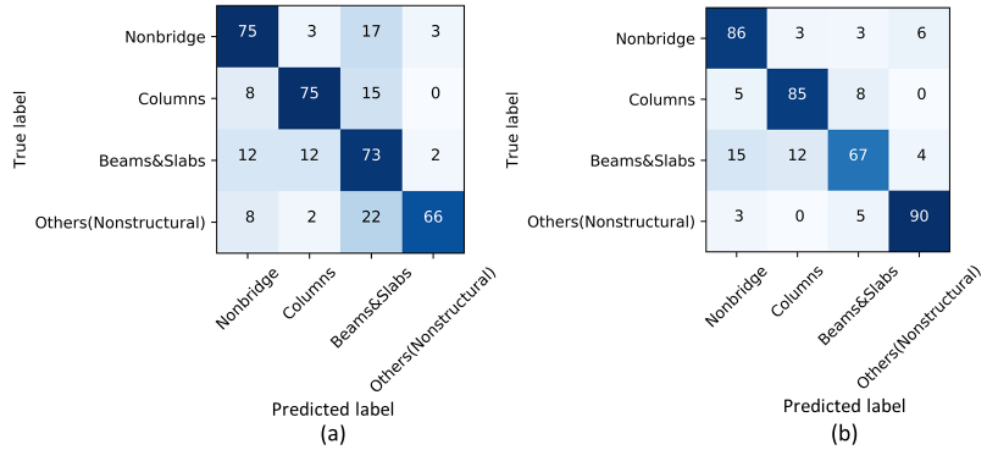


Figure 38. Confusion matrices (a) SimpleRNN, (b) ConvLSTM.

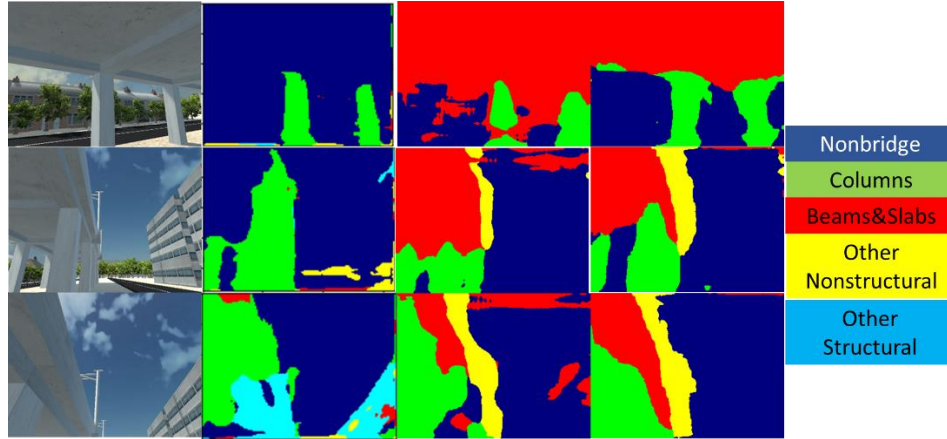


Figure 39. Example results (From left to right: Input image, FCN, FCN-SimpleRNN, FCN-ConvLSTM);

5.4 Summary and conclusions

This chapter investigated the use of recurrent neural networks (RNNs) with a pre-trained fully convolutional network (FCN) to perform the automated bridge component recognition from video data. The bridge component recognition task is not straightforward to solve, because the visual cues of the global structures are lost as the inspector approaches the component. To improve the recognition performance of the FCN using a single image, recurrent units are added to the FCN. By putting the recurrent units only after the lowest resolution prediction layer and training the recurrent unit independently, the RNN parameters were learned in a reasonable amount of time. The architecture with recurrent units outperformed the FCN both quantitatively (pixel-wise accuracy) and qualitatively (example estimated label maps). Moreover, the ConvLSTM units performed significantly better than the simple RNN when the FCN failed to recognize the bridge components. In the future, computation time will need to be thoroughly evaluated to apply the method in near real-time.

CHAPTER 6: BRIDGE COMPONENT RECOGNITION AND LOCALIZATION USING LARGE-SCALE SYNTHETIC DATASET

This chapter improves the accuracy of vision-based bridge component recognition discussed in Chapter 4 significantly, and further extends the capability of the network to the localization of target structural component. The semantic segmentation algorithm, fully convolutional network (FCN), is used to perform structural component recognition. The localization of the target structural components is achieved by estimating the depth map of an input image using an FCN-based algorithm. Tokaido dataset, a large-scale synthetic dataset discussed in section 3.3, is used to augment dataset necessary for training and testing the networks. The trained networks are evaluated using real-world images and synthetic images.

6.1 Bridge component recognition and localization methods

6.1.1 Deep fully convolutional network with depth-wise separable convolution

Following the discussions in Chapter 4, this chapter uses FCNs to perform bridge component recognition and localization tasks, with the extensions described herein. The motivations of the extensions are listed below:

Input image size: The synthetic Tokaido dataset contains images of the resolution $1,920 \times 1,080$, compared to the dataset discussed in Chapter 4 that contains images of the resolution 320×320 at maximum.

Rich geometric patterns with ground truth annotations: The Tokaido dataset contains significantly larger number of images compared to the dataset discussed in Chapter 4. This dataset allows the network to learn accurate recognition of structural component from a variety of viewpoints. Furthermore, the accuracy of manual labeling has been limited when the target object becomes small or slender in images: the network is not expected to learn such patterns accurately.

On the other hand, the synthetic Tokaido dataset contains ground truth labels for all objects in images, which can be learned through training process.

Computational complexity and model complexity: Compared to the case discussed in Chapter 4, the larger input image size causes increased computational burden. The same convolutional layer requires more than 20 times of computational cost for the layer with the dimension of $1,920 \times 1,080$ than the layer with the dimension of 320×320 . Furthermore, deeper network architectures are preferred for the analysis in this chapter to learn complex patterns in high-resolution images, which increases model complexity and makes the training process challenging.

To train semantic segmentation and depth estimation algorithms for this problem, a 58-layer FCN with depth-wise separable convolution has been defined. The depth-wise separable convolution first applies a 2D spatial convolution to each channel separately (depth-wise convolution), and then applies a 3D convolution with kernel size 1×1 that mixes the information contained in different channels (point-wise convolution). If we apply a standard convolution (kernel size: $K_1 \times K_2$, number of filters: L_{out}) to a layer with the size (X, Y) and the number of channels L_{in} , the operation requires the computational cost of $K_1 \times K_2 \times L_{\text{out}} \times X \times Y \times L_{\text{in}}$. On the other hand, the depth-wise separable convolution with the same kernel size and number of filters requires the computational cost of $K_1 \times K_2 \times X \times Y \times L_{\text{in}} + X \times Y \times L_{\text{in}} \times L_{\text{out}}$. At the same time, the number of parameters that need to be optimized decreases from $K_1 \times K_2 \times L_{\text{in}} \times L_{\text{out}}$ to $K_1 \times K_2 \times L_{\text{in}} + L_{\text{in}} \times L_{\text{out}}$. This is a significant decrease when the layer size (number of channels) or the filter size is large. Existing research has shown that the depth-wise separable convolution is effective to reduce computational and model complexities without decreasing performance to undesirable levels [208], [209].

The FCN58 architecture used in this research is illustrated in Figure 40, and the detail of each layer is provided in Table 14. The first two layers apply ordinary convolutions, reducing the feature dimensions from $1,920 \times 1,080$ to 640×360 and 320×180 , respectively. Then, 56 depth-wise separable convolutional layers are applied. Each convolutional layer consists of a convolution operation, followed by batch normalization [51] and ReLu activation function. To train the deep networks effectively, residual network connections are implemented, as specified in Figure 40. The features at multiple scales (layer outputs from Conv0, Conv1, and Maxpool0-2) are then extracted and integrated using bilinear up-sampling and skip connections as described in [18]. With this architecture, the networks output label maps or depth maps of the resolution 640×360 .

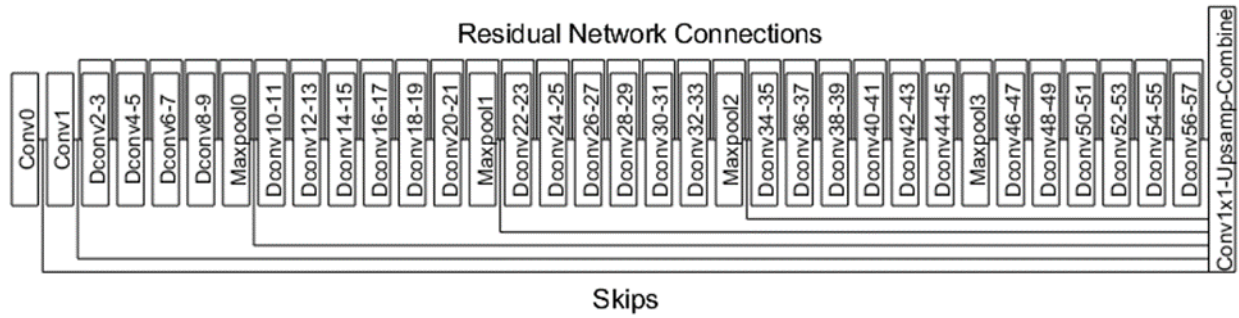


Figure 40 Illustration of FCN58 architecture.
Conv: Ordinary convolutional layer, Dconv: Depthwise separable convolutional layer

Table 14 Detail of the FCN58 architecture (Parameter names follow the Tensorflow convention)

Layer name	Filters	Kernel size	Strides
Conv0	16	7×7	3×3
Conv1	64	7×7	2×2
Dconv2-9	64	3×3	1×1
Maxpool0	-	2×2	2×2
Dconv10-21	128	3×3	1×1
Maxpool1	-	2×2	2×2
Dconv22-33	128	3×3	1×1
Maxpool2	-	2×2	2×2
Dconv34-45	128	3×3	1×1
Maxpool3	-	2×2	2×2
Dconv46-57	128	3×3	1×1

6.1.2 Deep fully convolutional network for depth map estimation

Additional extension to the model is needed to perform the task of depth map estimation.

Compared to the structural component recognition where each pixel is *classified* into an appropriate category, a network for depth map estimation should estimate continuous variables, that is, regression problem. Following the promising results of existing work about depth map prediction, this research uses reverse Huber loss function [64], [74], instead of the cross entropy loss function for semantic segmentation tasks. The reverse Huber loss function (berHu) is defined as follows:

$$B(x) = \begin{cases} |x| & |x| \leq c \\ \frac{x^2 + c^2}{2c} & |x| > c \end{cases} \quad (7)$$

where x is the error between the target value (y) and the predicted value ($\tilde{y}$), and the value of c is determined based on all predictions and the corresponding target values in a mini-batch by the following equation:

$$c := \frac{1}{5} \max_i (\tilde{y}_i - y_i) \quad (8)$$

The network is trained to predict depth scaled linearly to the range [0,1] (note that the dataset contains depth in the range [0.5m, 30m]), which is converted to the depth in meters if necessary. Tensorflow 2 [206] is used to implement the machine learning models discussed in this section.

6.2 Bridge component recognition and localization results

6.2.1 Dataset considerations

The Tokaido dataset is split into training, validation, and testing sets as specified in Table 15. To ensure the uniqueness of the testing images from those used during training process, data from 25 environments out of 200 environments of the Tokaido dataset are used as the testing set. The data from the remaining 175 environments is mixed, shuffled, and further divided into training

set and small validation set. The label maps and the depth maps are resized to 640×360 to accommodate the FCN58 networks.

The Tokaido dataset contains two types of images (defined in section 3.3.5): regular images and close-up images. The FCNs for the structural component recognition and depth map estimation need to be trained for both the global views of the structures and the close-up views of structural components. For that purpose, regular images are used for those tasks, and are split into training set (7,321 images), validation set (300 images), and testing set (1,094 images).

This chapter uses small amount of real-world data in combination with the synthetic Tokaido dataset for the structural component recognition task. The real-world image data is downloaded from the Google Street View by manually selecting viewpoints that result in images of the Tokaido shinkansen viaducts with the standard design. The collected images are labelled manually using Labelbox [210], a simple and user-friendly online tool for the image annotation for semantic segmentation tasks. Structural components that are not modeled in the Tokaido dataset are labelled as either “Other structural” (e.g. braces installed to improve the seismic resistance) or “Other nonstructural” (e.g. plumbing), and are not focused in the subsequent part of this paper. The real-world data for structural component recognition consists of 51 images for training and 50 images for testing, with the associated labels, as shown in Table 15. The numbers of real-world images are significantly smaller than the numbers of images used in the existing literature, and therefore not enough to obtain reliable results.

An appropriate real-world image dataset for depth estimation for bridge component localization is not available currently. Therefore, only synthetic data is used to train and test the depth estimation network. Improving the robustness of the depth estimation to accommodate real-world images is part of the future work.

Table 15 Dataset specifications for the training, validation, and testing of FCNs

Tasks	Training		Validation	Testing	
	Real-world	Synthetic	Synthetic	Real-world	Synthetic
Component recognition	51	7,321	300	50	1,094
Depth estimation	-	7,321	300	-	1,094

6.2.2 Network training schemes

When the networks are trained, training images, both real-world and synthetic, are mixed, shuffled, and flipped (left-right) randomly. Then, mini batches of 8 data samples are created and fed into the network to update the network weights using Adam optimizer [49]. Median frequency balancing technique [211] is used to reduce the effect of data imbalances for the structural component recognition and damage recognition tasks. For all tasks, weight decay of 0.0001 is applied to reduce the overfitting effect.

The number of training iterations are specified in terms of epochs, that is, how many times the networks see the entire training data. Each epoch contains 921 and 915 iterations for structural component recognition and depth map estimation tasks, respectively. For both structural component recognition and depth map estimation, the networks are first trained with the learning rates of 0.001 for 200 epochs, and then for 50 epochs with the learning rate of 0.0001 and 10 epochs with the learning rate of 0.00001. The loss function on the validation set is evaluated after every epoch to monitor the training progress.

6.2.3 Structural component recognition results

The FCN58 network is trained for the structural component recognition task using the data described in section 6.2.1 and the training scheme discussed in 6.2.2. The performance of the trained network on the testing set is evaluated by taking argmax of the softmax probability re-weighted by the inverse of the median frequency balancing weights [212]. Three performance

criteria are evaluated and presented in Table 16: precision ($TP/(TP+FP)$), recall ($TP/(TP+FN)$), and intersection over union (IoU, $TP/(TP+FP+FN)$), where TP, FP, and FN are the numbers of true positive, false positive, and false negative pixels. The performance for the “Others” class for the synthetic data and the performance for the “Rail” and “Sleeper” classes for the real-world data are not provided because the dataset does not contain such pixels. The performance for the “Others” class for the real-world data is provided, but is not focused in the subsequent discussions because those objects are not modeled in the synthetic environment (Section 6.2.1). The low performance for this class indicates that real-world images in the training set alone are not enough to train networks that can produce accurate results.

The performance of the network on the synthetic data is high, including more than 90% of IoU for columns and nonstructural components, as well as nearly 90% of IoU for beams and slabs. The results indicate that the training using 175 synthetic environments can be generalized well to new synthetic environments with different viaduct trajectories, dimensions, and textures. The performance for the rails and sleepers tends to be lower than that of other components, which could be explained by the fact that the dataset contains more images of substructures than those of rails and sleepers (camera is placed in the height range of [1m, 10m]).

The results of real-world images show that the training process can take advantage of the rich synthetic data to improve the network performance significantly. The results obtained in this section outperform the results obtained in Chapter 4, which show less than 50% IoU for columns (accurate localization was challenging because of their slender shapes), and less than 70% IoU for the beams and slabs. Example structural component recognition results are presented in Figure 41. The trained network can capture the global structures, as well as fine detail of the components (e.g. wires and poles in the first and third images). On the other hand, the ground truth labels created

manually follow the rules of the KITTI dataset when the structural components can be seen through objects, such as fences and transparent objects (e.g. the leftmost column in the first image): labels of the closest objects are assigned even when structural components are visible behind them. Those situations, as well as components categorized into “Others” class, can be modeled in the synthetic environment to further improve the accuracy.

Table 16 Network performance on testing set [%] (Structural component recognition)

	Synthetic images			Real-world images		
	Precision	Recall	IoU	Precision	Recall	IoU
No Bridge	98.6	98.4	97.0	88.4	92.2	82.3
Slab	93.2	95.2	89.0	85.7	84.5	74.0
Beam	93.3	92.7	86.8	84.7	83.4	72.5
Column	96.4	96.6	93.2	82.2	88.9	74.6
Nonstructural	95.7	93.9	90.1	90.1	71.9	66.6
Rail	91.9	87.7	81.4	NA	NA	NA
Sleeper	84.6	75.3	66.2	NA	NA	NA
Others	NA	NA	NA	56.8	35.5	28.0
Mean*	93.4	91.4	86.2	86.2	84.2	74.0

* "Others" class is not included

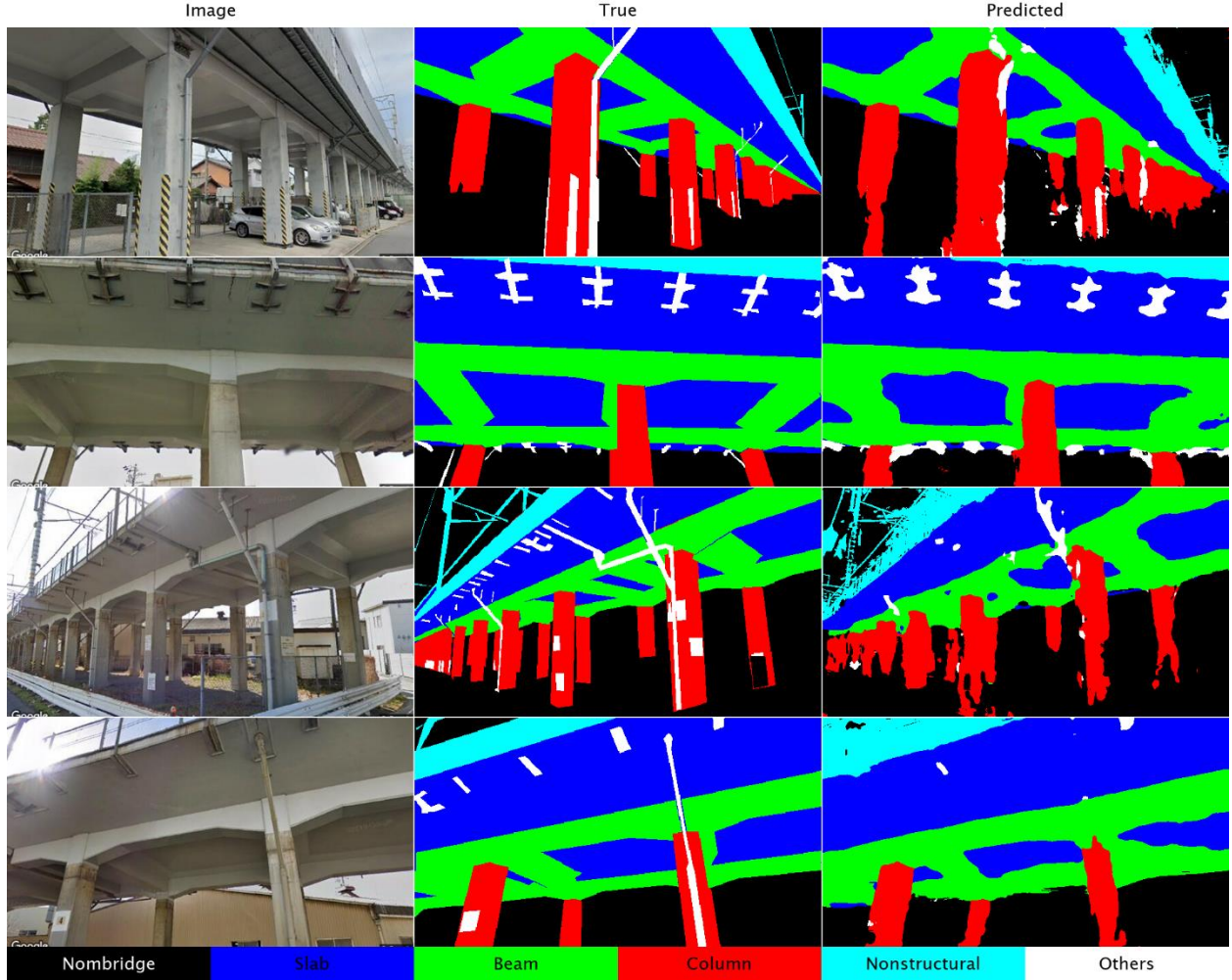


Figure 41 Example structural component recognition results (real-world images)

6.2.4 Depth map estimation results

The FCN58 network for depth map estimation is trained using the reverse Huber loss function and training scheme described in section 6.2.2. This research evaluates the trained network using three criteria used in existing literature about depth map prediction (e.g. [64], [74], [214], [215]). The definitions of those criteria are provided below, where e_i is depth estimation error at each pixel, and d_{true_i} is the true depth at the corresponding pixel.

- Mean absolute error (MAE) $:= \text{mean}_i(|e_i|)$
- Root mean squared error (RMSE) $:= \sqrt{\text{mean}_i(e_i^2)}$,

- Absolute relative distance (ARD) $\coloneqq \text{mean}_i(|e_i/d_{\text{true}_i}|)$

The performance evaluation criteria computed using the testing set is provided in Table 17, showing that the algorithm can localize structural components of interest. Example depth map estimation results are shown in Figure 42 with the corresponding maps of absolute estimation error. The example results indicate that the localization tends to be successful for structural components near the camera, while the accuracy decreases for far or partially occluded components, as well as image regions near object boundaries. These observations are consistent with results reported by the existing literature about general depth estimation problems (e.g. [74]).

Table 17 Performance evaluation criteria of the trained depth map estimation network

MAE [m]	RMSE [m]	ARD [%]
1.32	2.09	10.7

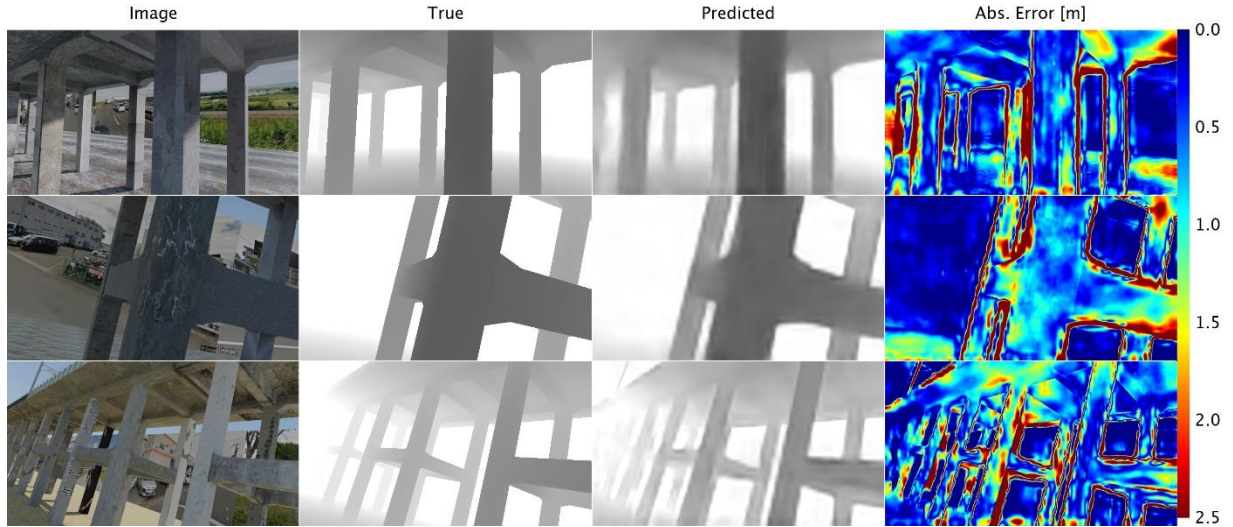


Figure 42 Example monocular depth map estimation results

6.3 Summary and conclusions

This chapter improves the accuracy of vision-based bridge component recognition discussed in Chapter 4 significantly, and further extends the capability of the network to the localization of target structural component. A large-scale synthetic dataset (Tokaido dataset)

developed for the vision-based structural condition assessment of Japanese high-speed railway bridges was used effectively to train deep FCN for semantic segmentation and depth map estimation. For the structural component recognition, the trained network achieved 86.2% of IoU for synthetic testing images, as well as 74.0% of IoU for the real-world testing images. The results for the depth map estimation were also promising, with 1.32m of mean absolute error, 2.09m of root mean squared error, and 10.7% of the absolute relative distance in the range [0.5m, 30m]. The Tokaido dataset produced in this research will facilitate the investigation of various state-of-the-art algorithms proposed for general semantic segmentation and depth recognition problems, so that the structural engineering community can be fully benefitted from the recent advance of the computer vision and data science field.

CHAPTER 7: VISION-BASED UAV NAVIGATION STRATEGIES FOR AUTOMATED IMAGE COLLECTION

This chapter investigates vision-based autonomous UAV navigation strategies for rapid post-earthquake inspection of reinforced concrete railway viaducts. The strategies are designed to mimic the logic followed by human inspectors. To perform the inspection, an inspector obtains initial understanding of the target viaduct by walking nearby. In this step, the inspector does not need to understand the entire structure. Instead, the inspector needs to know the rough idea of the target structure, as well as the clear idea of where the first few critical structural components are located. After the initiation of the task, the inspector's understanding of the target structure is improved incrementally and continuously as s/he experiences more viewpoints. The inspector does not need to have access to the complete design drawing of the structure to perform this task. Instead, s/he needs to have the prior knowledge about the typical geometric or visual contextual patterns of the structure (e.g. regular grid patterns of the column locations, typical dimensions of the columns etc.). The goal of this research is to formulate this strategy in the form that computer-based algorithms can execute it, and demonstrate the strategy using a synthetic environment of railway viaducts.

The next section defines the autonomous UAV navigation problem from technical perspectives, and clarifies the assumptions of this research. Then, the algorithmic components that embodies the autonomous navigation strategy are discussed. The strategy is then demonstrated in a synthetic environment of Japanese high-speed railway viaducts.

7.1 Problem statement

Based on the high-level description of the desired behavior of the robot, this research investigates the following problem:

Given measurement data from a monocular camera, incrementally determine waypoints that a UAV (or other robots that navigate in 3D) can follow to collect close-up images of the critical structural components for rapid post-earthquake inspection.

This system does not replace generic components of robotic platforms, such as odometry, motion control, obstacle avoidance, and local path planning given the waypoints. Rather, the system supplements the existing generic systems, and provides directions that such systems can execute to perform the desired task of this research. With this consideration, this research focuses on the formulation that is specific to the post-earthquake structural inspection problem, and either does not consider or simplifies those generic components.

In the following, the assumptions of this research are listed and explained in detail.

Assumption 1 (Target structure): This chapter investigates rapid post-earthquake inspection of reinforced concrete (RC) railway viaducts for a Japanese high-speed railway line (the Tokaido Shinkansen) operated and maintained by the Central Japan Railway Company, which was detailed in section 3.3.2.

Assumption 2 (Target structural component): This research focuses on the rapid inspection of viaduct columns, because post-earthquake surveys of RC railway viaducts have shown that cracks, spalling, and exposed rebar on the column surfaces are the major modes of structural damage [171], [172], [216].

Assumption 3 (Sensory input): This research uses data from a monocular camera to derive the waypoints. This assumption does not limit the sensory input for the entire navigation system to the use of monocular camera. For example, odometry and local path planning can use other types of sensors (e.g. LiDAR) if such sensor data is available. On the other hand, the system investigated in this research is intended to supplement other generic components of autonomous

navigation systems. The system based on a monocular camera is advantageous, because most robotic platforms (e.g. commercial UAVs) are equipped with a camera that can collect images.

Assumption 4 (GPS): This research does not use the GPS data to determine the waypoints (viewpoints) for the structural inspection directly, because GPS signals are either not available or unstable underneath viaducts. However, this research assumes that the GPS data is available around the target structure, and therefore the initial phase of the inspection can be guided by the GPS data.

Assumption 5 (Sparse point cloud): Besides raw image data, the strategies investigated in this research process the map of the environment represented by sparse point cloud data to determine waypoints. During navigation, the map is created progressively, typically by simultaneous localization and mapping (SLAM) algorithms, structure from motion (SfM) algorithms, or LiDAR-based odometry algorithms. Those mapping algorithms embody an independent active research area with rapid improvement in accuracy, point density, and computational efficiency (e.g. [75], [91], [93], [96], [217]). This research, on the other hand, simplifies the mapping component to focus on the problem formulation: this research computes sparse point cloud data batch-wise using SfM [218], and feeds the 3D points that fall in each frame incrementally to the algorithms that determines the waypoints.

7.2 Methods

7.2.1 Overview

The overview of the proposed autonomous navigation strategy for post-earthquake structural inspection of RC railway viaducts is shown in Figure 43. The system determines waypoints to collect close-up images of critical structural components, or columns in this research, by processing input image stream sequentially. The main component of the system is 3D viaduct

column recognition and localization that represents the columns as rectangular prismatic shapes. Once the columns are detected, waypoints can be set at pre-determined locations relative to the columns. To date, the civil structural component recognition using point cloud data has been investigated using dense point cloud and by off-line processing [219], [220]. On the other hand, the approach presented in this research performs the task (i) using sparse point cloud data (ii) by on-line processing (iii) that is aware of the key characteristics of the target structure. The point (i), the use of sparse point cloud data, is critical for this research, because accurate dense point cloud data is typically not available in the robotic navigation scenarios. The second point, on-line processing, is also appropriate for the application context of this research: the system does not have to wait until it gets the complete information of the target structure. Instead, the system can recognize some of the structural components from partial point cloud data of the structure, and start planning the waypoints quickly. The last point, the awareness to the characteristics of the structure, is meant to mimic the prior knowledge by the human inspector. For example, the inspector can infer unseen columns based on the regular grid patterns of the column locations. This research encodes typical patterns of railway viaducts to improve the column recognition results, as well as guess column information from insufficient or no 3D points associated with those columns.

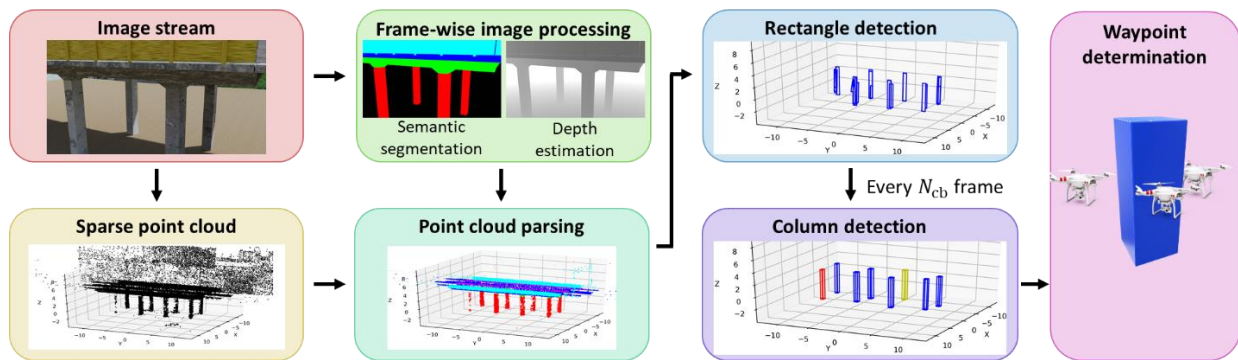


Figure 43 Overview of the proposed autonomous UAV navigation strategy for post-earthquake structural inspection of RC railway viaducts

The system investigated in this research begins with processing each frame of the input image stream. Two types of processing are performed on the raw image data: sparse point cloud generation and visual recognition. The sparse point cloud generation is typically performed by SLAM or SfM algorithms. Visual recognition has two components: semantic segmentation of structural components, and monocular depth estimation. The semantic segmentation results are used to incrementally parse the sparse point cloud data, and the depth estimation results are used during the initial phase of the navigation to estimate the approximate scale of the point cloud data. Once the parsed point cloud data is updated based on the semantic segmentation results, 3D rectangles are searched for from the points labelled as columns. Once in a predetermined number of frames (N_{cb}), the rectangles are further processed to find pairs that form rectangular prismatic shapes representing viaduct columns (blue shapes in Figure 43). If the sufficient number of columns are detected to identify the regular grid pattern, information about columns with unpaired rectangles (yellow) and no rectangle (red) are guessed based on the identified pattern. With this strategy, the UAV can fly around the viaducts using GPS data until it detects first few columns, and then start missions to inspect those components, while improving the column recognition results continuously.

7.2.2 Sparse point cloud generation and frame-wise image processing

The first steps after getting each frame of the image stream is to enrich the sparse point cloud data and perform the semantic segmentation and monocular depth estimation. The sparse point cloud data generation is simplified in this research (Assumption 5 discussed previously). Fully convolutional networks (FCNs) [18] trained in Chapter 6 are used in this research to perform the visual recognition tasks.

7.2.3 Point cloud parsing

In this step, the semantic segmentation results are fused with the labels of the existing points using Bayes' theorem, that is, for the i -th point, the probability that the point belongs to the class c given an image I and a prior probability $P(c)$ is expressed as

$$P_i(c|I) = \frac{P(I|c)P(c)}{\sum_c P(I|c)P(c)} \quad (9)$$

or equivalently,

$$\ln P_i(c|I) = \ln P(I|c) + \ln P(c) + \text{Const.} \quad (10)$$

The prior probability is initialized to 0.125 ($=1/8$) when the point is first obtained, and the posterior probability from the previous update is used during subsequent steps. The likelihood, $P(I|c)$, is obtained from the FCN output. Using the theorem, the log-probability is updated every time new image frame is streamed. A point is defined to have a valid label when the following condition is satisfied:

$$\max_c P_i(c|I) > 99.99\% \quad (11)$$

This step works as the on-line semantic segmentation, as well as the removal of outlier points.

After updating the point labels, the approximate scale of the point cloud is estimated using the monocular depth estimation results. In this research, the point cloud scale is represented by two parameters, mean μ_s and standard variance σ_s^2 . The parameters (μ_s, σ_s^2) are initialized to $(1.0, \infty)$, and updated every time new frame is available, similar to the approach discussed in [217]. When an image I is obtained, its depth map, $D(\mathbf{x})$, and uncertainty (error variance), $U(\mathbf{x})$, are estimated, where $\mathbf{x}$ denotes the image pixel location. At the same time, when the image point $\mathbf{x}$ is the projection of a 3D point in the sparse point cloud data, the depth in the point cloud coordinate system is available, that is, $D_{pc}(\mathbf{x})$. The new observation of the point cloud scale is then modeled by a normal distribution

$$s_{\text{obs}}(\mathbf{x}) \sim \mathcal{N}\left(\frac{D(\mathbf{x})}{D_{\text{pc}}(\mathbf{x})}, \frac{U(\mathbf{x})}{D_{\text{pc}}^2(\mathbf{x})}\right) \quad (12)$$

By defining $\mu_{\text{obs}}(\mathbf{x})$ and $\sigma_{\text{obs}}^2(\mathbf{x})$ as the mean and the standard deviation of this normal distribution and assuming that the scale observations are independent and identically distributed (i.i.d), the scale parameters can be updated by the following equations

$$\mu_s \leftarrow \frac{\sigma_{\text{obs}}^2(\mathbf{x})\mu_s + \sigma_s^2\mu_{\text{obs}}(\mathbf{x})}{\sigma_{\text{obs}}^2(\mathbf{x}) + \sigma_s^2} \quad (13)$$

$$\sigma_s^2 \leftarrow \frac{\sigma_{\text{obs}}^2(\mathbf{x})\sigma_s^2}{\sigma_{\text{obs}}^2(\mathbf{x}) + \sigma_s^2} \quad (14)$$

where the first step can be addressed by the L'Hospital's Rule. To avoid negative impact from the image points with low quality depth estimations or 3D points, this research applies the update equations only to the points where the following condition is satisfied:

$$\mu_s - 3\sigma_s < \mu_{\text{obs}}(\mathbf{x}) < \mu_s + 3\sigma_s \quad (15)$$

The monocular depth estimation and scale parameter update are performed until the global pattern of the structure is identified.

7.2.4 Rectangle detection

This research recognizes and models viaduct columns by rectangular prismatic shapes, and therefore the first step after point cloud parsing is to find 3D rectangles from the points with the “column” label. The rectangle detection of this research is based on the online expectation maximization (EM) method discussed in [221]. The method first parameterizes a rectangle by a 9-element vector, $\boldsymbol{\theta}$, that represents the surface normal (3 degree of freedom, or 3DOF), distance to the plane (1DOF), location on the plane (2DOF), size (2DOF), and rotation in the plane (1DOF), and models the probability density function of the points as the mixture of J Gaussian distributions

$$p(\mathbf{z}_i | \boldsymbol{\theta}_j) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{d^2(\mathbf{z}_i, \boldsymbol{\theta}_j)}{2\sigma^2}\right) \quad (1 \leq j \leq J) \quad (16)$$

and a uniform distribution that represents points that are not assigned to any of the rectangles

$$p(\mathbf{z}_i|\boldsymbol{\theta}_*) = \begin{cases} 1/|V| & \mathbf{z}_i \text{ is in a bounding volume } V \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

In this probabilistic model, $d(\mathbf{z}_i, \boldsymbol{\theta}_j)$ indicates the minimum distance between the point $\mathbf{z}_i$ and a rectangle $\boldsymbol{\theta}_j$, and σ^2, V are design parameters determined for each of the specific applications. After initialization, the method iteratively evaluates log-likelihood (E-step) and updates the mixture weights (M-step) to find the best set of rectangles and point assignments.

This research makes the following adjustment to the method to better accommodate the proposed autonomous UAV navigation strategy, that is, the use of two-dimensional (2D) Gaussian distributions instead of the 1D distribution:

$$p(\mathbf{z}_i|\boldsymbol{\theta}_j) = \frac{1}{2\pi|\Sigma_d|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{d}^T(\mathbf{z}_i, \boldsymbol{\theta}_j)\Sigma_d^{-1}\mathbf{d}(\mathbf{z}_i, \boldsymbol{\theta}_j)\right) \quad (18)$$

where $\mathbf{d}(\mathbf{z}_i, \boldsymbol{\theta}_j) = [d_{\text{ip}}(\mathbf{z}_i, \boldsymbol{\theta}_j), d_{\text{op}}(\mathbf{z}_i, \boldsymbol{\theta}_j)]^T$ is now a 2D vector that contains in-plane and out-of-plane distance, and

$$\Sigma_d = \begin{bmatrix} \sigma_{\text{ip}}^2 & 0 \\ 0 & \sigma_{\text{op}}^2 \end{bmatrix} \quad (19)$$

is the associated covariance matrix. This adjustment enables the use of sparse points in-plane, while maintaining the quality of rectangles by only allowing small deviations in the out-of-plane directions. Based on the preliminary investigation that tests different values of $V, \sigma_{\text{ip}}^2, \sigma_{\text{op}}^2$, the values of σ_{ip} and σ_{op} are determined to be $1.0/\mu_s$ and $0.005/\mu_s$, respectively (μ_s is the current estimate of the point cloud scale), and the value of V is determined such that $p(\mathbf{z}_i|\boldsymbol{\theta}_j)$ and $p(\mathbf{z}_i|\boldsymbol{\theta}_*)$ balances at the distance of 1m in the in-plane direction.

Every time an image is obtained, the method initializes a predetermined number of rectangles (N_{rect}). While the original method initializes the rectangle parameters ($\boldsymbol{\theta}$) randomly, this research initializes rectangles by (i) selecting N_{rect} points with the ‘‘Column’’ label randomly from the points that have not been assigned to any of the rectangles, (ii) finding small number of

points with the “Column” label nearest to the selected point (5 points in this research), and (iii) fitting a rectangle to the points. After the rectangles are initialized, the online EM algorithm is applied for 10 iterations using points with the “Column” label that have not been assigned to any of the rectangles. Finally, model selection using Bayesian prior is applied to keep informative rectangles only, following [221].

After rectangles are detected by the online EM method, the results are refined by checking their dimensions. Based on the prior knowledge about the column dimensions (Table 2), rectangles are verified if one of their sides has the length in the range $[w^-/\mu_s, w^+/\mu_s]$, and the other side has the length in the range $[h^-/\mu_s, h^+/\mu_s]$. The rectangles that do not satisfy either of the conditions are discarded in this step. This research defines $[w^-, w^+]$ to be $[0.3, 1.425]$ (tolerance of 50% of the minimum and maximum dimensions), and $[h^-, h^+]$ to be $[2.0, 14.5]$, respectively. With this procedure, candidate column faces are identified incrementally, and at the same time the effect of outlier points are further reduced, because of the capability of the online EM method to model outlier points and the post-processing that discards low quality rectangles.

7.2.5 Column detection

Based on the rectangle detection results, the rectangular prismatic shapes are detected to recognize and localize bridge columns. This step is run in lower rate (every N_{cb} steps) to reduce the computational load. This step consists of three subtasks: (i) detect rectangular prismatic shapes from pairs of rectangles, (ii) recover rectangular prismatic shapes from unpaired rectangles, (iii) identify the grid patterns of the column locations, based on which the information about the unseen columns are inferred. The prior knowledge about the target structure encoded in this subsystem includes the dimensions of the grid patterns and the fact that each column faces non-bridge points at the bottom and beam points at the top. Such patterns are shared with many other railway viaducts

that are not necessarily designed based on the standardized procedure adopted by the Tokaido Shinkansen line.

(i) Column detection based on pairs of rectangles

Ideally, rectangular prismatic shapes can be detected by finding pairs of closely spaced rectangles that are either in parallel or perpendicular to each other. To identify such rectangle pairs, the system first finds the world coordinate system defined by the vertical direction and the two horizontal directions associated with the surface normal of column faces. First, when the predetermined number of rectangles (more than 4 rectangles in this research) has been identified, random sample consensus (RANSAC) algorithm [222] is applied to the rectangle surface normal vectors as follows: (i) select $n = \max(2, 0.25 * N)$ rectangles randomly (N denotes the total number of rectangles), (ii) create a $n \times 3$ matrix, A , whose i -th row is the i -th surface normal vector, (iii) considering that the rectangle surface normal is expected to be in the horizontal plane, estimate the vertical direction, n_v , by singular value decomposition of the matrix A , (iv) evaluate the dot product of n_v and each of the rectangle surface normal vectors, (v) select surface normal vectors with the absolute values of the dot product less than the threshold (0.05 in this research) as inliers, and (vi) repeat (i)-(v) for a predetermined number of iterations (10 iterations in this research), and select the case with the largest number of inliers. The final estimate of the vertical vector is obtained using those inliers.

The two other axes are determined by applying the RANSAC to the projection of rectangle surface normal vectors to the horizontal plane. First, two horizontal axes are initialized arbitrarily. This research defines the first horizontal axis in the direction of X axis of the point cloud coordinate system projected to the horizontal plane, and the second axis is defined accordingly. Using this initial coordinate system, the rotation that brings the X axis into the projected surface normal

vector, α , can be identified. Because the projected surface normal vectors are expected to exist in the multiples of 90-degree intervals, the remainder calculated by dividing α by 90 degrees, α_{rem} , is expected to be close to each other. Therefore, when more than 4 rectangles exist after identifying the vertical direction, RANSAC is applied to find the two typical horizontal directions of the column faces (number of samples $n = \min(5, \max(2, 0.25 * N))$, 20 iterations, threshold defined by the sin of angle difference of 0.05).

After identifying the world coordinate system that imposes consistency in the rectangle surface normal directions, this research applies consistency in the rectangle orientations about the surface normal direction. In this step, the two in-plane axes that define the rectangle sides are compared with the vertical direction of the world coordinate system. The rectangle is kept if the sin of the minimum absolute angle difference between the vertical direction and the rectangle in-plane axis is less than 0.1, and discarded otherwise. This step removes severely inclined rectangles that corrupt column detection results.

Each rectangle projected to the horizontal plane can be expressed using the following parameters: (i) a 2D vector of projected surface normal, $\mathbf{n}_{2D}$, (ii) a 2D vector of projected rectangle center, $\mathbf{c}_{2D}$, and (iii) the length of line segment obtained by projecting the rectangle (a scalar), L_{2D} . Example rectangle detection results expressed using those parameters are shown in Figure 44. Using the parameterization, two rectangles, $(\mathbf{n}_{2D}, \mathbf{c}_{2D}, L_{2D})$ and $(\mathbf{n}'_{2D}, \mathbf{c}'_{2D}, L'_{2D})$, are compared to evaluate whether they should be paired or not. The logic applied in this step is shown in Table 18. First, the two rectangles are classified into either nearly perpendicular or nearly parallel based on the dot product, $\mathbf{n}_{2D} \cdot \mathbf{n}'_{2D}$. At the same time, the 2D distance between the rectangle centers, $\|\mathbf{c}'_{2D} - \mathbf{c}_{2D}\|$, is evaluated and decomposed into the direction of the surface normal, $\mathbf{n}_{2D}$, and the direction perpendicular to $\mathbf{n}_{2D}$, yielding distance components, d_n and d_{ip} .

The logic for the nearly perpendicular case has been derived from the fact that the distance components of the two perpendicular column faces are halves of the width of the faces. Similarly, the logic for pairing nearly parallel rectangles has been derived from the fact that the distance between parallel faces in the surface normal direction is the width of the perpendicular column face. In addition to those cases, nearly parallel rectangles that are close to each other in the surface normal direction are merged, assuming that those rectangles come from the same column face. This case occurs when, for example, the lower part of the column face is detected first, and the upper part is detected later.

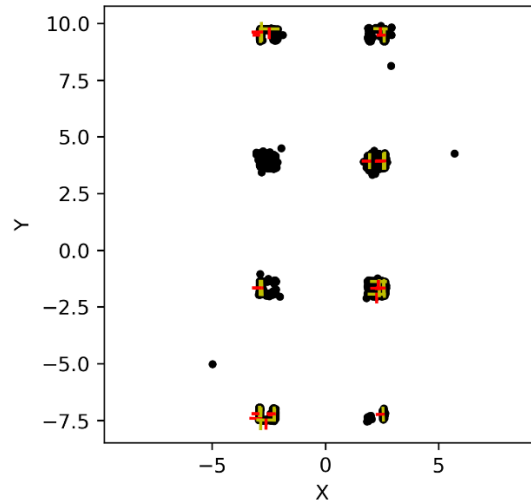


Figure 44 Example of detected rectangles (yellow) and their surface normal vectors (red) illustrated in the identified X-Y plane. Points indicate the 3D points from the sparse point cloud with the valid “Column” label.

Table 18 Logic applied to find rectangle pairs.

d_n and d_{ip} indicate distances between $\mathbf{c}_{2D}$ and $\mathbf{c}'_{2D}$ in the surface normal direction and its perpendicular direction, respectively.

Nearly perpendicular ($ \mathbf{n}_{2D} \cdot \mathbf{n}'_{2D} < 0.05$)	Nearly parallel ($ \mathbf{n}_{2D} \cdot \mathbf{n}'_{2D} > 0.95$)
Paired if $w^-/2\mu_s < d_n < w^+/2\mu_s$ and $w^-/2\mu_s < d_{ip} < w^+/2\mu_s$	Paired if $w^-/\mu_s < d_n < w^+/\mu_s$ and $d_{ip} < w^+/2\mu_s$ Merged if $d_n < 0.005/\mu_s$ and $d_{ip} < 0.5/\mu_s$

Once rectangle pairs are identified, rectangular prismatic shapes representing the viaduct columns are initialized. In this step, the 2D projection of the column (rectangular cross section) is first obtained using $(\mathbf{n}_{2D}, \mathbf{c}_{2D}, L_{2D})$ and d_n . The ends of the columns in the vertical axis is not

straightforward to identify based on the rectangles alone, because in this online approach, the rectangles do not necessarily span the entire column face from bottom to top. To address this challenge, this research uses a prior knowledge about the target structure, that is, columns face non-bridge surface (ground) at the bottom and beams at the top. Using 3D points with the valid labels of those categories, this research determines the lower and upper ends of the columns by the following steps: (i) Select the rectangle vertex with either the minimum (lower end) or maximum (upper end) vertical coordinate, (ii) select 20 nearest points with the valid label of either non-bridge (lower end) or beam (upper end), and (iii) compute the mean of the vertical coordinates of those selected points using RANSAC (20 iterations, sampling 5 points in each iteration, and the threshold of selecting inliers to be 0.15). If the RANSAC fails (e.g. no inlier point is found because sufficient number of points do not exist near the column ends), the vertical coordinates of the column ends are replaced by those of the nearest column. To complete the column initialization, the horizontal distances between column centers are checked, and closely spaced columns (the horizontal distance is less than w^+/μ_s) are regarded as duplicates, from which only one column is chosen for the subsequent analysis.

The final step of column detection using rectangle pairs is to fit the rectangular prismatic shapes to 3D points. This step applies the iterative closest point (ICP) algorithm with the point-to-plane distance metric [223]. To apply the method, this research first parameterizes each column by a transform of a unit cube centered on the origin $[0, 0, 0]^T$, that is,

$$T(\theta_{cb}) = \begin{bmatrix} RS & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (20)$$

where R is a rotation matrix, $\mathbf{t}$ is the location of the column center, and S is a matrix that applies appropriate scale to the unit cube, or

$$S = \begin{bmatrix} \lambda_0 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_2 \end{bmatrix} \quad (21)$$

This transform is 9 DOF (note that [223] focuses on 6DOF rigid body motion):

$$\boldsymbol{\theta}_{cb} = [\lambda_0 \quad \lambda_1 \quad \lambda_2 \quad \theta_x \quad \theta_y \quad \theta_z \quad t_x \quad t_y \quad t_z]^T \quad (22)$$

where $[\theta_x \quad \theta_y \quad \theta_z]^T$ are Euler angles and $\mathbf{t} = [t_x \quad t_y \quad t_z]^T$. Using this parameterization, the incremental transform,

$$d\boldsymbol{\theta}_{cb} = [d\lambda_0 \quad d\lambda_1 \quad d\lambda_2 \quad d\theta_x \quad d\theta_y \quad d\theta_z \quad dt_x \quad dt_y \quad dt_z]^T \quad (23)$$

can be defined as the transform that leads to the following composite transform:

$$T(\boldsymbol{\theta}_{cb}) = \begin{bmatrix} R'S' & \mathbf{t}' \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (24)$$

where

$$S' = \begin{bmatrix} \lambda_0 + d\lambda_0 & 0 & 0 \\ 0 & \lambda_1 + d\lambda_1 & 0 \\ 0 & 0 & \lambda_2 + d\lambda_2 \end{bmatrix} \quad (25)$$

$$\begin{bmatrix} R' & \mathbf{t}' \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} dR & d\mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (26)$$

(dR and $d\mathbf{t}$ are the rotation matrix and translation vector associated with $d\boldsymbol{\theta}_{cb}$). Among the 9 parameters of $d\boldsymbol{\theta}_{cb}$, 5 parameters ($d\lambda_2, d\theta_x, d\theta_y, d\theta_z, dt_z$) are constrained, because those parameters specify the world coordinate system and the vertical location of the column ends. Therefore, the rectangular prismatic shape fitted in this step has 4DOF.

The ICP in this research proceeds by the following steps. First, 3D points are transformed to the coordinate system of the unit cube by applying $T^{-1}(\boldsymbol{\theta}_{cb})$, as shown in Figure 45. Then, the points that exist in the height range of $[-0.5, 0.5]$ and the horizontal distance range of $[0, \sqrt{5}/2]$ are selected for use of the ICP algorithm (other points are grayed out in Figure 45). The selected points are assigned to one of the four sides of the cube, based on the four regions illustrated by different colors in Figure 45. The objective function used at each iteration is written as

$$J = \sum_i (d_i * \lambda_i)^2 \quad (27)$$

where summation is computed over all the selected points that are assigned to one of the sides, d_i is the horizontal distance from each point to the nearest side, and λ_i is the scale of the column in the corresponding direction. The objective function J is minimized with respect to the 5 parameters of θ_{cb} using least square method and, based on the updated parameters, the point assignments are updated. This process is repeated until the change of the objective function becomes less than 0.001 or the number of iterations reaches 10 times.

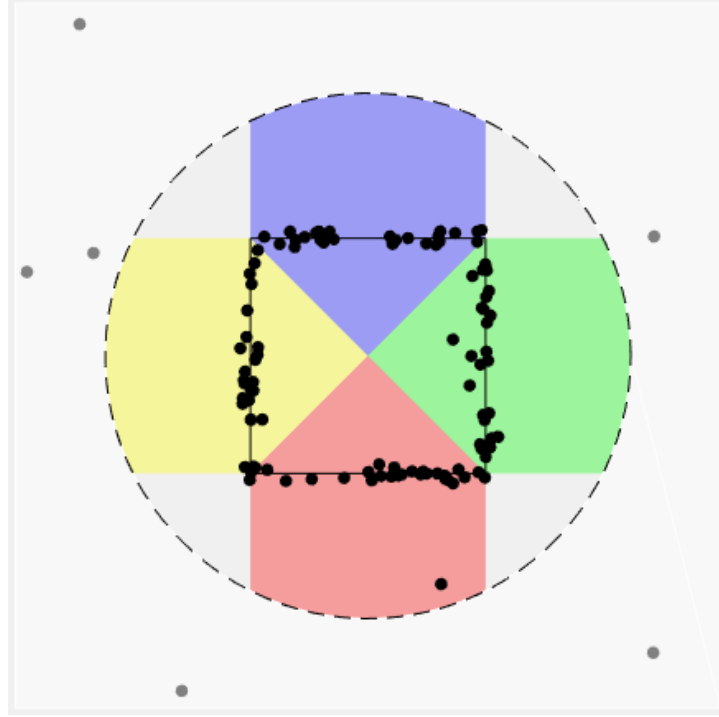


Figure 45 Point assignment for the iterative closest point (ICP) algorithm. The length of each side of the square is 1, and the radius of the circle is $\sqrt{5}/2$.

The ICP does not always converge to the correct solution when the number of points or the quality of points are not sufficient. To handle such cases, this research checks if the dimensions of the cross-sections of the column after the ICP are in the range $[0.55/\mu_s, 1.0/\mu_s]$, and reset the parameters to the initial values if the condition is not satisfied (this condition is more strict than the conditions used previously, that is, $[w^-/\mu_s, w^+/\mu_s]$, because this is the final check).

The columns detected by pairs of rectangles are reliable and robust to measurement noise. On the other hand, not all columns are detected by this approach, because at least two faces of the columns need to be detected accurately. To augment the column detection results, methods to infer information about columns from one or no rectangle is discussed next.

(ii) Column detection based on unpaired rectangles

Mathematically, a column (square prismatic shape) cannot be identified uniquely from only one rectangle. To obtain reasonable estimates of such columns, this research first detects the global grid-type pattern of the column locations. In the standard design adopted by the Tokaido Shinkansen, the column intervals are 5.2m in the transverse direction, and 6.0m in the longitudinal direction. This prior knowledge can be used to improve the accuracy and robustness of the column detection results. Moreover, this grid-type pattern is shared with many other railway viaducts, making the generalization to those structures straightforward by simply adjusting the values of the column interval distances.

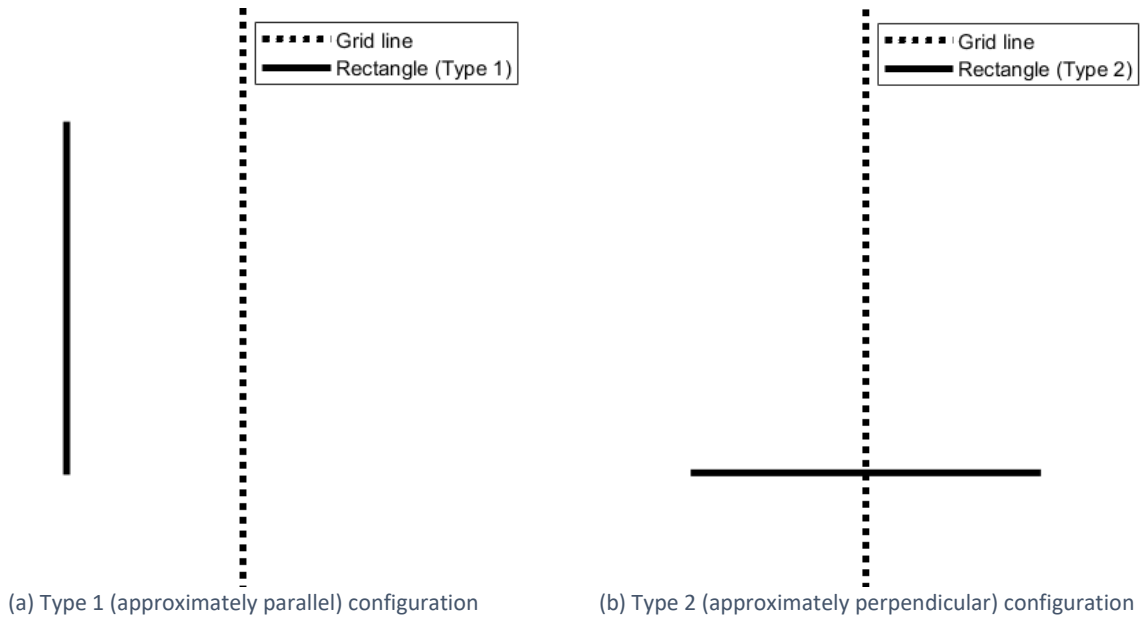
The grid lines perpendicular to the X and Y axes of the world coordinate system can be represented by the X- and Y-coordinates:

$$\begin{aligned} \text{Grid lines (X): } g_x &= \{g_{x1}, g_{x2}, \dots\} \\ \text{Grid lines (Y): } g_y &= \{g_{y1}, g_{y2}, \dots\} \end{aligned} \tag{28}$$

The first step of detecting the grid lines is to assign each column to one of the grid lines in X and Y directions. This research initializes the grid lines using the columns detected based on the rectangle pairs. For each column, a new grid line is defined if no other column exists within the distance w^+/μ_s in the associated direction, and otherwise the column is assigned to the existing grid line defined by the closest column. The grid lines are then defined by computing the mean of the X or Y coordinates of the associated column centers. When at least two grid lines are detected in both directions, the prior knowledge about the column intervals is applied to align the X and Y

axes to the transverse and longitudinal directions, respectively (Figure 44): if the minimum grid line interval of g_y is less than that of g_x , the world coordinate system is rotated by 90 degrees.

The next step is to assign unpaired rectangles to the grid lines. Two types of rectangle configurations are considered herein: (i) the rectangle is approximately parallel to the grid line, and (ii) the rectangle is approximately perpendicular to the grid line. The two types of configurations are illustrated in Figure 46. Based on this classification, the type 1 rectangles are assigned to the grid line when the distance from the rectangle center to the line is in the range $[w^-/2\mu_s, w^+/2\mu_s]$, and the type 2 rectangles are assigned to the grid line when the distance from the rectangle center to the line is less than $w^+/2\mu_s$.



The information about the column is then inferred based on the rectangle and the associated grid line. For type 1 rectangles, the column is defined by creating another face symmetrically from the rectangle about the grid line. Steps to determine other parameters, such as vertical coordinates

of the bottom and top ends, follow the steps discussed for the column detection using rectangle pairs. A column is not defined for type 2 rectangles at this step. Instead, a grid line passing the rectangle center is added to the set of grid lines when the rectangle is not assigned to any of the grid lines of the type 2 configuration.

To complete this step, grid lines are refined, and the accurate estimate of the scale of the point cloud, μ_s , is obtained. First, the grid lines in the longitudinal direction are determined by selecting the two most reliable lines from the set g_x . The selection is based on the score defined by the following equation:

$$n_x = |I_{\text{paired}}| + 0.5 * |I_{\text{unpaired}}| \quad (29)$$

where I_{paired} and I_{unpaired} denote the sets of columns detected by pairs of rectangles and by unpaired rectangles, respectively, and $|\cdot|$ denotes the number of elements in a set. This score represents the number of detected column faces associated with each grid line. The selected lines, $g_x = \{g_{x1}, g_{x2}\}$, are used to obtain the accurate estimate of the scale μ_s by

$$\mu_s = \frac{D_{\text{trans}}}{|g_{x2} - g_{x1}|} \quad (30)$$

where D_{trans} is the known grid line interval in the transverse direction (5.2 in this research).

Once this estimate is obtained, the scale updating based on the monocular depth estimation is terminated.

The grid lines perpendicular to Y axis, g_y , are refined by applying the prior knowledge that the column is located with the regular interval D_{long} (=6.0 in this research). First, a distance matrix D is defined such that the (i, j) element contains distance between the i -th and j -th lines, that is,

$$D_{ij} = |g_{yj} - g_{yi}| \quad (31)$$

If the grid lines are detected accurately, the elements of D is the multiple of D_{long} . This research interprets that the i -th grid line is valid if there is at least one value of j ($\neq i$) that satisfies

$$\left| D_{ij} - D_{\text{long}} \cdot \text{Round}\left(\frac{D_{ij}}{D_{\text{long}}}\right) \right| < \frac{w^+}{2\mu_s} \quad (32)$$

where $|\cdot|$ denotes absolute value, and $\text{Round}(\cdot)$ denotes the rounding operation to the nearest integer. The refined grid lines are used to add missing columns in the next subsection.

(iii) Column detection from no rectangle

The final stage of the column detection is to add missing columns based on the awareness to the global structure, or more specifically, grid lines, g_x and g_y . First, g_y is converted to integer values that represents the number of intervals (D_{long}) counted from the line with the minimum value of g_{yi} . If missing integer values are identified between the minimum and maximum values, those values are added to g_y as new grid lines. Then, the intersections between g_x and g_y are identified as the expected column locations. If no column exists in the square with the side w^+/μ_s centered at each candidate location, a column is added to that location, where the parameters other than the column center is obtained from the nearest column detected previously. At the same time, columns detected previously whose centers do not exist inside the square with the side w^+/μ_s centered at each candidate location are regarded as outliers and discarded.

7.2.6 Waypoint determination

Vision-based recognition accuracy of structural damage, such as cracks, spalling, and exposed rebar, is sensitive to the viewpoint to the target surface and, in particular, the distance to the damage. For example, Narazaki et al. [224] performed the semantic segmentation of structural damage using deep FCN, where the distance to the target surface is at least 2 pixels per centimeter (pixel/cm). The network does not always perform accurately for far surfaces, and therefore the regions with less than 2 pixel/mm has been excluded from the consideration. To accommodate such image post-processing approaches, this research determines the waypoints by the following

strategy. First, a distance to the target structure in pixel/cm, N_{px} , and the ratio of overlap between adjacent images, R , are determined. Then, for each column, the waypoints that collect images of all faces with the distance N_{px} and the overlap R are determined. Example waypoints and the path connecting the waypoints are shown in Figure 47.

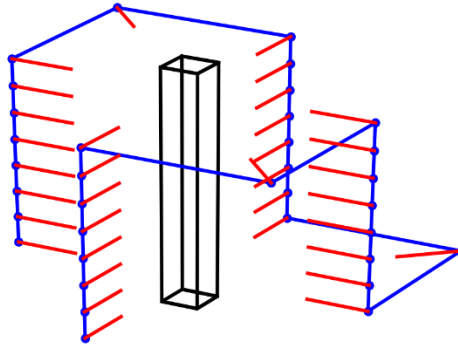


Figure 47 Example waypoints with camera directions shown by red bars. The waypoint after scanning each face is inserted to avoid the UAV flying too close to the column.

($N_{px} = 7$, $R = 0.5$, focal length: 35mm, sensor size: 36mm, resolution: $1,920 \times 1,080$)

The UAV starts the mission by flying along a predetermined GPS-based path around the target structure until the following two conditions are satisfied: (i) the scale of the structure (grid lines) has been identified, and (ii) at least two columns are detected. Once those conditions are satisfied, the UAV switches from the GPS-based navigation to the navigation using the inspection waypoints (Figure 47). The navigation path is defined such that the UAV inspects the columns in the following order: from reliable columns (columns detected based on rectangle pairs) to less reliable ones (columns detected based on a single rectangle, and then columns detected from no rectangle). The order of inspection within each reliability category is determined based on the distance from the camera (from close columns to far columns). This research uses the A* algorithm [225] in the PythonRobotics library [226] to define the 2D path connecting inspection waypoints of different columns in the horizontal plane, and linearly interpolates the vertical coordinates of

the origin and destination to realize the 3D path. Throughout the navigation process, the column detection results and the navigation path are updated every 10 frames (if the 10-th frame is during the column inspection depicted in Figure 47, the updates are deferred until the end of the inspection).

The autonomous UAV navigation strategy for rapid post-earthquake inspection of RC railway viaducts discussed in this section is summarized in the pseudocode in Figure 48. With this strategy, the system can start planning the navigation path once the partial information of the structure (grid lines and first two columns) is obtained, without waiting for the complete information of the structure. The system can also make inferences of missing or partially occluded columns during early stages of the navigation, and progressively improve the information for those columns as the system experiences more viewpoints.

```

Set frequency of running column detection:  $N_{cb} = 10$ 
Set rectangle detection parameters:  $N_{rect}, V, \sigma_{ip}^2, \sigma_{op}^2$ 
Set known prior knowledge about the target structure:  $w^-, w^+, h^-, h^+, D_{long}, D_{trans}$ 
Set waypoint generation parameters:  $N_{px}, R$ , focal length, sensor size, resolution
For each frame
    (Update sparse point cloud data)

    # Frame-wise image processing and point cloud parsing
     $S \leftarrow$  Semantic segmentation
    Point labels and validity  $\leftarrow$  Update using  $S$ 
    If  $\mu_s$  has not been fixed
         $D \leftarrow$  Monocular depth estimation
         $(\mu_s, \sigma_s) \leftarrow$  Update using  $D$ 

    # Rectangle detection
    Select  $N_{rect}$  seed points randomly  $\leftarrow$  Column points that have not been assigned to any of the rectangles.
    Rectangles  $\leftarrow$  5 nearest column points for each seed point.
    Update rectangles by EM
    Size check  $\leftarrow w^-, w^+, h^-, h^+$ 

    # Column detection and waypoint determination
    For every  $N_{cb}$  steps
        # Column detection based on pairs of rectangles
        Identify world coordinate system.
        Find rectangle pairs.
        Columns  $\leftarrow$  Column detection based on pairs of rectangles.
        Fit columns to points using ICP (reset if ICP is not successful).

        # Column detection based on unpaired rectangles
        Initialize grid lines.
        Align Y axis to the longitudinal direction.
        Assign unpaired rectangles to existing grid lines, or create new grid lines.
        Add columns  $\leftarrow$  Type 1 unpaired rectangles.
        Refine grid lines and update scale.

        # Column detection from no rectangle
        Add missing columns using intersections of the refined grid lines and remove duplicates.

    #Waypoint determination
    If grid lines are identified and at least two columns are detected
        Sort columns by reliability and distance.
        Update inspection waypoints  $\leftarrow N_{px}, R$ 
        Update navigation path.

```

Figure 48 Pseudocode for the autonomous UAV navigation strategy for rapid post-earthquake inspection of RC railway viaducts.

7.3 Demonstration using synthetic environment

7.3.1 Synthetic environment of RC railway viaducts

This research demonstrates the proposed autonomous UAV navigation strategy using a synthetic environment of RC railway viaducts developed for [224]. In the synthetic environment, viaducts are modeled following the standard design procedure adopted by the Tokaido Shinkansen

line. Following the procedure discussed in section 3.3, this research generates an environment that contains three viaducts with straight trajectory, zero slope, and the height of 7m, where the autonomous navigation strategy for the inspection of the central viaduct is investigated. The overview of the synthetic environment is shown in Figure 49.

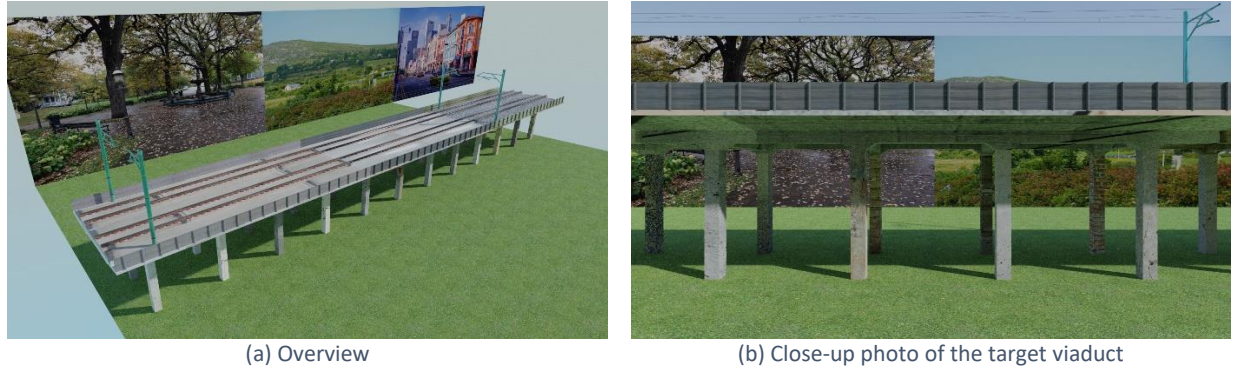


Figure 49 Synthetic environment of RC railway viaducts

Based on the assumptions 4-5 discussed previously, this section defines a sample UAV path around the bridge that collects 235 images with the resolution of $1,920 \times 1,080$, sensor size of 36mm, and focal length of 35mm. The UAV scans each side of the viaduct for three height levels (2m, 5m, 8m) and flies to the other side, performing the scan at the same three height levels. During flight, the UAV collects images at 1m intervals. The sparse point cloud data is then generated using Agisoft Metashape photogrammetry software [218]. The sparse point cloud data and the UAV path are shown in Figure 50. The sparse point cloud data contains 67,114 points in total.

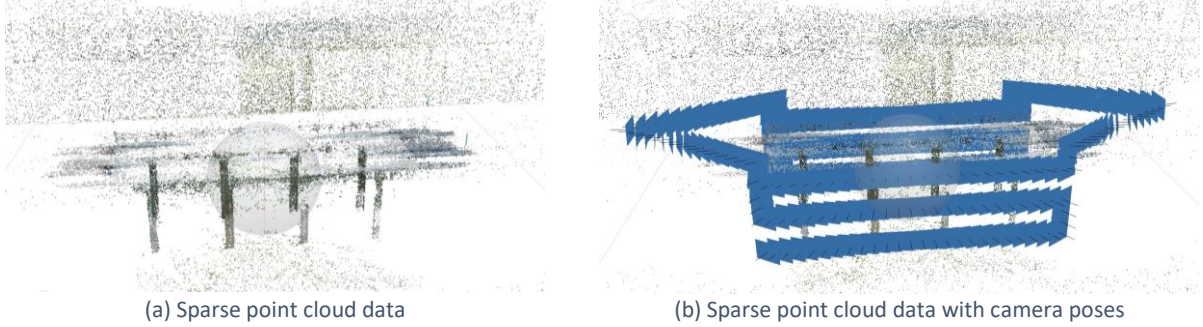


Figure 50 Sparse point cloud data obtained by the sample UAV flight

This section first demonstrates the UAV navigation strategy by feeding the part of the sparse point cloud data associated with each image to the waypoint determination system incrementally. After evaluating the system's capability of detecting columns robustly and reliably, the section extends the discussion to incremental sparse point cloud building and progressive improvement and addition of waypoints.

7.3.2 Demonstration using images from sample UAV flight

Example results of point cloud parsing, rectangle detection, and column detection for the sample UAV flight data with the number of rectangles per step (N_{rect}) of 1 are shown in Figure 53. In an online procedure, the 2D semantic segmentation can parse the point cloud data and extract points associated with the target structural components. Then, the rectangle detection can extract key geometry of the column faces. Finally, the columns can be detected from rectangle pairs, a single rectangle, or no rectangle. As the UAV experiences more views, the number of columns detected from rectangle pairs increases, indicating the progressive improvement of the results. Furthermore, the improved column locations and dimensions stay near the corresponding estimations from a single or no rectangle, demonstrating the effectiveness of the column detection based on the awareness to the structure.

To investigate the effect of the selection of the value of N_{rect} , the column detection is performed 100 times for the sample UAV flight data with $N_{\text{rect}} = 1, 2, 3, 4$. The numbers of rectangles detected for each of the four cases are shown in Figure 52. Compared to the case with $N_{\text{rect}} = 1$, the case with $N_{\text{rect}} = 2$ detects more rectangles at faster rate. When the value of N_{rect} further increases, the number of rectangles increases accordingly. However, the refinement of the detection results at every $N_{\text{cb}} (= 10)$ frame lower the rate of increase by either discarding or merging rectangles. When the value of N_{rect} is too large, most of the detected triangles are low quality and discarded by the refinement process (RANSAC for identifying the world coordinate system).

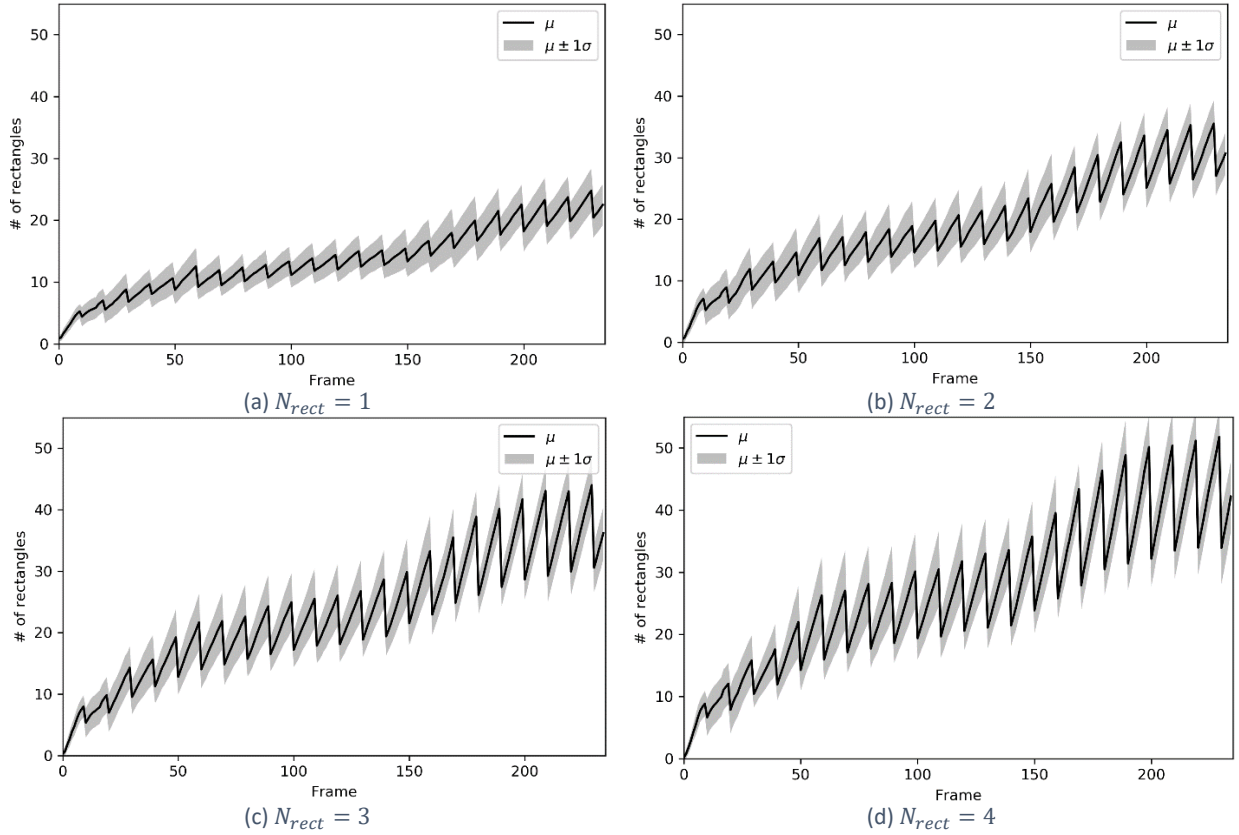


Figure 52 Number of rectangles detected with different values of N_{rect} .

The numbers of columns detected for each of the four cases are shown in Figure 53. Each figure shows five lines, corresponding to the maximum, top 25%, median, top 75%, and minimum number of rectangles detected during 100 runs. The figures also show that the columns tend to be detected more quickly for larger values of N_{rect} . When $N_{rect} \geq 2$, eight columns are detected before processing all frames in all 100 runs. On the other hand, when $N_{rect} \geq 3$, the number of detected columns decreases occasionally after detecting 8 columns, which is caused by the increasing number of noisy rectangles observed in Figure 52. While this issue can be addressed by setting larger number of iterations for RANSAC, the subsequent part of this research discusses the case with $N_{rect} = 2$ with the current setting, because the case simply detects less noisy rectangles.

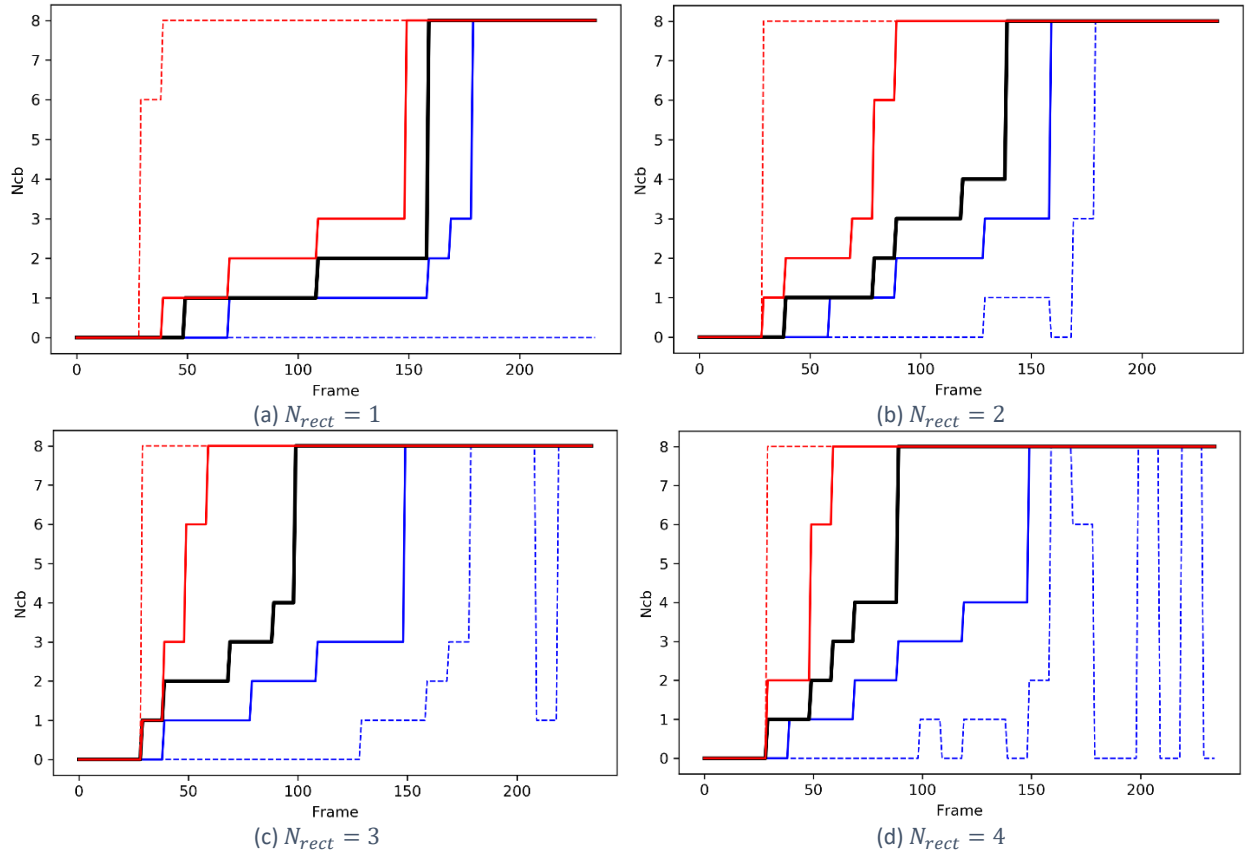


Figure 53 Number of columns detected with different values of N_{rect} . (Red dotted: maximum, red solid: top 25%, thick black: median, blue solid: top 75%, blue dotted: minimum)

The number of points with the valid “Column” label that are not assigned to any of the rectangles, as well as the point cloud scale estimation results are shown in Figure 54 for the case with $N_{\text{rect}} = 2$. Among 67,114 points contained in the point cloud data, up to about 4,000 points (6.0%) are analyzed to detect rectangles and columns. The number of points further decreases as the method finds valid rectangles representing column faces. The point cloud scale is approximate initially (about 0.62), which converges to the true value (0.94) once the grid pattern is identified (every run gives the same value for the scale estimation based on the monocular depth estimation, and therefore the standard deviation is zero initially).

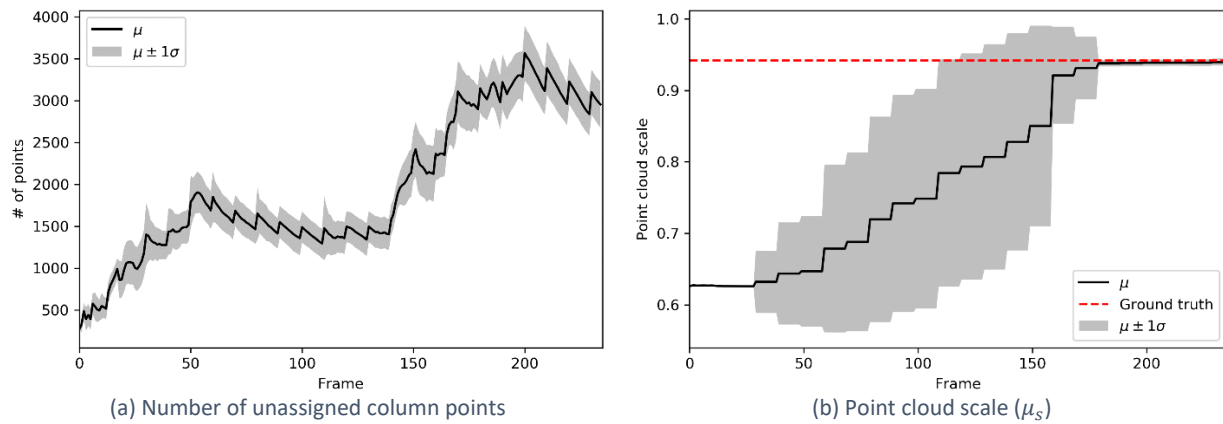


Figure 54 Detailed results for the case with $N_{\text{rect}} = 2$.

The accuracy of the estimation of the locations and dimensions of the detected columns are evaluated using the ground truth values obtained from the synthetic environment. The error standard deviation of the estimated column center locations and orientations are evaluated after the 100 runs with $N_{\text{rect}} = 2$ and presented in Table 19 for three cases: columns detected based on rectangle pairs (“Blue”), columns detected based on unpaired rectangles (“Yellow”), and columns detected from no rectangle (“Red”). The error standard deviations are less than 7cm when the column is detected from at least one rectangle, and less than 11cm even when no rectangle is

detected. The means and standard deviations of the column dimensions estimated after the 100 runs with $N_{\text{rect}} = 2$ are shown in Table 20, where w_x is the column side length in longitudinal direction (ground truth is 0.6m), w_y is the column side length in the transverse direction (ground truth is 0.6m), and h is the column height (ground truth is 5.3m).

Table 19 Column detection accuracy (error standard deviation of column center location and orientation)
Blue: columns detected based on rectangle pairs, yellow: columns detected based on unpaired rectangles, red: columns detected from no rectangle.

	STD(Blue)	STD(Yellow)	STD(Red)
X[m]	0.053	0.060	0.106
Y[m]	0.040	0.046	0.065
Z[m]	0.065	0.023	0.043
θ_z [rad]	0.015	0.015	0.016

Table 20 Column detection accuracy (dimensions)[m]
Blue: columns detected based on rectangle pairs, yellow: columns detected based on unpaired rectangles, red: columns detected from no rectangle.

	Mean(Blue)	STD(Blue)	Mean(Yellow)	STD(Yellow)	Mean(Red)	STD(Red)
w_x	0.625	0.079	0.611	0.093	0.597	0.047
w_y	0.584	0.079	0.605	0.106	0.580	0.027
h	5.439	0.045	5.433	0.055	5.419	0.055

The step processing time for the case with $N_{\text{rect}} = 2$ is evaluated using a desktop computer with an i7 6700 3.40 GHz processor, NVIDIA GTX 1070 8 GB graphics card and 16 GB RAM. The results are shown in Figure 55. The processing time for point cloud parsing and rectangle detection is shorter than that of column detection step, which runs in the lower rate (and possibly not needed after identifying all columns reliably). While the results are promising, the investigation of the computational aspect of the autonomous navigation strategy is preliminary in this research: further investigations are needed to enable the on-board processing toward field implementation of the strategy, such as the optimization of the FCN architectures, multi-thread processing, and implementation of additional logics that remove data associated with columns that have been

inspected already or has not been assigned to valid rectangles for a long time. Those investigations are part of future work.

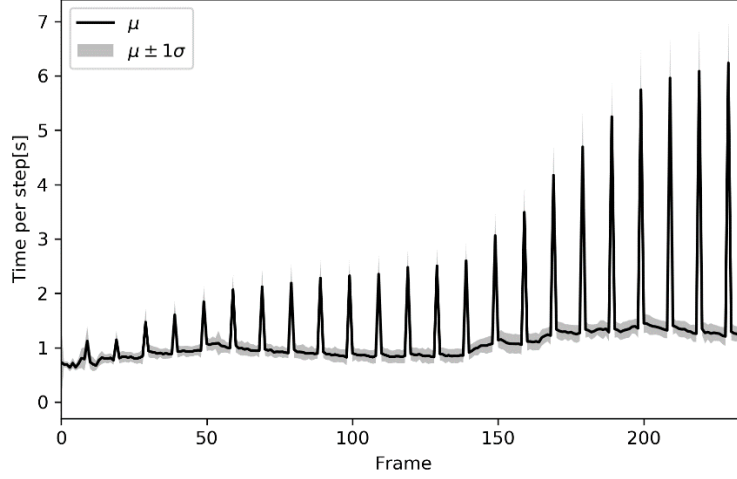


Figure 55 Step processing time ($N_{rect} = 2$).

7.3.3 Demonstration of autonomous navigation strategy

This section demonstrates the autonomous UAV navigation strategy following the procedure illustrated in Figure 56. First, the UAV flies along a predetermined GPS-based path (Figure 50) to initiate the mission. In contrast to the previous section that computed the sparse point cloud data off-line using the entire dataset, this section takes a more realistic approach of incremental reconstruction using batches of 10 images. Every time the new 10 images are processed by the SfM, the column detection algorithm discussed in this research is applied. If the scale (grid lines) and at least two columns are detected, the UAV switches from the initial GPS-based navigation to the navigation using the inspection waypoints (“Inspection phase” that uses waypoints shown in Figure 47), and otherwise stays in the GPS-based path. Once the inspection phase is initiated, image batches are rendered in the synthetic environment as needed to enable the demonstration of the progressive path planning and refinement approach.

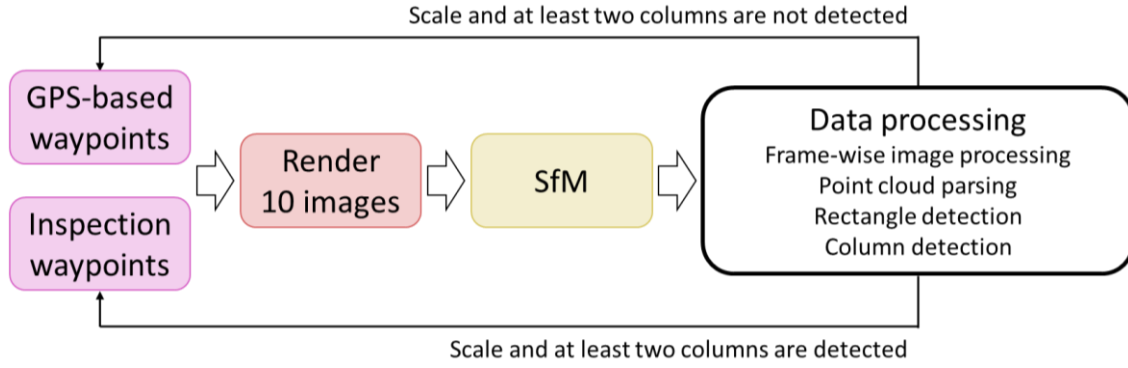


Figure 56 Data processing procedure for the demonstration of the autonomous navigation strategy

During the demonstration, the conditions for switching from the GPS-based navigation to the one using the inspection waypoints are satisfied at the 60th frame. The parsed sparse point cloud data, navigation path in 2D plane, and navigation path in 3D space are shown in Figure 57 at every 80 frames after entering the inspection phase. The figures illustrate that the proposed strategy can plan a UAV navigation path for rapid post-earthquake structural inspection automatically based on the understanding of the target structure. Furthermore, columns in the adjacent viaducts are detected toward the end of the inspection of the current viaduct, enabling smooth transition to the inspection of the next structure.

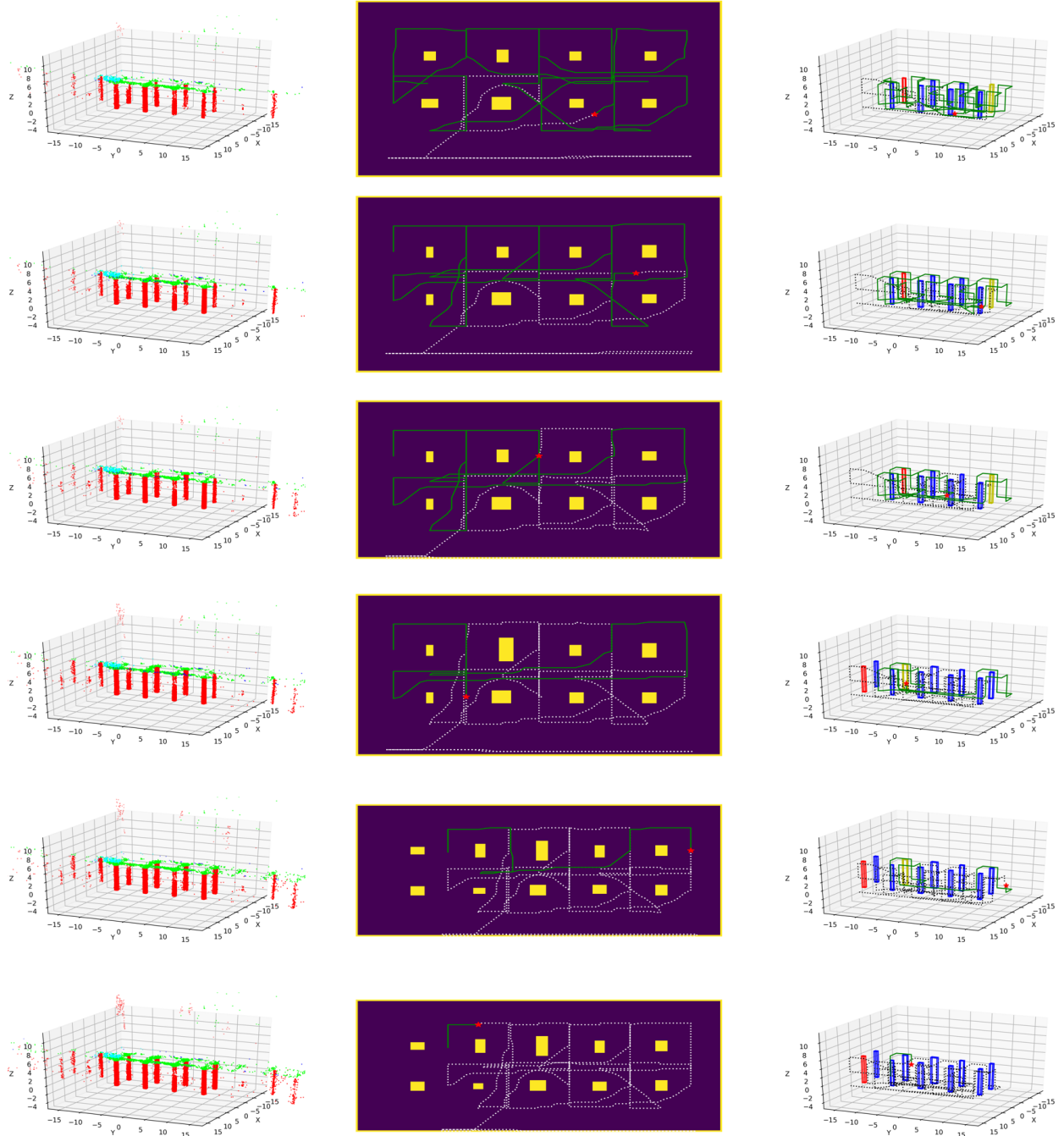


Figure 57 Parsed sparse point cloud, navigation path (2D), and navigation path (3D) at every 80 frames from 80th to 480th frame after entering the inspection phase. The dotted lines are finished parts, and the green solid lines show the planned path to be followed.

Images of the first column inspected during the inspection phase are shown in Figure 58.

As expected, images from the close and controlled distance are collected with overlaps, which are

ideal for the automated post-earthquake structural damage recognition as discussed in Narazaki et al. [224].



Figure 58 Images of the four faces of a column collected by the proposed UAV navigation strategy ($N_{px} = 7$, $R = 0.5$).

7.4 Summary and conclusions

This chapter developed vision-based autonomous UAV navigation strategies for rapid post-earthquake inspection of reinforced concrete railway viaducts. The strategy mimics the way human inspectors perform the task: the system does not require the complete 3D model of the

environment, and instead use the key characteristics of the target structure as prior knowledge. In an on-line manner, the strategy first parses the sparse point cloud data using frame-wise semantic segmentation results, and identifies the approximate scale using the frame-wise depth estimation results. Then, rectangles are fitted to the points with the “Column” label, from which columns are detected by finding near-parallel or near-perpendicular pairs. In addition to the columns thus detected, the strategy tries to find the grid-line pattern of the column locations, based on which the detection results are refined, and information about missing columns are inferred. With this approach, the UAV can initiate the inspection mission before obtaining complete information of the target structure. The information (map) can be improved progressively as the UAV experiences more views during inspection. The capability of the developed strategy in detecting columns robustly and reliably was evaluated using the synthetic data. The results showed that all eight columns of the target viaducts are detected in all 100 trials with centimeter-level accuracy. Finally, the entire strategy was demonstrated in the synthetic environment, showing the significant potential of collecting high-quality images for post-earthquake structural inspection efficiently and quickly.

The strategy investigated in this research is a prototype for the autonomous UAV navigation for the post-earthquake structural inspection, and therefore needs to be improved to accommodate field implementation. First, this research postulates a simplifying assumption on the sparse point cloud data generation: this research applies the SfM to image batches. While producing accurate results, the approach is computationally expensive, and is not necessarily optimal for the mobile robotics applications. Investigation of other computational aspects of the strategy is also preliminary in this research, and further investigation is needed to enable real-time and on-board data processing. Finally, the demonstration in this research is performed in the

synthetic environment, which is free of uncertainty encountered in the field environments, such as obstacles and modeling imperfections. The robustness of the proposed approach will be improved in the future to enable the field implementation.

CHAPTER 8: VISION-BASED POST-EARTHQUAKE STRUCTURAL ASSESSMENT OF RC RAILWAY VIADUCTS

This chapter discusses an approach to post-process images of RC railway viaducts to perform rapid post-earthquake structural inspection automatically. The approach is based on the deep fully convolutional network for semantic segmentation (discussed in Chapter 6), and integrates the discussions in previous chapters to obtain results that are consistent with the current practice of human-based visual inspection. This chapter completes the prototype system for the automated rapid post-earthquake inspection using vision-based techniques that is envisioned in this research.

8.1 Vision-based post-earthquake structural assessment methods

8.1.1 Overview

Steps of the vision-based post-earthquake structural assessment is depicted in Figure 59, with the relations to previous chapters of this research. The vision-based structural assessment investigated in this chapter has two main components: (i) damage recognition, and (ii) the refinement and interpretation of the damage recognition results. Damage recognition is performed by applying a semantic segmentation algorithm (FCN) to images collected by the automated process discussed in Chapter 7. Damage recognition results are then refined and interpreted with the help of bridge component recognition and localization results discussed in Chapter 4-Chapter 6. By these processing steps, the input image is converted to a masked image that contains information about damage existence, damage sizes and shapes, structural components where the damage is observed, and the reliability of the structural assessment results. The synthetic environment discussed in Chapter 3 keeps playing important roles for developing, implementing, and testing methods investigated in this chapter.

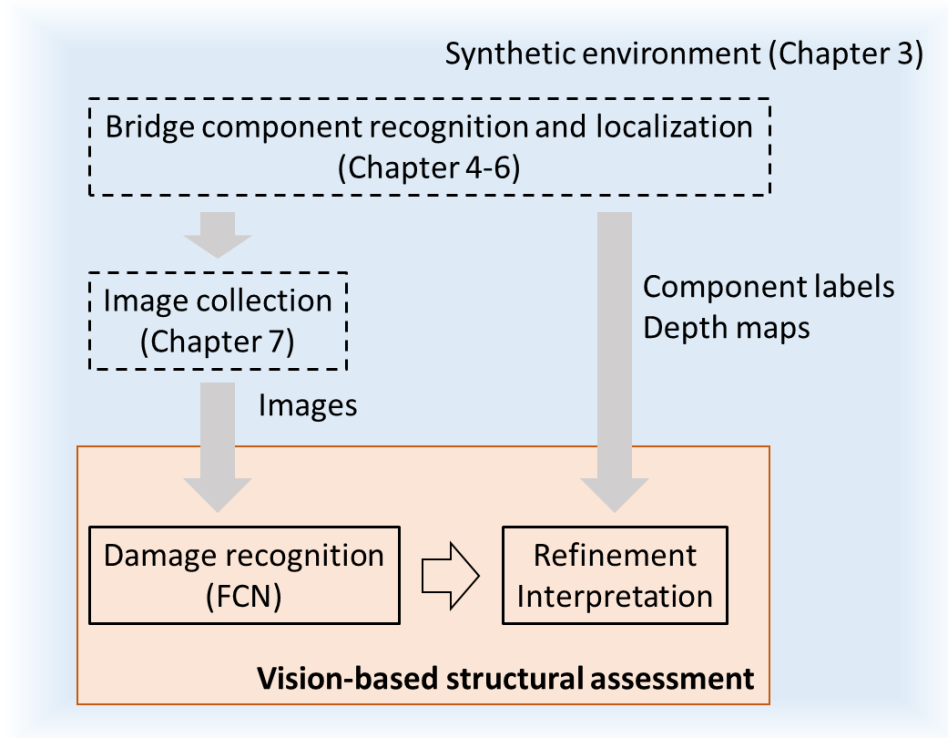


Figure 59 Steps of vision-based post-earthquake structural assessment and the relations to other chapters of this research

8.1.2 Deep fully convolutional network for damage recognition

The deep FCN discussed in Chapter 6 is used to perform the semantic segmentation task of damage recognition. The synthetic Tokaido dataset is used to augment the training data made of small number of real-world images. A challenge of damage recognition using FCN is that damage often exhibits thin or subtle patterns. Preliminary analysis has shown that learning accurate segmentation for structural damage with various sizes and shapes in the entire distance range of the dataset is difficult, even with the deep FCN58 architecture. In particular, when the damage exists on the surface of structural components that are far from the camera, images do not have enough resolutions to capture such damage patterns. On the other hand, ground truth damage labels are always available, regardless of whether the damage patterns are represented by the available pixels sufficiently. These ground truth labels confuse the network during training, introducing large errors to the recognition results by the trained networks. Example results of training FCN

using regular images (defined in section 3.3) are shown in Figure 60. The results show that the damage recognition is inaccurate (large amount of false positive detections, particularly for far columns) due to the challenge discussed herein.

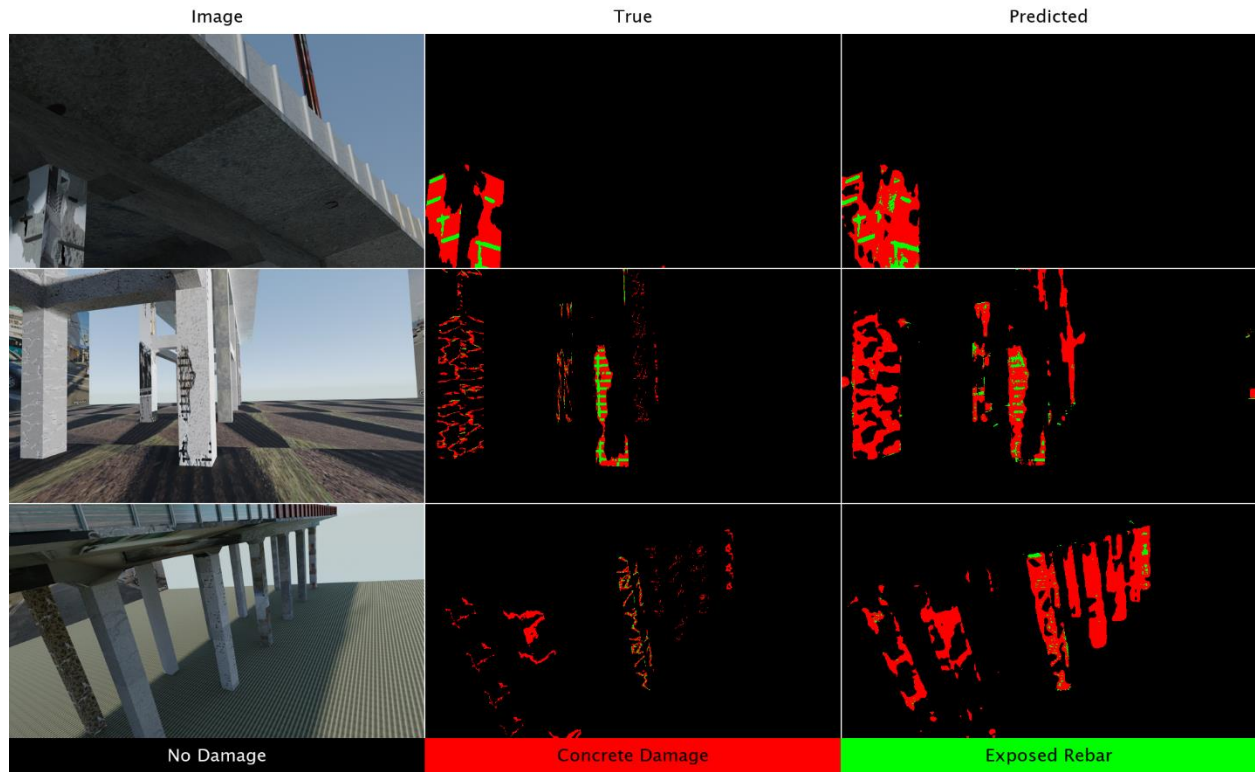


Figure 60 Example “undesirable” damage recognition results (regular images of Tokaido dataset)

To alleviate the challenge, this research applies pre-processing to the Tokaido dataset and discards far damage labels. The steps of the pre-processing are as follows:

- Set a threshold t_d that specifies minimum number of pixels per centimeter (pixel/cm) required to perform damage recognition reasonably.
- Identify regions of reliable damage recognition using ground truth depth maps. The region of reliable damage recognition is defined as an image region where the resolution is more than t_d pixel/cm.

- Discard all ground truth damage labels that are not in the regions of reliable damage recognition.
- Discard all images that do not contain damage labels in the regions of reliable damage recognition.

After applying the threshold, images in the dataset are split into training set (175 environments) and testing set (25 environments), as discussed in Chapter 6. Because the thresholding reduces the number of training images significantly, both regular images and close-up images (defined in Chapter 3) are used in this chapter. The numbers of training, validation, and testing images for different values of the threshold t_d is presented in Table 21. The numbers are roughly proportional to the volume defined by the threshold: compared to the case with $t_d = 1.5$, the volume of the regions of reliable damage recognition for the case $t_d = 2.0$ is approximately $\left(\frac{1.5}{2.0}\right)^3 = 42.2\%$. This value matches the ratio of the numbers of images between those two cases (2,283/5,233=43.6%).

Table 21 Number of images in the Tokaido dataset after pre-processing

t_d [pixel/cm]	Training & Validation	Testing	Total
None	13,987	1,985	15,972
1.5	4,599	634	5,233
2.0	2,010	273	2,283

The pre-processing steps reduce the number of images significantly, causing overfitting issues. For example, results of training the FCN58 network using regular and close-up images with $t_d = 2.0$ are shown in Figure 61 for both training and testing data. The figures clearly indicate that the network is overfitted to the training data, and further improvement of accuracy cannot be expected by increasing the training iterations (similar tendency has been observed for the case $t_d = 1.5$). To supplement the reduced amount of data, this research uses pure texture images

(2,700 images for training and validation, 300 images for testing) in combination with the regular and close-up images. Similar to the structural component recognition and depth map estimation, the validation set is created by selecting 300 images randomly.

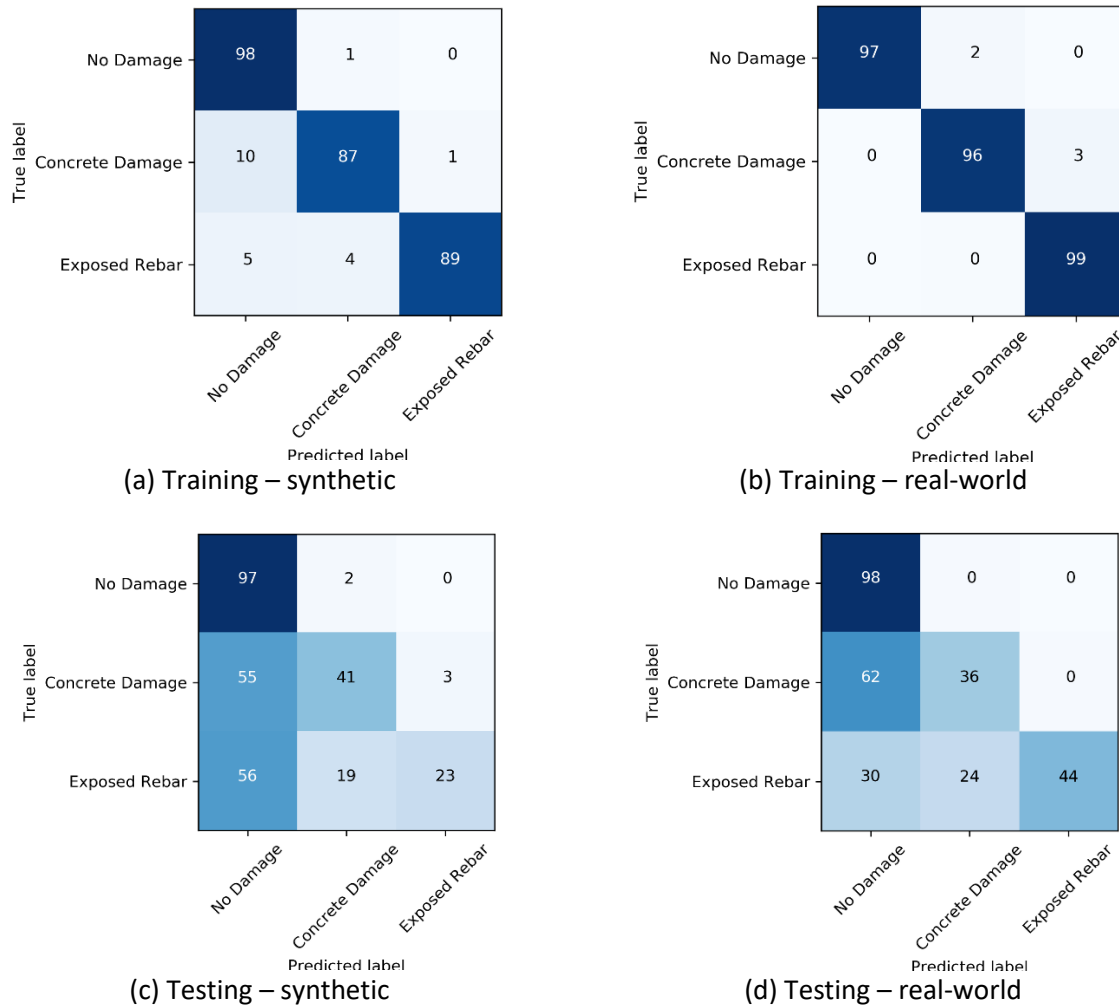


Figure 61 Confusion matrices of example *preliminary* analysis results (regular and close-up images, td=2.0). Overfitting is observed.

8.1.3 Refinement and interpretation of damage recognition results

After applying the FCN for damage recognition, the results are refined and interpreted to the form consistent with the current practice of visual inspection by human experts. The visual inspection results differ from the raw semantic segmentation results in two aspects mainly:

- Human inspectors can assess how much they are sure of their damage recognition naturally. When the damage-like patterns exist on far surfaces, the inspectors simply walk towards the components to take a closer look. The semantic segmentation algorithm can address the problem partially by providing softmax probability. However, softmax probability combines all sorts of uncertainty into a single variable. The network may not be very sure about the damage because either the shape and color of the damage were not learned sufficiently during training, or the damage is too far to make an accurate assessment. We need an explicit mechanism to evaluate whether the suspected damage is in the regions of reliable damage recognition or not.
- In addition to the damage itself, where it occurs is important for structural condition assessment. For example, damage on the column surface has more significance than the same size and shape of damage on the surface of nonstructural components. In an extreme case, damage patterns that are not on the surface of the target structure need to be neglected.

This research applies steps depicted in Figure 62 to refine and interpret the damage recognition results. First, regions of reliable damage recognition (RRDRs) are estimated using the depth map estimation results (discussed in Chapter 6). All damage labels outside of the RRDRs are discarded, because the FCN for damage recognition has not been trained for such cases. Then, structural component recognition results (also discussed in Chapter 6) is applied to identify the structural component types where the damage has occurred. At the same time, damage labels in non-bridge regions of images are discarded to maintain the consistency of the results to actual visual inspection process. The output of the refinement and interpretation unit is masked images

that contains information about damage, where the damage has occurred, and whether the structural assessment is reliable or not.

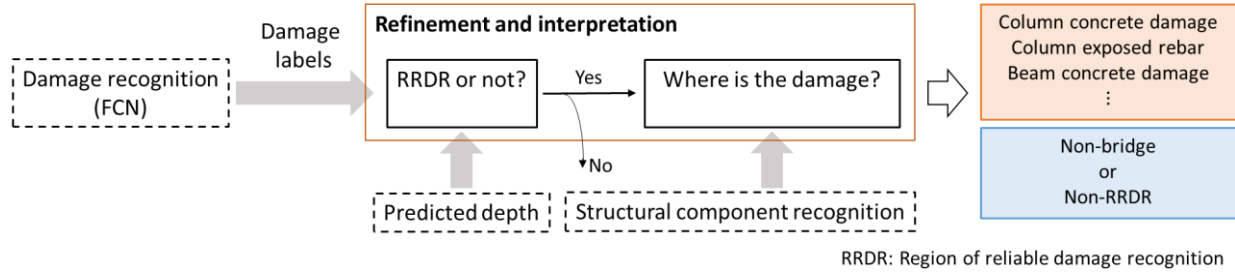


Figure 62 Steps to refine and interpret damage recognition results

8.2 Vision-based post-earthquake structural assessment results

8.2.1 Dataset considerations

The pre-processed Tokaido dataset (Table 21) is split into training, validation, and testing sets as specified in Table 22. In addition to the synthetic Tokaido dataset, small number of real-world images are used to improve the generalizability of the networks. The real-world images are collected from two sources: (i) images of general RC damage downloaded from the internet, and (ii) images of damaged RC viaducts collected by Professor Yoshikazu Takahashi at the Kyoto University after the 2011 Tohoku Earthquake [227]. The viaducts in the set (ii) have the structures similar to the ones with the standard design. The collected images are labelled manually using the Labelbox. In addition to the data thus created, the training set includes 101 real-world images from the structural component recognition task with “No Damage” label assigned to everywhere (“+101” in Table 22). When the image size is not consistent with those of the Tokaido dataset, the image is rescaled, and multiple sub-images are cropped to cover the entire region of the original image (numbers in parentheses in Table 22). Note again that the numbers of real-world images are not enough to train deep neural networks (example “unsatisfactory” results are shown in Figure 38).

Table 22 Dataset specifications for the training, validation, and testing of FCNs, numbers in parentheses indicates the numbers of cropped images

t_d [pixel/cm]	Training		Validation	Testing	
	Real-world	Synthetic	Synthetic	Real-world	Synthetic
1.5	61+101 (262)	6,999	300	60 (165)	934
2.0	61+101 (262)	4,410	300	60 (165)	573

The FCN58 network is first trained with the learning rates of 0.001 for 200 epochs, and then for 50 epochs with the learning rate of 0.0001 and 10 epochs with the learning rate of 0.00001. Each epoch contains 907 iterations for $t_d = 1.5$, and 584 iterations for $t_d = 2.0$. The accuracy and loss function on the validation set are evaluated after every epoch to monitor the training progress. Other detail of the training procedure follows the discussions in Chapter 6.

8.2.2 Damage recognition results

During the evaluation, argmax of the softmax probability is computed without re-weighting, because the approach produces better results of detecting damage that has rare probability of occurrence (“Concrete Damage” and “Exposed Rebar” classes constitute 3.9% and 0.4% of the total number of pixels, respectively). The performance of the trained network on the testing set is shown in Figure 63 using confusion matrices. Compared to the “undesirable” results discussed in section 8.1.2, the performance for the synthetic data has been improved significantly, approaching the performance presented for the training set. The performance for different pre-processing threshold values is similar, with slightly higher value of the matrix diagonal for concrete damage when $t_d = 1.5$. For the case $t_d = 1.5$, performance evaluation criteria, Precision, Recall, and IoU, are presented in Table 23. The recall values for the synthetic testing images show that the Tokaido dataset is effective for training networks that can detect concrete damage and exposed rebar, even when the input images are complex, containing multiple structural

components in general scenes. On the other hand, precision and IoU values are relatively low, indicating that accurate pixel-level localization of the structural damage is challenging: concrete damage and exposed rebar are often thin and subtle in images, as discussed during dataset considerations.

True label	No Damage	97	2	0
	Concrete Damage	9	85	5
	Exposed Rebar	2	16	80
		No Damage	Concrete Damage	Exposed Rebar
		Predicted label		

(a) Synthetic images ($t_d = 1.5$)

True label	No Damage	95	3	0
	Concrete Damage	32	61	6
	Exposed Rebar	29	30	40
		No Damage	Concrete Damage	Exposed Rebar
		Predicted label		

(b) Real-world images ($t_d = 1.5$)

True label	No Damage	97	2	0
	Concrete Damage	11	82	5
	Exposed Rebar	3	15	80
		No Damage	Concrete Damage	Exposed Rebar
		Predicted label		

(c) Synthetic images ($t_d = 2.0$)

True label	No Damage	96	3	0
	Concrete Damage	32	62	5
	Exposed Rebar	27	33	39
		No Damage	Concrete Damage	Exposed Rebar
		Predicted label		

(d) Real-world images ($t_d = 2.0$)

Figure 63 Confusion matrices (Damage recognition with different pre-processing thresholds)

The network performance for the real-world images, example damage recognition results, and the corresponding image-wise performance criteria are shown in Table 23, Figure 64, and Table 24, respectively. The first three images in Figure 64 demonstrate the effectiveness of the Tokaido dataset and the trained network to detect and localize concrete damage and exposed rebar in various sizes, shapes, and contexts. The fourth image show both potential and challenge. While the network returns reasonable results partially, the overall accuracy of damage recognition for this image is not as high as other images. Two types of errors can be observed herein: (i) deformed rebar and missing core concrete are not labeled accurately (right column), and (ii) far damage is not recognized at all (left column). The first type of error is caused because synthetic dataset does not contain such images, and the number of real-world images that contain such patterns is not enough to perform reliable recognition (only a few images of such damage are included in the 61 real-world images used for training). The second type of error is caused by the pre-processing of the synthetic dataset: far damage labels were removed before using the data to train FCNs. This type of error leads to the reduced recall values in Table 23.

The first type of error can be reduced by either increasing real-world training images or modeling additional phenomena, such as deformed rebar and core concrete removal, in the synthetic environment. Algorithms that implement attention mechanisms can also be investigated to improve the accuracy for smaller or thinner damage, as well as near and large damage (e.g. multi-scale attention masks can be aggregated by hierarchical approach, as proposed in [228]). The second type of error can be controlled by combining the damage recognition results with the depth estimation results, which is discussed in the next sections.

Table 23 Network performance on testing set [%] (Damage recognition).

	Synthetic			Real-world		
	Precision	Recall	IoU	Precision	Recall	IoU
No Damage	99.7	97.7	97.4	93.9	95.9	90.3
Concrete Damage	52.5	85.0	48.1	66.8	61.1	46.9
Exposed Rebar	47.3	81.0	42.5	52.5	40.1	29.4
Mean	66.5	87.9	62.7	71.1	65.7	55.5

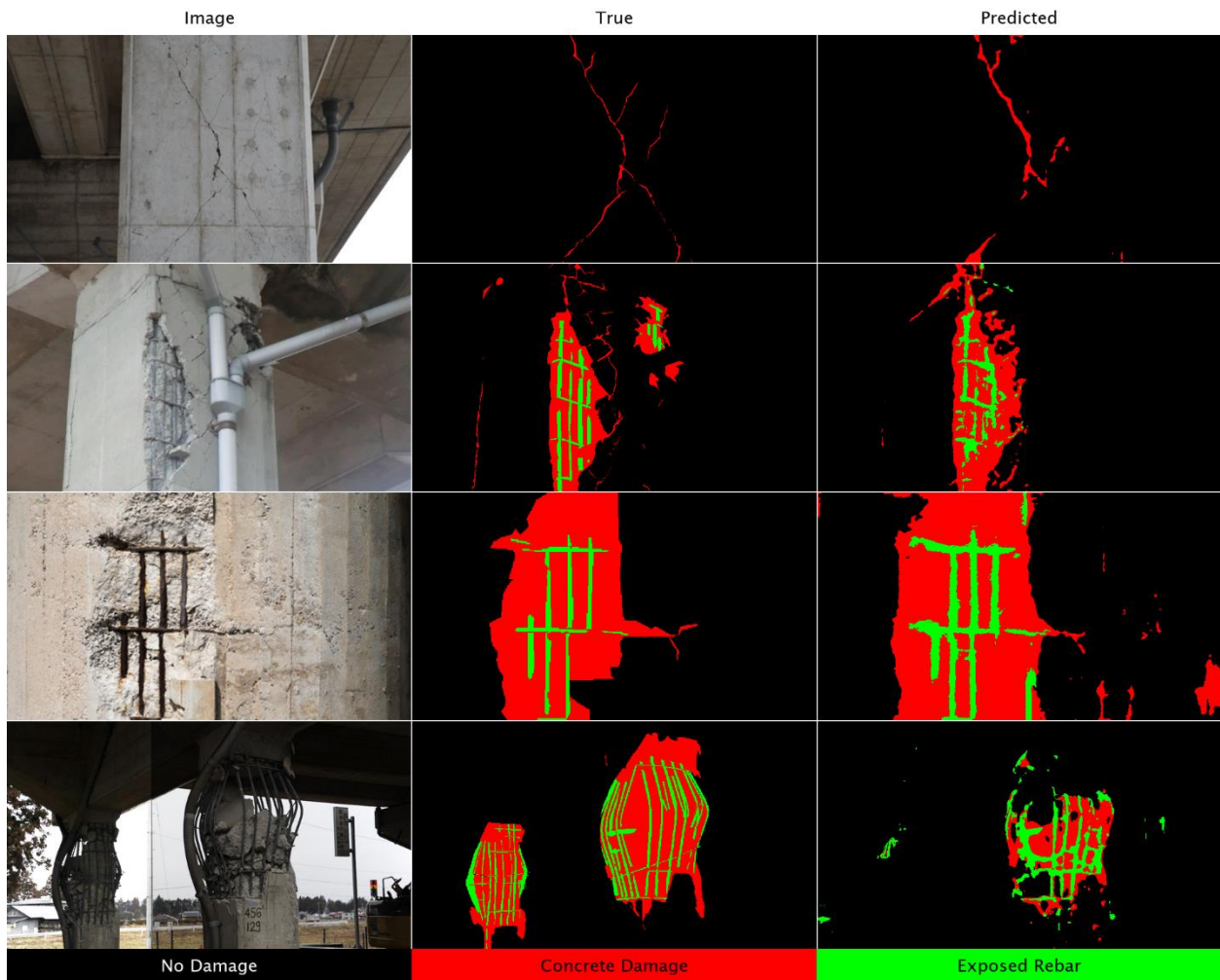


Figure 64 Example damage recognition results (real-world images from [227])

Table 24 Image-wise performance criteria of example damage recognition results.

Image	Concrete Damage			Exposed Rebar		
	PR	REC	IoU	PR	REC	IoU
1	27.7	58.3	23.1	NA	NA	NA
2	55.6	64.6	42.6	48.7	50.1	32.8
3	71.8	88.3	65.6	47.9	96.6	47.1
4	76.0	24.2	22.5	43.3	31.4	22.3

PR: Precision, REC: Recall, IoU: Intersection over Union.

8.2.3 Vision-based structural assessment results

The results of structural component recognition, damage recognition, and depth map estimation can be combined to perform automated structural condition assessment of the high-speed railway viaducts. This process discussed in 8.1.3 is demonstrated in Figure 65 for synthetic testing images. The example results show the effectiveness of the proposed approach to extract information relevant to the structural conditions of railway viaducts.

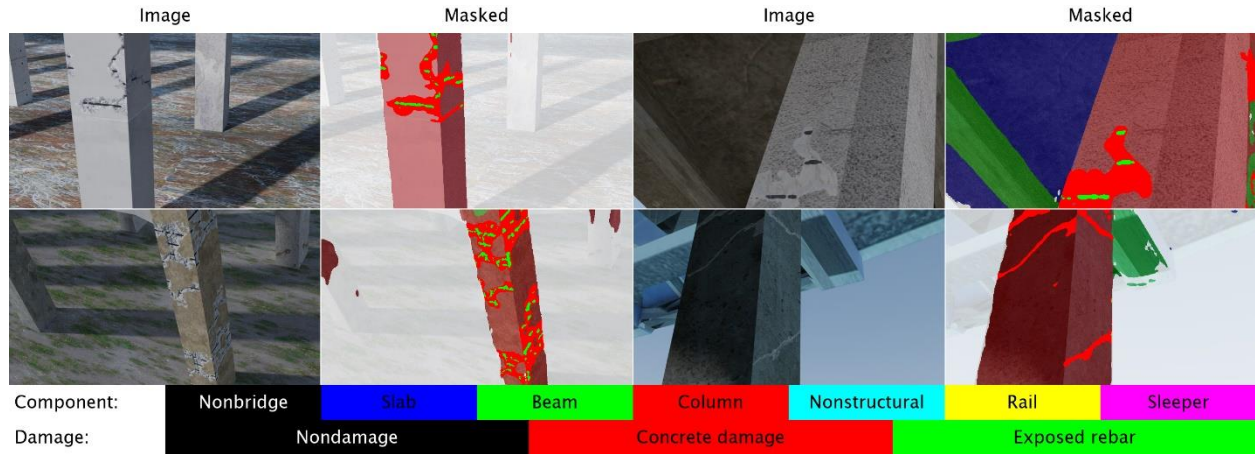


Figure 65 Example structural condition assessment results. Far or non-bridge regions are grayed out.

Performance improvement for damage recognition is also expected by specifying regions of reliable recognition derived using the predicted depth map. The performance criteria presented in Table 23 is evaluated again in Table 25, using labels in the regions of reliable damage recognition (RRDR) only. The RRDRs are defined to have the predicted depth between 0.5m and

15m (camera focal lengths are not available, and therefore the threshold cannot be applied in the form of pixel/cm). Note that predicted depth of real-world images is not accurate, because the network is not trained for the structures contained in the images. The predicted depth map indicates whether the network “thinks” that each part of image is close to the camera or not. With this post-processing, the recall values improve significantly, approaching the levels presented in Table 23 for synthetic images.

Table 25 Network performance on testing set [%] (Damage recognition, after post-processing).

	Real-world		
	Precision	Recall	IoU
No Damage	99.7	95.9	95.6
Concrete Damage	39.8	84.4	37.0
Exposed Rebar	36.7	51.2	27.2
Mean	58.7	77.2	53.3

To further explain the results presented in Table 25, image-wise IoUs for concrete damage are plotted against pixel ratio (number of concrete damage pixels divided by total number of pixels in an image) in Figure 66. By considering RRDRs only, damage recognition results with relatively low accuracy for each range of the pixel ratio can be removed automatically. The apparent decrease of IoU values after post-processing observed in Table 25 is caused by larger proportion of images with small concrete damage pixel ratios, where accurate recognition is challenging inherently.

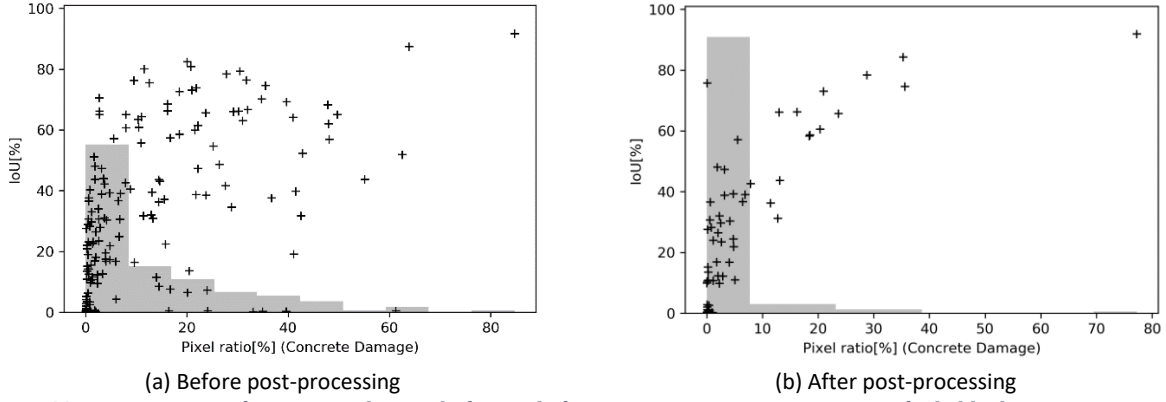


Figure 66 Image-wise IoU for concrete damage before and after post-processing using regions of reliable damage recognition. Histogram shows proportion of testing images with the given range of concrete damage pixel ratio.

8.3 Summary and conclusions

This chapter discussed an approach to post-process images of RC railway viaducts to perform rapid post-earthquake structural inspection automatically. The approach is based on the deep fully convolutional network for semantic segmentation (discussed in Chapter 6), and integrates the discussions in previous chapters to obtain results that are consistent with the current practice of human-based visual inspection. The system discussed in this chapter converts input image of the target structure to a masked image that contains information of structural damage, structural component types where the damage is observed, and the image regions where the damage recognition is reliable (regions of reliable damage recognition, RRDRs). The results presented in this section are promising: The trained damage recognition network can detect concrete damage (Recall: 85.0% for synthetic images, 61.1% for real-world images, and 84.4% for real-world images with post-processing), as well as exposed rebar (Recall: 81.0% for synthetic images, 40.1% for real-world images, and 51.2% for real-world images with post-processing). The example results showed that the system discussed in this chapter can detect, localize, and interpret RC damage of various sizes, shapes, appearances, and distances.

CHAPTER 9: CONCLUSIONS AND FUTURE DIRECTIONS

This research investigated a framework for automated vision-based post-earthquake inspection of RC railway bridges that reduces the time needed for the railway lines to return to the operation. Currently, after an earthquake, coarse preliminary structural assessment is performed based on the information immediately available (e.g. ground acceleration data). Then, human-based visual inspection is performed to obtain detailed information about the structures with suspected damage. This procedure is often time-consuming and labor-intensive, because the preliminary assessment tends to make conservative decisions, putting significant amount of workload for the subsequent detailed inspection. For example, after the 2011 Tohoku Earthquake, the challenge of the post-earthquake inspection lead to 16-18 hours of operational shutdown of railways in Tokyo metropolitan area, Japan, causing more than 5 million people who were unable to go home. To address this problem, the framework investigated in this research aims at obtaining improved structural assessment during the initial response after earthquakes to reduce the workload of the human-based detailed inspections.

This framework investigated in this research has four steps: (i) a UAV flies to the location near a viaduct using GPS data (GPS-based navigation), (ii) the UAV navigates to the critical components of the viaduct, based on the recognition of critical structural components and their locations (vision-based component localization and mapping), (iii) images of the target structure are collected and evaluated for the safety of the structure (vision-based structural inspection), and (iv) a synthetic environment for developing, implementing, and testing methods for automated bridge inspection. This research particularly focused on steps (ii)-(iv), because the first step, GPS-based navigation, has been well-established to date.

In Chapter 2, existing research about related topics are reviewed. The review is organized into four topics: vision-based inspection of civil engineering structures, computer vision and machine learning, autonomous navigation techniques, automated vision data collection for structural inspection using UAVs, and automated vision data evaluation for structural inspection. The review has clarified that, while potentially useful techniques are available in either civil engineering or related fields, those techniques need to be re-formulated and/or combined appropriately with the customized dataset from the application context to produce the desired results.

The investigation of the automated vision-based post-earthquake structural inspection starts with the development of synthetic environments in Chapter 3. The purpose of the synthetic environments is to enable experiment of different algorithms, efficient training data generation, and ground truth measurement that helps the evaluation of algorithms. This research developed two types of synthetic environments to support the investigation of automated post-earthquake inspection of railway bridges: (i) a game environment to simulate UAV-based video data collection, and (ii) graphics models of RC railway viaducts of the Tokaido Shinkansen (one of Japanese high-speed railway lines operated and maintained by the Central Japan Railway Company). The first environment can produce images of RC railway bridges and the associated ground truth structural component labels (pixel-wise annotation) during random UAV flights. The second environment is the extension of the first environment in the sense that the viaduct modeling process is programmed and therefore random instances can be generated efficiently. Besides structural component labels, the environment can produce ground truth depth maps and the pixel-wise annotations of structural damage generated randomly. The developed environments support the subsequent discussions of this research strongly.

To begin with the discussions of vision-based component localization and mapping, automated vision-based bridge component recognition using real-world image data is discussed in Chapter 4. Two different semantic segmentation algorithms (Fully Convolutional Networks, or FCNs and SegNet), as well as three different configurations of semantic segmentation networks (naïve, sequential, and parallel configurations) are investigated to realize a network that performs robustly even for complex urban scenes. The FCN with 45 convolutional layers (*FCN45*) and the SegNets extended from the FCN45 architecture (*SegNet45*, *SegNet45-S*) were first compared using the scene classification dataset. The *FCN45* and the *SegNet45* achieved high accuracy (including 70% mean intersection over union and more than 85% frequency-weighted intersection over union) and outperformed the *SegNet45-S*.

Then, the bridge component classifier was trained using three configurations to impose high-level scene consistency, i.e., the naïve configuration, the parallel configuration, and the sequential configuration. The accuracy criteria such as mean precision, mean/global pixel accuracy, mean/frequency-weighted intersection over union, and mean/global false positive rate showed equal level of performance for the three configurations (differences of these criteria for different configurations are limited within 1%). However, when the input image is from a complex scene that does not necessarily include bridges, the performance of the bridge component classifier was improved by explicitly imposing the scene-level consistencies, showing fewer false-positive detections. The most significant improvement was observed for the sequential configuration, which showed false positive rates less than 1% for all non-bridge scene categories.

Chapter 5 extends the work presented in Chapter 4 to video data (image sequences) obtained during UAV flights. To get consistent results throughout the UAV flight, recurrent neural networks are applied, and the information from the previous frames is utilized as hidden states

during subsequent frames. By putting the recurrent units only after the lowest resolution prediction layer and training the recurrent unit independently, the RNN parameters were learned in a reasonable amount of time. The architecture with recurrent units outperformed the FCN both quantitatively (pixel-wise accuracy) and qualitatively (example estimated label maps). Moreover, the ConvLSTM units performed significantly better than the simple RNN when the FCN failed to recognize the bridge components. In the future, computation time will need to be thoroughly evaluated to apply the method in near real-time. Large amount of video data generated effectively in the synthetic environment is used to train and test the proposed network with the recurrent neural networks.

Chapter 6 improves the accuracy of vision-based bridge component recognition discussed in Chapter 4 significantly, and further extends the capability of the network to the localization of target structural component. The semantic segmentation algorithm, fully convolutional network (FCN), is used to perform structural component recognition. The localization of the target structural components is achieved by estimating the depth map of an input image using an FCN-based algorithm. Tokaido dataset, a large-scale synthetic dataset discussed in section 3.3, is used to augment dataset necessary for training and testing the networks. The trained networks are evaluated using real-world images and synthetic images. For the structural component recognition, the trained network achieved 86.2% of IoU for synthetic testing images, as well as 74.0% of IoU for the real-world testing images. The results for the depth map estimation were also promising, with 1.32m of mean absolute error, 2.09m of root mean squared error, and 10.7% of the absolute relative distance in the range [0.5m, 30m]. The Tokaido dataset produced in this research has shown to facilitate the investigation of various state-of-the-art algorithms proposed for general semantic segmentation and depth recognition problems, so that the structural engineering

community can be fully benefitted from the recent advance of the computer vision and data science field.

Chapter 7 investigates vision-based autonomous UAV navigation strategies for rapid post-earthquake inspection of reinforced concrete railway viaducts. The strategies are designed to mimic the logic followed by human inspectors. To perform the inspection, an inspector obtains initial understanding of the target viaduct by walking nearby. In this step, the inspector does not need to understand the entire structure. Instead, the inspector needs to know the rough idea of the target structure, as well as the clear idea of where the first few critical structural components are located. After the initiation of the task, the inspector's understanding of the target structure is improved incrementally and continuously as s/he experiences more viewpoints. The inspector does not need to have access to the complete design drawing of the structure to perform this task. Instead, s/he needs to have the prior knowledge about the typical geometric or visual contextual patterns of the structure (e.g. regular grid patterns of the column locations, typical dimensions of the columns etc.). Toward this goal, this chapter has developed an on-line data processing approach that first parses the sparse point cloud data using frame-wise semantic segmentation results, and identifies the approximate scale using the frame-wise depth estimation results. Then, rectangles are fitted to the points with the "Column" label, from which columns are detected by finding near-parallel or near-perpendicular pairs. In addition to the columns thus detected, the strategy tries to find the grid-line pattern of the column locations, based on which the detection results are refined, and information about missing columns are inferred. With this approach, the UAV can initiate the inspection mission before obtaining complete information of the target structure. The information (map) can be improved progressively as the UAV experiences more views during inspection. The capability of the developed strategy in detecting columns robustly and reliably was evaluated using

the synthetic data. The results showed that all eight columns of the target viaducts are detected in all 100 trials with centimeter-level accuracy. Finally, the entire strategy was demonstrated in the synthetic environment, showing the significant potential of collecting high-quality images for post-earthquake structural inspection efficiently and quickly.

Chapter 8 discusses an approach to post-process the collected images of RC railway viaducts to perform rapid post-earthquake structural inspection automatically. The approach is based on the deep fully convolutional network for semantic segmentation (discussed in Chapter 6), and integrates the discussions in previous chapters to obtain results that are consistent with the current practice of human-based visual inspection. The developed data post-processing approach converts input image of the target structure to a masked image that contains information of structural damage, structural component types where the damage is observed, and the image regions where the damage recognition is reliable (regions of reliable damage recognition, RRDRs). The results presented in this chapter were promising: the trained damage recognition network can detect concrete damage (Recall: 85.0% for synthetic images, 61.1% for real-world images, and 84.4% for real-world images with post-processing), as well as exposed rebar (Recall: 81.0% for synthetic images, 40.1% for real-world images, and 51.2% for real-world images with post-processing). The example results showed that the system discussed in this chapter can detect, localize, and interpret RC damage of various sizes, shapes, appearances, and distances. This chapter completes the prototype system for the automated rapid post-earthquake inspection using vision-based techniques that is envisioned in this research.

9.1 Future Directions

The future work is organized into four areas: (i) improving robustness to accommodate the uncertainty encountered in the field environment, (ii) incorporating physics to the synthetic environment, (iii) extension to the city-scale inspection activities that deals with the collections of structures, instead of a single structure, and (iv) real-time and on-board implementation.

9.1.1 Improving robustness

While the synthetic environments are effective tools to develop, test, and optimize data processing algorithms, those environments do not model every detail of the field environment (most notably, obstacles). To bring the developed system into the field environment, the system's robustness needs to be improved by experimentally identify uncertainties that may cause unexpected or undesirable behavior. Once those uncertainties are identified, the methods that prevents such outcome will be implemented, and at the same time, those uncertainties are implemented in the synthetic environment to enable realistic simulations in the future. To improve the robustness of the visual recognition (depth estimation in particular), additional real-world data will be collected and incorporated into the system.

Besides the elaboration of the data post-processing methods, sensor input can be enriched to improve the robustness. Currently, monocular camera is used to determine UAV navigation paths and perform structural assessment. The quality of the inspection results are expected to be installed by developing a framework to leverage different types of sensors used in each application scenario, such as cameras, LiDAR data, infrared (IR) camera, and pre-installed contact sensors (accelerometers, strain gages etc.). These extensions will improve the robustness of the proposed framework and further open up a way to apply the autonomous inspection framework to routine inspection that typically requires higher levels of detail.

9.1.2 Incorporating physics to the synthetic environment

Generation of damage scenarios in the synthetic environment, which is purely random currently, can be improved by incorporating physics, typically by finite element analysis. Hoskere et al. [164] recently proposed a photo-realistic synthetic model, termed physics-based graphics model (PBGM) whose damage conditions are realized based on the information about finite element results (damage hotspots). The modeling approach showed broad range of potential applications, including the vision-based structural inspection and vision-based displacement and strain estimation (Narazaki et al. [166], [167]). The PBGM is expected to improve the evaluation and demonstration process of the developed system by generating random but realistic damage scenarios for each assumed earthquake (Figure 67).

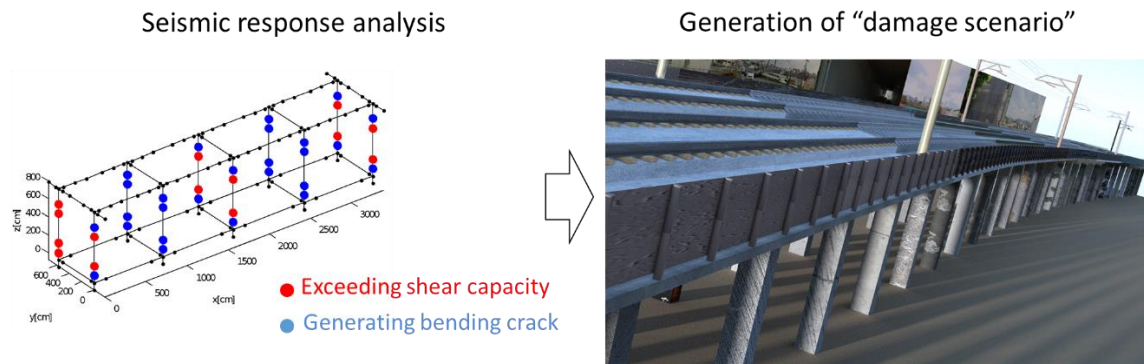


Figure 67 Incorporating physics to the synthetic environment to obtain realistic damage scenarios.

9.1.3 City-scale inspection

This research mainly investigated the automated inspection of a single viaduct using a single UAV. The scope will be expanded in the future to the inspection of the collection of structures spreading across the entire affected area using multiple (or a “swarm” of) UAVs, that is, the city-scale inspection. Based on the current locations of the UAVs. In this scenario, UAVs communicate and coordinate to finish the inspection task as efficiently as possible.

9.1.4 Real-time and on-board implementation.

This research focuses on developing a prototype system for the automated post-earthquake structural inspection, and the investigation of the issues related to the field implementation is preliminary. The hardware used for this research is desktop computer with powerful GPUs, which may not be the desirable setting for the field implementation of the proposed framework. In particular, the prototype UAV navigation system uses sparse point cloud data obtained by applying the structure from motion (SfM) algorithm batch-wise, which would pose a bottleneck when the system is implemented in UAVs performing missions in the field. The developed system will be optimized regarding the computational aspects in the future.

An approach to investigate the real-time and on-board implementation is shown in Figure 68. In this example, a UAV flies randomly in a Unity synthetic environment, and the images collected by the UAV camera is streamed to the Robotic Operating System (ROS) component in real-time, where the simultaneous localization and mapping algorithm (LSD-SLAM) is applied in a way that actual robots (UAVs) process the data. The two components, Unity and ROS, are interfaced using the ROS# library, which can transfer data (messages) between the two components. Recent advances in hardware supports the investigation of real-time and on-board implementation of the developed system (e.g. Jetson Nano small AI computer developed by NVIDIA [229], see Figure 69). The optimal combination of software and hardware will be investigated in the synthetic environment (ROS - ROS# - Unity) first, and then in the field environment.

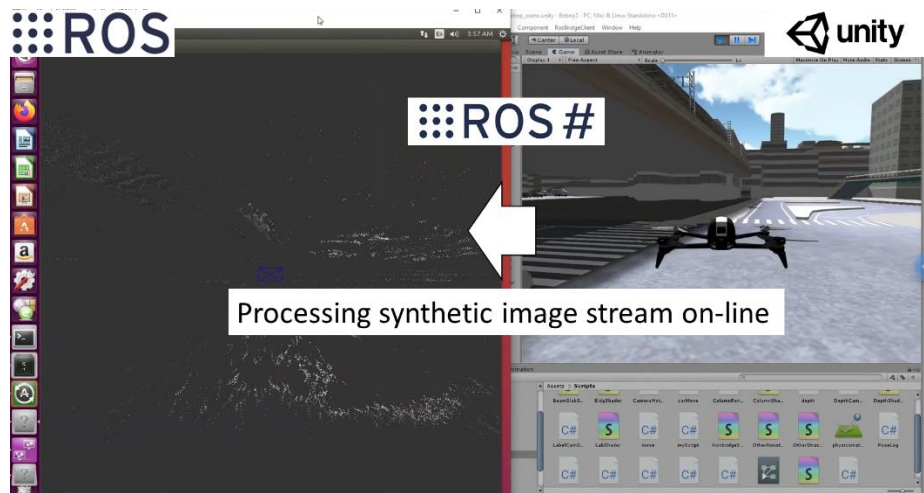


Figure 68 Simulation of real-time and on-board implementation using ROS, Unity, and ROS# (an example of applying LSD-SLAM algorithm to the image stream).



Figure 69 Jetson Nano small AI computer developed by NVIDIA[229]

REFERENCES

- [1] “交通政策基本法に基づく政策展開：平成28（2016）年版交通政策白書データ集 - 国土交通省.” [Online]. Available: https://www.mlit.go.jp/sogoseisaku/transport_policy/sosei_transport_policy_fr1_000015.html. [Accessed: 08-Jul-2020].
- [2] “鉄道：鉄道統計年報 [平成26年度] - 国土交通省.” [Online]. Available: https://www.mlit.go.jp/tetudo/tetudo_tk2_000038.html. [Accessed: 08-Jul-2020].
- [3] “MLIT report about the post-earthquake response of railways [in Japanese],” MLIT (Ministry of Land, Infrastructure, Transport and Tourism), 2012.
- [4] 高津俊司 and 土井充, “日本の高速鉄道--その軌跡と今後の展望,” 鉄道ピクトリアル, vol. 58, no. 2, pp. 142–153, 2008.
- [5] “気象庁 Japan Meteorological Agency.” [Online]. Available: <http://www.jma.go.jp/jma/index.html>. [Accessed: 08-Jul-2020].
- [6] “11th report of the Shizuoka Earthquake (in Japanese),” Oct. 2011.
- [7] “The 1995 Southern Hyogo Prefecture Earthquake (in Japanese),” 1997.
- [8] “阪神・淡路大震災の経験に学ぶ--震災時における社会基盤利用のあり方について,” Ministry of Land, Infrastructure, Transport and Tourism, Japan. [Online]. Available: <https://www.kkr.mlit.go.jp/plan/daishinsai/>. [Accessed: 08-Jul-2020].
- [9] “防災情報のページ - 内閣府,” Cabinet Office, Government of Japan. [Online]. Available: <http://www.bousai.go.jp/>. [Accessed: 08-Jul-2020].
- [10] “The Asahi Shimbun evening newspaper,” 11-Aug-2009.
- [11] “事故などの場合の取り扱い：JR東日本,” JR-EAST. [Online]. Available: <https://www.jreast.co.jp/kippu/24.html>. [Accessed: 08-Jul-2020].
- [12] “大規模地震発生時における首都圏鉄道の運転再開のあり方に関する協議会報告書 [A report of the committee on the resumption of train operation in Tokyo area after large-scale earthquakes, in Japanese].” 2012.
- [13] “UCK Inc., Japan.” [Online]. Available: <https://www.uck-inc.com/en/uck-blog/20160311444>. [Accessed: 29-Jan-2018].
- [14] T. TAKAHAMA, S. MIDORIKAWA, and M. OHORI, “A Study on Railroad Traffic Interruption After an Earthquake Based on the Questionnaire Survey to Railway Companies,” *J. Soc. Saf. Sci.*, vol. 8, no. 8, pp. 253–258, Nov. 2006.
- [15] H. Suzuki and M. Shimamura, “地震時運転規制方法の研究 [A Study on Railroad Traffic Interruption After an Earthquake Based on the Questionnaire Survey to Railway Companies, in Japanese],” *JR EAST Tech. Rev.*, pp. 53–60, 2006.
- [16] “Unity.”
- [17] “Blender.” [Online]. Available: <https://www.blender.org/>.
- [18] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015, vol. 39, no. 4, pp. 640–651.
- [19] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, 2017.

- [20] Christopher Rojahn *et al.*, *ATC-20-1 Field Manual: Postearthquake Safety Evaluation of Buildings*. 2005.
- [21] 被災建築物応急危険度判定マニュアル [An emergency safety evaluation guideline, in Japanese]. 1998.
- [22] J. S. O’connor, “Post-Earthquake Bridge Inspection Guidelines,” 2010.
- [23] J. A. Ramirez, R. J. Frosch, M. A. Sozen, and A. M. Turk, “HANDBOOK FOR THE POST-EARTHQUAKE SAFETY EVALUATION OF BRIDGES AND ROADS,” 2000.
- [24] 2015年改訂版 再使用の可能性を判定し、復旧するための震災建築物の被災度区分判定基準および復旧 技術指針 [Guidelines for detailed post-earthquake safety evaluation of buildings, ver. 2015, in Japanese]. 日本建築防災協会, 2016.
- [25] T. 7 W. Author, J. E. Ryan, P. E. Mann, Z. M. Chill, B. T. Ott, and M. Baker, “Baker Project Manager,” 2002.
- [26] Y. Tanimura, M. Sogabe, M. Okamoto, and H. Ueda, “Outline of the Model Code for Maintenance of Concrete Railway Structures,” *Concr. J.*, vol. 46, no. 2, pp. 18–23, Feb. 2008.
- [27] J. Zink and B. Lovelace, “Unmanned Aerial Vehicle Bridge Inspection Demonstration Project,” Project report for the Minnesota Department of Transportation, 2015.
- [28] J. Wells and B. Lovelace, “Unmanned Aircraft System Bridge Inspection Demonstration Project Phase II Final Report,” A project for the Minnesota Department of Transportation, 2017.
- [29] L. D. Otero, N. Gagliardo, D. Dalli, W. H. Huang, and P. Cosentino, “Proof of Concept for Using Unmanned Aerial Vehicles for High Mast Pole and Bridge Inspection,” A project for the Florida Department of Transportation, 2015.
- [30] C. Brooks *et al.*, “Evaluating the Use of Unmanned Aerial Vehicles for Transportation Purposes,” A project for the Michigan Department of Transportation, 2015.
- [31] M. N. Gillins, D. T. Gillins, and C. Parrish, “Cost-Effective Bridge Safety Inspections Using Unmanned Aircraft Systems (UAS),” in *Geotechnical and Structural Engineering Congress 2016*, 2016, pp. 1931–1940.
- [32] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, 1986.
- [33] C. Harris, C. Harris, and M. Stephens, “A combined corner and edge detector,” *PROC. FOURTH ALVEY Vis. Conf.*, pp. 147–151, 1988.
- [34] D. H. Ballard, “Generalizing the Hough transform to detect arbitrary shapes,” *Pattern Recognit.*, vol. 13, no. 2, pp. 111–122, Jan. 1981.
- [35] R. Grompone Von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, “LSD: A fast line segment detector with a false detection control,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 722–732, Apr. 2010.
- [36] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005, vol. I, pp. 886–893.
- [37] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *Int. J. Comput. Vis.*, 2004.
- [38] K. Mikolajczyk and C. Schmid, “Scale & affine invariant interest point detectors,” *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, Oct. 2004.
- [39] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

- [40] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
- [41] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.
- [42] Y. Lecun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The handbook of brain theory and neural networks*, MIT Press, 1995.
- [43] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.
- [44] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.
- [45] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
- [46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [47] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning Transferable Architectures for Scalable Image Recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 8697–8710, Jul. 2017.
- [48] C. Farabet, C. Couprie, L. Najman, and Y. Lecun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.
- [49] D. P. Kingma and J. Ba, "A method for stochastic optimization," in *Proc. International Conference for Learning Representations*, 2015, pp. 1–15.
- [50] A. Krogh and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," in *NIPS*, 1991, vol. 4.
- [51] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *The 32nd International Conference on Machine Learning*, 2015.
- [52] D. Eigen and R. Fergus, "Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2650–2658.
- [53] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
- [54] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICRL)*, 2015, pp. 1–14.
- [55] "The PASCAL Visual Object Classes Homepage." [Online]. Available: <http://host.robots.ox.ac.uk/pascal/VOC/>. [Accessed: 26-Dec-2018].
- [56] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan. 2009.
- [57] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and Recognition using Structure from Motion Point Clouds," in *Computer Vision–ECCV 2008*, 2008, pp. 44–57.
- [58] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9351, pp.

- 234–241.
- [59] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, “UNet++: A Nested U-Net Architecture for Medical Image Segmentation,” Jul. 2018.
 - [60] H. Huang *et al.*, “UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation,” pp. 1055–1059, Apr. 2020.
 - [61] M. Cordts *et al.*, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-December, pp. 3213–3223, Apr. 2016.
 - [62] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3234–3243.
 - [63] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, “Indoor Semantic Segmentation using depth information,” *arXiv Prepr. arXiv1301.3572*, 2013.
 - [64] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, “Fast Scene Understanding for Autonomous Driving,” in *Deep Learning for Vehicle Perception (DLVP)*, 2017.
 - [65] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” Jun. 2014.
 - [66] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.
 - [67] R. Girshick, “Fast R-CNN,” *arXiv Prepr. arXiv1504.08083*, 2015.
 - [68] S. Ren, K. He, R. Gershick, J. Sun, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *Adv. Neural Inf. Process. Syst.*, vol. 39, no. 6, pp. 91–99, Jun. 2015.
 - [69] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
 - [70] J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” 2017.
 - [71] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” 2017.
 - [72] B. De Brabandere, D. Neven, and L. Van Gool, “Semantic Instance Segmentation with a Discriminative Loss Function,” *arXiv Prepr. arXiv1708.02551*, 2017.
 - [73] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun, “Monocular Object Instance Segmentation and Depth Ordering with CNNs,” *arXiv Prepr. arXiv1505.03159*, 2015.
 - [74] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, “Deeper depth prediction with fully convolutional residual networks,” in *Proceedings - 2016 4th International Conference on 3D Vision, 3DV 2016*, 2016, pp. 239–248.
 - [75] K. Tateno, F. Tombari, I. Laina, and N. Navab, “CNN-SLAM: Real-Time Dense Monocular SLAM With Learned Depth Prediction,” in *CVPR*, 2017, pp. 6243–6252.
 - [76] J. P. Werbos, “Backpropagation through time: what it does and how to do it,” *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
 - [77] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung, “Learning Video Object Segmentation from Static Images,” *Comput. Vis. Pattern Recognit.*, pp. 2663–2672, 2017.
 - [78] S. Hochreiter and J. “Urgen Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*,

- vol. 9, no. 8, pp. 1735–1780, 1997.
- [79] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, “On the Properties of Neural Machine Translation: Encoder-Decoder Approaches,” Sep. 2014.
 - [80] M. Siam, S. Valipour, M. Jagersand, and N. Ray, “Convolutional gated recurrent networks for video segmentation,” in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3090–3094.
 - [81] D. Nilsson and C. Sminchisescu, “Semantic Video Segmentation by Gated Recurrent Flow Propagation,” *arXiv Prepr. arXiv1612.08871*, 2016.
 - [82] X. Shi, Z. Chen, H. Wang, and D.-Y. Yeung, “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting,” *Adv. neural Inf. Process. Syst.*, pp. 802–810, 2015.
 - [83] S. Thrun *et al.*, “Stanley: The Robot that Won the DARPA Grand Challenge,” *J. F. Robot.*, vol. 23, no. 9, pp. 661–692, 2006.
 - [84] “Waymo.”
 - [85] “Autopilot | Tesla.” [Online]. Available: <https://www.tesla.com/autopilot>. [Accessed: 11-Jul-2020].
 - [86] “Elon Musk says full self-driving Tesla tech ‘very close’ - BBC News.” [Online]. Available: <https://www.bbc.com/news/technology-53349313>. [Accessed: 11-Jul-2020].
 - [87] A. Kelly, *Mobile robotics: Mathematics, models, and methods*. Cambridge University Press, 2013.
 - [88] S. J. Julier and J. K. Uhlmann, “New extension of the Kalman filter to nonlinear systems,” in *Signal Processing, Sensor Fusion, and Target Recognition VI*, 1997, vol. 3068, no. 28, p. 182.
 - [89] G. Chowdhary and R. Jategaonkar, “Aerodynamic parameter estimation from flight data applying extended and unscented Kalman filter,” *Aerosp. Sci. Technol.*, vol. 14, no. 2, pp. 106–117, Mar. 2010.
 - [90] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.
 - [91] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” Springer, Cham, 2014, pp. 834–849.
 - [92] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, “ORB-SLAM: A Versatile and Accurate Monocular SLAM System,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.
 - [93] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras,” *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2016.
 - [94] R. Valencia and J. Andrade-Cetto, *Mapping, Planning and Exploration with Pose SLAM*, vol. 119. Springer, 2018.
 - [95] C. Kerl, J. Sturm, and D. Cremers, “Dense visual SLAM for RGB-D cameras,” in *IEEE International Conference on Intelligent Robots and Systems*, 2013, pp. 2100–2106.
 - [96] T. Schöps, T. Sattler, and M. Pollefeys, “BAD SLAM: Bundle Adjusted Direct RGB-D SLAM,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 134–144.
 - [97] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright, and H. M. Tai, “Autonomous local path planning for a mobile robot using a genetic algorithm,” in *Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004*, 2004, vol. 2,

- pp. 1338–1345.
- [98] N. Buniyamin, W. Wan Ngah, N. Sariff, and Z. Mohamad, “A Simple Local Path Planning Algorithm for Autonomous Mobile Robots,” *Int. J. Syst. Appl. Eng. Dev.*, vol. 5, no. 2, 2011.
 - [99] R. A. Brooks and T. Lozano-Perez, “A Subdivision Algorithm in Configuration Space for Findpath with Rotation,” *IEEE Trans. Syst. Man Cybern.*, vol. SMC-15, no. 2, pp. 224–233, 1985.
 - [100] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 1985, pp. 500–505.
 - [101] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *Int. J. Rob. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
 - [102] L. E. Kavraki, P. Švestka, J. C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.
 - [103] S. M. Lavalle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” 1998.
 - [104] C. Goerzen, Z. Kong, and B. Mettler, “A survey of motion planning algorithms from the perspective of autonomous UAV guidance,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 57, no. 1–4, pp. 65–100, Jan. 2010.
 - [105] E. Frazzoli, M. A. Dahleh, and E. Feron, “Real-time motion planning for agile autonomous vehicles,” in *Proceedings of the American Control Conference*, 2001, vol. 1, pp. 43–49.
 - [106] S. M. LaValle and J. J. Kuffner, “Randomized Kinodynamic Planning,” *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.
 - [107] A. Bry, C. Richter, A. Bachrach, and N. Roy, “Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments,” *Int. J. Rob. Res.*, vol. 34, no. 7, pp. 969–1002, Jun. 2015.
 - [108] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, “Real-time motion planning with applications to autonomous urban driving,” *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, 2009.
 - [109] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
 - [110] M. Likhachev and D. Ferguson, “Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles,” *Int. J. Rob. Res.*, vol. 28, no. 8, pp. 933–945, Aug. 2009.
 - [111] B. Yamauchi, “Frontier-based approach for autonomous exploration,” in *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA*, 1997, pp. 146–151.
 - [112] A. Bachrach, R. He, and N. Roy, “Autonomous Flight in Unknown Indoor Environments,” *Int. J. Micro Air Veh.*, vol. 1, no. 4, pp. 217–228, Dec. 2009.
 - [113] F. Fraundorfer *et al.*, “Vision-based autonomous mapping and exploration using a quadrotor MAV,” in *IEEE International Conference on Intelligent Robots and Systems*, 2012, pp. 4557–4564.
 - [114] “Litchi for DJI Mavic / Phantom / Inspire / Spark.” [Online]. Available: <https://flylitchi.com/>. [Accessed: 12-Jul-2020].
 - [115] “FreeFlight 6 | Parrot Store Official.” [Online]. Available: <https://www.parrot.com/us/freeflight-6>. [Accessed: 12-Jul-2020].
 - [116] “近畿日本鉄道、キヤノンMJ、KDDIが4G LTE対応ドローンを活用した鉄道災害時の情報収集の実証実験を開始 | 2017年 | KDDI株式会社.” [Online]. Available:

- <http://news.kddi.com/kddi/corporate/newsrelease/2017/11/29/2813.html>. [Accessed: 12-Jul-2020].
- [117] J. Sun, B. Li, Y. Jiang, and C. Wen, "A Camera-Based Target Detection and Positioning UAV System for Search and Rescue (SAR) Purposes," *Sensors*, vol. 16, no. 11, p. 1778, Oct. 2016.
 - [118] "GPS.gov: GPS Accuracy." [Online]. Available: <https://www.gps.gov/systems/gps/performance/accuracy/>. [Accessed: 12-Jul-2020].
 - [119] M. N. Gillins, "Unmanned Aircraft Systems for Bridge Inspection : Testing and Developing End-to-End Operational Workflow," Oregon State University, 2016.
 - [120] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, "Occupancy Anticipation for Efficient Exploration and Navigation," p. 2020, Aug. 2020.
 - [121] M. Srinivasan Ramanagopal, A. P. Van Nguyen, and J. Le Ny, "A Motion Planning Strategy for the Active Vision-Based Mapping of Ground-Level Structures," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 1, pp. 356–368, Jan. 2018.
 - [122] A. Howard, M. J. Matarić, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Auton. Robots*, vol. 13, no. 2, pp. 113–126, 2002.
 - [123] W. Sheng, H. Chen, and N. Xi, "Navigating a miniature crawler robot for engineered structure inspection," *IEEE Trans. Autom. Sci. Eng.*, vol. 5, no. 2, pp. 368–373, Apr. 2008.
 - [124] A. Ibrahim, A. Sabet, and M. Golparvar-Fard, "BIM-driven mission planning and navigation for automatic indoor construction progress detection using robotic ground platform," in *The 2019 European Conference on Computing in Construction*, 2019.
 - [125] M. Stokkeland, K. Klausen, and T. A. Johansen, "Autonomous visual navigation of Unmanned Aerial Vehicle for wind turbine inspection," in *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, 2015, pp. 998–1007.
 - [126] X. Hui, J. Bian, X. Zhao, and M. Tan, "Vision-based autonomous navigation approach for unmanned aerial vehicle transmission-line inspection," *Int. J. Adv. Robot. Syst.*, vol. 15, no. 1, p. 172988141775282, Jan. 2018.
 - [127] V. A. H. Higuti, A. E. B. Velasquez, D. V. Magalhaes, M. Becker, and G. Chowdhary, "Under canopy light detection and ranging-based autonomous navigation," *J. F. Robot.*, vol. 36, no. 3, pp. 547–567, May 2019.
 - [128] C. Koch, S. P. German, A. Rashidi, Z. Zhu, M. König, and I. Brilakis, "Achievements and Challenges in Machine Vision-Based Inspection of Large Concrete Structures," *Adv. Struct. Eng.*, vol. 17, no. 3, pp. 303–318, 2014.
 - [129] Z. Zhu and I. Brilakis, "Concrete Column Recognition in Images and Videos," *J. Comput. Civ. Eng.* 24.6, pp. 478–487, 2010.
 - [130] C. M. Yeum, S. J. Dyke, J. Ramirez, and B. Benes, "Big visual data analytics for damage classification in civil engineering," in *International Conference on Smart Infrastructure and Construction*, 2016, pp. 569–574.
 - [131] Y. Gao and K. M. Mosalam, "Deep Transfer Learning for Image-Based Structural Damage Recognition," *Comput. Civ. Infrastruct. Eng.*, vol. 00, pp. 1–21, 2018.
 - [132] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning Deep Features for Discriminative Localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
 - [133] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *International Conference on Computer Vision*, 2016, vol. 7, no. 8.

- [134] X. Xiong, A. Adan, B. Akinci, and D. Huber, "Automatic creation of semantically rich 3D building models from laser scanner data," *Autom. Constr.*, vol. 31, pp. 325–337, 2013.
- [135] Y. Perez-Perez, M. Golparvar-Fard, and K. El-Rayes, "Semantic and Geometric Labeling for Enhanced 3D Point Cloud Segmentation," *Constr. Res. Congr. 2016*, pp. 2542–2552, 2016.
- [136] I. Armeni *et al.*, "3D Semantic Parsing of Large-Scale Indoor Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1534–1543.
- [137] J. DeGol, M. Golparvar-Fard, and D. Hoiem, "Geometry-Informed Material Recognition," Jul. 2016.
- [138] Y. Ham, K. K. Han, J. J. Lin, and M. Golparvar-Fard, "Visual monitoring of civil infrastructure systems via camera-equipped Unmanned Aerial Vehicles (UAVs): a review of related works," *Vis. Eng.*, vol. 4, no. 1, p. 1, Dec. 2016.
- [139] R. Lu, I. Brilakis, and C. R. Middleton, "Detection of Structural Components in Point Clouds of Existing RC Bridges," *Comput. Civ. Infrastruct. Eng.*, vol. 34, no. 3, pp. 191–212, Mar. 2019.
- [140] I. Abdel-Qader, O. Abudayyeh, and M. E. Kelly, "Analysis of Edge-Detection Techniques for Crack Identification in Bridges," *J. Comput. Civ. Eng.*, vol. 17, no. 4, pp. 255–263, Oct. 2003.
- [141] Z. Zhu, S. German, and I. Brilakis, "Visual retrieval of concrete crack properties for automated post-earthquake structural safety evaluation," *Autom. Constr.*, vol. 20, no. 7, pp. 874–883, Nov. 2011.
- [142] Z. Chen and J. Chen, "Mobile Imaging and Computing for Intelligent Structural Damage Inspection - Open Access Library," *Adv. Civ. Eng.*, 2014.
- [143] C. M. Yeum and S. J. Dyke, "Vision-Based Automated Crack Detection for Bridge Inspection," *Comput. Civ. Infrastruct. Eng.*, vol. 30, no. 10, pp. 759–770, Oct. 2015.
- [144] L. Daniel Otero, A. Peter, and M. Moyou, "Remote Sensing with Mobile LiDAR and Imaging Sensors for Railroad Bridge Inspections Final Report for Rail Safety IDEA Project 26," 2016.
- [145] S. German, I. Brilakis, and R. DesRoches, "Rapid entropy-based detection and properties measurement of concrete spalling with machine vision for post-earthquake safety assessments," *Adv. Eng. Informatics*, vol. 26, no. 4, pp. 846–858, Oct. 2012.
- [146] B. F. Spencer, V. Hoskere, and Y. Narazaki, "Advances in Computer Vision-Based Civil Infrastructure Inspection and Monitoring," *Engineering*, vol. 5, no. 2, pp. 199–222, 2019.
- [147] L. Sun, Z. Shang, Y. Xia, S. Bhowmick, and S. Nagarajaiah, "Review of Bridge Structural Health Monitoring Aided by Big Data and Artificial Intelligence: From Condition Assessment to Damage Detection," *J. Struct. Eng. (United States)*, vol. 146, no. 5, p. 04020073, May 2020.
- [148] Y. J. Cha, W. Choi, and O. Büyüköztürk, "Deep Learning-Based Crack Damage Detection Using Convolutional Neural Networks," *Comput. Civ. Infrastruct. Eng.*, vol. 32, no. 5, pp. 361–378, May 2017.
- [149] Y. Xu, Y. Bao, J. Chen, W. Zuo, and H. Li, "Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images," *Struct. Heal. Monit.*, vol. 18, no. 3, pp. 653–674, May 2019.
- [150] M. Maguire, S. Dorafshan, and R. J. Thomas, "SDNET2018: A concrete crack image dataset for machine learning applications," *Browse all Datasets*, 2018. [Online]. Available: https://digitalcommons.usu.edu/all_datasets/48.

- [151] C. M. Yeum, "Computer vision-based structural assessment exploiting large volumes of images," *Theses Diss. Available from ProQuest*, Jan. 2016.
- [152] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-Based Convolutional Networks for Accurate Object Detection and Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.
- [153] Y. J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, and O. Büyüköztürk, "Autonomous Structural Visual Inspection Using Region-Based Deep Learning for Detecting Multiple Damage Types," *Comput. Civ. Infrastruct. Eng.*, vol. 33, no. 9, pp. 731–747, Nov. 2018.
- [154] X. Pan and T. Y. Yang, "Postdisaster image-based damage detection and repair cost estimation of reinforced concrete buildings using dual convolutional neural networks," *Comput. Civ. Infrastruct. Eng.*, vol. 35, no. 5, pp. 495–510, May 2020.
- [155] X. Liang, "Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with Bayesian optimization," *Comput. Civ. Infrastruct. Eng.*, Dec. 2018.
- [156] V. Hoskere, Y. Narazaki, T. A. Hoang, and B. F. Spencer, "MaDnet: Multi-task Semantic Segmentation of Multiple types of Structural Materials and Damage in Images of Civil Infrastructure," *Submitt. to J. Civ. Struct. Heal. Monit.*, 2020.
- [157] E. McLaughlin, N. Charron, and S. Narasimhan, "Automated Defect Quantification in Concrete Bridges Using Robotics and Deep Learning," *J. Comput. Civ. Eng.*, vol. 34, no. 5, p. 04020029, Sep. 2020.
- [158] C. Xiong, Q. Li, and X. Lu, "Automated regional seismic damage assessment of buildings using an unmanned aerial vehicle and a convolutional neural network," *Autom. Constr.*, vol. 109, p. 102994, Jan. 2020.
- [159] M. M. Torok, M. Golparvar-Fard, and K. B. Kochersberger, "Image-Based Automated 3D Crack Detection for Post-disaster Building Assessment," *J. Comput. Civ. Eng.*, vol. 28, no. 5, p. A4014004, Sep. 2013.
- [160] M. J. Olsen, M. Asce, ; Falko Kuester, B. J. Chang, S. M. Asce, and T. C. Hutchinson, "Terrestrial Laser Scanning-Based Structural Damage Assessment."
- [161] B. Guldur and J. F. Hajjar, "Automated Classification of Detected Surface Damage from Point Clouds with Supervised Learning," in *The 33rd International Symposium on Automation and Robotics in Construction (ISARC 2016)*, 2016.
- [162] D. Huber, "ARIA: the Aerial Robotic Infrastructure Analyst," *SPIE Newsroom*, Jun. 2014.
- [163] M. KOBAYASHI, K. MIZUNO, M. KURAOKA, S. NOZAWA, and T. ISHIBASHI, "STUDY ON DAMAGE CAUSED TO COLUMNS OF RC VIADUCTS BY THE 2011 OFF THE PACIFIC COAST OF TOHOKU EARTHQUAKE," *J. Japan Soc. Civ. Eng. Ser. A1 (Structural Eng. ^|^ Earthq. Eng.*, vol. 69, no. 4, p. I_790-I_797, 2013.
- [164] V. Hoskere, Y. Narazaki, and B. F. Spencer, "Learning to Detect Important Visual Changes for Structural Inspections using Physics-based Graphics Models," in *9th International Conference on Structural Health Monitoring of Intelligent Infrastructure*, 2019.
- [165] V. Hoskere, Y. Narazaki, B. F. Spencer, and M. D. Smith, "Deep learning-based damage detection of miter gates using synthetic imagery from computer graphics." DEStech Publications Inc., pp. 3073–3080, 01-Jan-2019.
- [166] Y. Narazaki, V. Hoskere, B. A. Eick, M. D. Smith, and B. F. Spencer, "Vision-based dense displacement and strain estimation of miter gates with the performance evaluation using physics-based graphics models," *Smart Struct. Syst.*, vol. 24, no. 6, pp. 709–721, 2019.
- [167] Y. Narazaki, F. Gomez, V. Hoskere, M. D. Smith, and B. F. Spencer, "Efficient

- development of algorithms for vision-based dense 3D displacement measurement using physics-based graphics models,” *Submitt. to J. Struct. Heal. Monit.*, 2020.
- [168] Y. Narazaki, V. Hoskere, T. A. Hoang, and B. F. Spencer, “Automated Bridge Component Recognition using Video Data,” in *The 7th World Conference on Structural Control and Monitoring (7WCSCM)*, 2018.
 - [169] M. Kono and Y. Matsumoto, “DESIGN OF THE STANDARD RIGID FRAME RAILWAY BRIDGE IN NEW TOKAIDO LINE,” *Trans. Japan Soc. Civ. Eng.*, vol. 1965, no. 115, pp. 13–25, Mar. 1965.
 - [170] M. Ohba, “The Design History of the Railway Viaduct from the Design of Tokaido Shinkansen to the Recent Design,” *Concr. J.*, vol. 51, no. 1, pp. 112–115, 2013.
 - [171] K. Masashi, K. Shinoda, K. Mizuno, S. Nozawa, and T. Ishibashi, “Study on damage caused to Shinkansen RC viaducts by the 2011 off the pacific coast of Tohoku earthquake,” *J. Japan Soc. Civ. Eng. A1*, vol. 70, no. 4, p. I_688-I_700, 2014.
 - [172] H. Inaguma and M. Seki, “鉄道高架橋柱のポリエステル繊維巻き耐震補強に関する実験的研究 (Experimental study on earthquake strengthening using polyester sheets of RC railway viaduct columns, in Japanese),” *構造工学論文集 A (JSCE J. Struct. Eng. A)*, vol. 50A, no. 2, pp. 515–526, 2004.
 - [173] “Blender API Documentation.” [Online]. Available: <https://docs.blender.org/api/2.79/>. [Accessed: 03-Sep-2019].
 - [174] “Bulk Bing Image downloader.” [Online]. Available: <https://github.com/ostrolucky/Bulk-Bing-Image-downloader>. [Accessed: 16-May-2020].
 - [175] B. Burley and W. Disney, “Physically-Based Shading at Disney,” 2012.
 - [176] “Textures for 3D, graphic design and Photoshop!” [Online]. Available: <https://www.textures.com/>. [Accessed: 05-Jun-2020].
 - [177] M. Pharr and G. Humphreys, *Physically Based Rendering: From Theory To Implementation*. Elsevier Inc., 2010.
 - [178] “Blender Video Tutorials.” [Online]. Available: <http://www.littlewebhut.com/blender/>. [Accessed: 16-May-2020].
 - [179] M. Ikuma and M. Naito, “Structure Planning for Shinkansens,” *Concr. J.*, vol. 49, no. 1, pp. 27–31, 2011.
 - [180] M. Tateyama, “鉄道における土工技術と性能規定化の動向 (in Japanese),” *建設の施工企画*, vol. 709, pp. 33–39, 2009.
 - [181] M. Ikuma and M. Naito, “新幹線の構造物計画(in Japanese),” *Concr. J.*, vol. 49, no. 1, pp. 27–31, 2011.
 - [182] S. Matsushige, “国鉄東海道新幹線工事について(in Japanese),” *Manuf. Technol.*, vol. 16, no. 9, pp. 1–7, 1964.
 - [183] S. Fujita, “III. 電車線路設備,” *J. Inst. Electr. Eng. Japan*, vol. 102, no. 2, pp. 113–114, Feb. 1982.
 - [184] “noise · PyPI.” [Online]. Available: <https://pypi.org/project/noise/>. [Accessed: 18-May-2020].
 - [185] “Building Code Requirements for Structural Concrete (ACI 318-14) Commentary on Building Code Requirements for Structural Concrete (ACI 318R-14) An ACI Standard and Report from IHS,” 2014.
 - [186] N. Jaipuria *et al.*, “Deflating Dataset Bias Using Synthetic Data Augmentation,” Apr. 2020.
 - [187] K. Mikolajczyk, C. Schmid, and A. Zisserman, “Human Detection Based on a Probabilistic

- Assembly of Robust Part Detectors,” in *European Conference on Computer Vision*, 2004, pp. 69–82.
- [188] Bo Wu, R. Nevatia, and Yuan Li, “Segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
 - [189] E. Cervantes, L. L. Yu, A. D. Bagdanov, M. Masana, and J. van de Weijer, “Hierarchical part detection with deep neural networks,” in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 1933–1937.
 - [190] Y. Peng, X. He, and J. Zhao, “Object-Part Attention Model for Fine-grained Image Classification,” *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1487–1500, Apr. 2017.
 - [191] G. L. Oliveira, A. Valada, C. Bollen, W. Burgard, and T. Brox, “Deep learning for human part discovery in images,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1634–1641.
 - [192] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, “Caltech-UCSD Birds-200-2011,” 2011. [Online]. Available: <http://www.vision.caltech.edu/visipedia/CUB-200-2011.html>. [Accessed: 25-Sep-2018].
 - [193] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, “3D Object Representations for Fine-Grained Categorization,” in *4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13)*, 2013.
 - [194] Y. Narazaki, V. Hoskere, T. A. Hoang, and B. F. Spencer Jr., “Vision-based automated bridge component recognition integrated with high-level scene understanding,” in *The 13th International Workshop on Advanced Smart Materials and Smart Structures Technology (ANCRiSST)*, Tokyo, Japan, 2017.
 - [195] Y. Narazaki, V. Hoskere, T. A. Hoang, and B. F. S. Jr, “Automated Vision-based Bridge Component Extraction Using Multiscale Convolutional Neural Networks,” in *3rd Huixian International Forum on Earthquake Engineering for Young Researchers*, 2017.
 - [196] V. Hoskere, Y. Narazaki, T. A. Hoang, and B. F. Spencer, “Vision-based Structural Inspection using Multiscale Deep Convolutional Neural Networks,” in *3rd Huixian International Forum on Earthquake Engineering for Young Researchers*, 2017.
 - [197] J. Davis and M. Goadrich, “The relationship between Precision-Recall and ROC curves,” in *Proceedings of the 23rd international conference on Machine learning - ICML '06*, 2006, pp. 233–240.
 - [198] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3234–3243.
 - [199] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
 - [200] S. Gould, R. Fulton, and D. Koller, “Decomposing a Scene into Geometric and Semantically Consistent Regions,” in *Proceedings of International Conference on Computer Vision (ICCV)*, 2009, pp. 1–8.
 - [201] Ce Liu, J. Yuen, and A. Torralba, “SIFT Flow: Dense Correspondence across Scenes and Its Applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 978–994, May 2011.

- [202] “Google Street View.” [Online]. Available: <https://www.google.com/streetview/>. [Accessed: 25-Sep-2018].
- [203] “Google Street View API.” [Online]. Available: <https://developers.google.com/maps/documentation/streetview/intro>. [Accessed: 25-Sep-2018].
- [204] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, 2012, pp. 1097–1105.
- [205] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *The 3rd International Conference for Learning Representations*, 2014.
- [206] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” Mar. 2016.
- [207] Y. LeCun, Y. Bengio, G. Hinton, L. Y., B. Y., and H. G., “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [208] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017.
- [209] R. P. K. Poudel, S. Liwicki, and R. Cipolla, “Fast-SCNN: Fast Semantic Segmentation Network,” Feb. 2019.
- [210] “Labelbox: The leading training data platform.” [Online]. Available: <https://labelbox.com/>. [Accessed: 19-May-2020].
- [211] D. Eigen and R. Fergus, “Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2016, vol. 11-18-Dece, pp. 2650–2658.
- [212] D. D. Margineantu, “When Does Imbalanced Data Require Cost-Sensitive Learning? more than,” 2000.
- [213] “The KITTI Vision Benchmark Suite.” [Online]. Available: <http://www.cvlibs.net/datasets/kitti/>. [Accessed: 20-Apr-2020].
- [214] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, “Deep Ordinal Regression Network for Monocular Depth Estimation,” 2018.
- [215] J. Uhrig, M. Cordts, U. Franke, and T. Brox, “Pixel-level Encoding and Depth Layering for Instance-level Semantic Labeling,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9796 LNCS, pp. 14–25, Apr. 2016.
- [216] Y. Takahashi, “2011年東北地方太平洋沖地震 被害調査報告 [Report on damage caused by the 2011 Tohoku Earthquake, in Japanese],” 2011 (in Japanese).
- [217] X. Yang, Y. Gao, H. Luo, C. Liao, and K. T. Cheng, “Bayesian DeNet: Monocular Depth Prediction and Frame-Wise Fusion with Synchronized Uncertainty,” *IEEE Trans. Multimed.*, vol. 21, no. 11, pp. 2701–2713, Nov. 2019.
- [218] “Agisoft Metashape.” [Online]. Available: <https://www.agisoft.com/>. [Accessed: 30-Aug-2020].
- [219] H. Kim, J. Yoon, and S. Sim, “Automated bridge component recognition from point clouds using deep learning,” *Struct. Control Heal. Monit.*, vol. 27, no. 9, Sep. 2020.
- [220] Y. Perez-Perez, M. Golparvar-Fard, and K. El-Rayes, “Artificial Neural Network for Semantic Segmentation of Built Environments for Automated Scan2BIM,” 2019, pp. 97–104.
- [221] C. Martin and S. Thrun, “Real-time acquisition of compact volumetric 3D maps with mobile robots,” *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 1, pp. 311–316, 2002.

- [222] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [223] Y. Chen and G. Medioni, “Object modeling by registration of multiple range images,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 1991, vol. 3, pp. 2724–2729.
- [224] Y. Narazaki, V. Hoskere, K. Yoshida, B. F. Spencer, and Y. Fujino, “Synthetic environments for vision-based structural condition assessment of Japanese high-speed railway viaducts,” *Submitt. to Mech. Syst. Signal Process.*, 2020.
- [225] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.
- [226] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin, and A. Paques, “PythonRobotics: a Python code collection of robotics algorithms,” Aug. 2018.
- [227] Y. Takahashi, “東北地方太平洋沖地震 構造物被害報告 (in Japanese),” 2011.
- [228] A. Tao, K. Sapra, and B. Catanzaro, “Hierarchical Multi-Scale Attention for Semantic Segmentation,” May 2020.
- [229] “NVIDIA Jetson Nano Developer Kit | NVIDIA Developer.” [Online]. Available: <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>. [Accessed: 29-Sep-2020].