DISSERTATION

DIGITAL TWINS FOR STRUCTURAL INSPECTION, ASSESSMENT, AND MANAGEMENT

Submitted by

Brandon J. Perry

Department of Civil and Environmental Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2023

Doctoral Committee:

Advisor: Yanlin Guo

Rebecca Atadero
John van de Lindt
Hussam Mahmoud
Francisco Ortega

ABSTRACT

DIGITAL TWINS FOR STRUCTURAL INSPECTION, ASSESSMENT, AND MANAGEMENT

With the rapid advancements in remote sensing, uncrewed aircraft systems (UAS), computer vision, and machine learning, more techniques to maintain and evaluate the performance of the built infrastructure become available; however, these techniques are not always straightforward to adopt due to the remaining challenges in data analytics and the lack of executable actions that can be taken. The paper proposes a Digital Twin, which is a virtual representation of structures and has a myriad of applications to better assess and manage civil infrastructure. The proposed Digital Twin includes the techniques to store, visualize, and analyze the data collected from a UAS-enabled remote sensing inspection and computational models that support decision-making regarding the maintenance and operation of structures. The data analysis module identifies the location, extent, and growth of a defect over time, the structural components, and connections from the collected image with artificial intelligence (AI) and computer vision. In addition, the three-component (3C) dynamic displacements are measured from videos of the structure. A model library within the digital twin to assess the structure's performance, which includes three types of models, is proposed: 1) a visualization model to provide location-based data query, 2) an automatically generated finite element (FE) model as a basis for simulation, and 3) a surrogate model which can quickly predict a structure's behavior. Ultimately, the models in the library suggest executable actions that can be taken on a structure to better maintain and repair it. A discussion is presented showing how the Digital Twin can assist decision-making for structural management.

ACKNOWLEDGEMENTS

I would first like to thank my advisor, Dr. Yanlin Guo, and her lab group at Colorado State University. Dr. Guo was always available and eager to assist whenever I had a question about research, writing, or life.

I would also like to thank my committee members and other professors, who without their passionate participation and input, this project would not be successful: Dr. Rebecca Atadero, Dr. John W. van de Lindt, Dr. Hussam Mahmoud, Dr. Paul R. Heyliger, and Dr. Francisco Ortega.

I want to recognize the assistance Mr. Christopher Robertson and his team provided at the CSU Drone Center for their valuable support and expertise in the drone experiments of this study as well as Mr. Todd Atadero of Colorado State University for his insight and assistance on camera hardware and experimental design.

The funding for this project was provided, in part by the Mountain-Plains Consortium, a University Transportation Center funded by the U.S. Department of Transportation (FASTACT Grant No. 69A3551747108), Colorado State University Thornton Tomasetti through the 6th Annual Student Innovation Fellowship.

DEDICATION

I would like to dedicate this dissertation to my parents, who have shown me unwavering support, and to my partner, family, and friends for their constant encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background

As computer vision, machine learning, and uncrewed aircraft systems (UAS) technologies evolve, more techniques to maintain and evaluate the built infrastructure become available. Despite the recent progress, the full potential of these advancements still is rarely realized in practice due to the lack of executable actions a bridge manager can take regarding repairs and maintenance decisions. Department of Transportations (DOTs) find the use of new technology difficult and impractical from a perceived lack of benefit and additional work required to collect and analyze the data [2]. Digital Twins (DTs), which first appeared in the manufacturing realm and aircraft maintenance with the United States Air Force (USAF) provides a basis to analyze the data and suggest executable actions automatically. A DT is a virtual representation of the real world with a myriad of applications. DTs have made an appearance in both research and practical applications as they provide stakeholders with a potential framework for the management of assets. DTs represent the current state of a specific asset and are used as a basis for simulation to assess the current condition and predict future wear and performance. The application of DTs to civil infrastructure is explored as a means to better manage the built environment, maintain the existing infrastructure, and achieve the American Society of Civil Engineering's (ASCE's) *Vision 2025* for a more sustainable and efficient future [3].

1.2 Motivation

A DT platform provides a basis for a streamlined inspection, data analytics, simulation, performance assessment, performance prediction, and decision-making. Under current bridge management practices, there is little opportunity to provide a holistic overview of the structure's current state and future performance. First, during the inspection process, conventional bridge inspec-

tion procedures regulated by the Federal Highway Administration's (FHWA) Bridge Inspection Reference Manual and The American Association of State Highway Transportation Officials' (AASHTO) Manual for Bridge Evaluation [4, 5], can be relatively subjective, unsafe, slow, and costly. Inconsistency in ratings due to the subjectivity of human inspectors is very common [6]. Variations of ratings in routine, bi-yearly inspections of ±2 points or greater have been reported for 32% of bridges [7]. The direct and indirect costs of the current inspection procedure including renting/purchasing/operating the equipment, the wages of the inspectors, and the economic impact due to prolonged traffic interruptions can be significant. Moreover, once the inspection is complete, typically a report is written documenting the defects or found problems. In the next inspection cycle, the report will typically be reviewed prior to the inspection and may not be used again to assist in future decision-making. The full potential of the collected data is not realized. There is a lack of executable action that can be directly attained from the inspections and created reports. To take full advantage of information that can be gathered during an inspection and provide a more comprehensive framework to represent the current and future condition of a structure, a new framework needs to be developed. To tackle these aforementioned issues, a DT is proposed which is a comprehensive platform to provide a holistic overview of a structure's health. In this context, a DT is developed to provide a complete digital representation of the physical structure. In the proposed platform, a model library will be incorporated within the DT to provide a basis for simulation, performance assessment, and future performance predictions. The DT provides: 1) a framework to capture data with optical cameras attached to a UAS in a cheap and portable platform, 2) a module for the analysis of the collected data to identify defects and anomalies with computer vision and artificial intelligence (AI), and 3) a model library surrogate models, visualization models, and finite element (FE) models to allow for more in-depth analysis and simulations. A comprehensive framework is proposed to provide various metrics to assess the current state, models to predict future performance, and a module to assist in the decision-making of the structure. A DT is constructed on a bridge-by-bridge basis through the proposed framework to allow for performance assessments and decision-making on specific structures.

1.3 Literature Review

In the literature, researchers often refer to building information models (BIMs) and model updating in general as DTs; however, there are distinctions among the three [8, 9]. BIMs are a convenient tool for understanding the performance of a structure. BIMs incorporate a simple storage solution with localized information (i.e., a visualization model with clickable attributes to display information) [10, 11]. For instance, BIMs are able to display information about defects in a structure such as localization and quantitative metrics [10, 12, 13]. In [10], the authors were able to build a visualization model with images collected with UASs and map the component-level defects onto a visualization model with the use of "damage cubes" (the term "damage cubes" was coined in [12]). However, a BIM cannot model the structure's behavior, nor can it provide decision-making support on how to maintain and repair the structure. On the other hand, model updating in general can provide updated or real-time information about a structure and provide assessments of the current condition [14–17]. For instance, in [15], the authors created a large-scale, high-fidelity FE model of a steel substructure of a power plant, updated the FE model with data from ten accelerometers, and estimated the fatigue damage. In [16], using accelerometers attached to a structure (i.e., the Canton Tower in China), an FE model was updated and validated based on sensor data, and the uncertainty of the stiffness parameters was estimated. In another example in [17], a model updating methodology is discussed to estimate the stiffness reduction of a structure due to damage. Although model updating in general is a powerful tool to improve a model based on data, it does not offer the benefits of a BIM nor provide support to facilitate operations and inspections of a structure. A DT, in contrast, incorporates a variety of models including data-driven (i.e., AI), visualization, and computational models to holistically represent the physical structure. A DT supports decision-making and management by providing structural performance assessments, predictions, and simulations under various operational and maintenance conditions [18, 19]. Essentially, a DT can offer more advantages beyond those of BIM and model updating.

DTs have been used within the USAF and have been explored for civil structures. Within the USAF, DTs represent the current status of a specific aircraft and are used as a basis for simulation to predict future wear and decay [19, 20]. During inspections, the data is linked to the DT and updated to provide the most accurate information possible. Simulations are executed virtually to understand how the aircraft will perform in certain situations and modify operational modes if needed. Data from the aircraft and environment is fused together and provides real-time assessments of the aircraft [20]. Compared to the application of DT in aircraft, its adoption in civil structures is still rudimentary. [21] developed a small-scale DT which incorporated a weigh-in-motion (WIM) sensor and Revit model to determine in real-time the weight of a scale truck over a bridge and save the data with the BrIM. [22] developed a scale model DT of a two-story frame with attached sensors to demonstrate a proof of concept of physical-to-virtual interactions with a Bayesian model to demonstrate decision-making. A DT has been proposed in [23] for a bridge outfitted with 291 sensors and provided real-time assessments of the structure using a visualization model. Although real-time surveillance provided a wealth of knowledge, attaching multiple sensors on many structures within a DOT would be unpractical. A cost-effective DT for civil infrastructure constructed using a portable remote-sensing platform is still in need of development.

Chapter 2

Digital Twin Framework

Leveraging the advancements of structural health monitoring (SHM), a DT provides a platform to store and update collected data, analyze and organize the data, model structural behavior, and perform assessments and predictions of the structure for decision-making. Despite the DT's benefit for decision-making, manually assembling, annotating, and updating a DT on a bridge-to-bridge basis would be too tedious, time-consuming, and impractical. Therefore, to streamline decision-making and provide quantitative information about a structure, a DT needs to be established and updated with minimal user input. Leveraging the power of UASs provides a cheap, easy, and portable data-collection tool for infrastructure inspection. This section will discuss the workflow of building and using a DT from a UAS inspection of the structure.

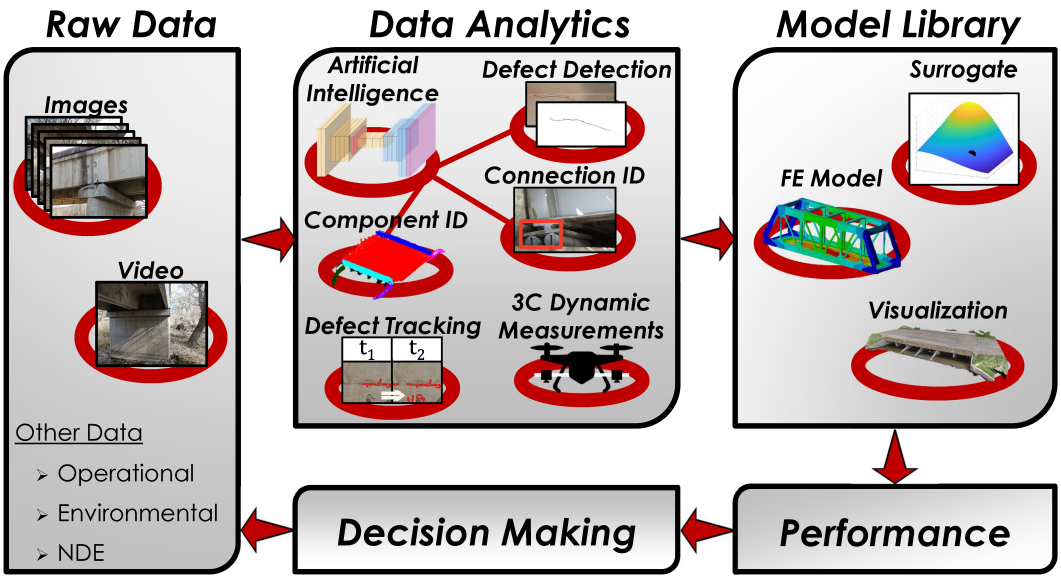


Figure 2.1: Evolution and modules of the Digital Twin

The evolution of the DT follows the framework presented in Figure 2.1. First, an inspection of a structure is performed using optical sensors attached to a UAS. The optical sensors will collect the raw images and record video. Additional data such as operational (i.e., average daily traffic,

vehicle loads, etc.), environmental (i.e., wind, temperature, etc.), and non-destructive evaluation (NDE) results (i.e., delamination measurements, concentration of rust, rebar section loss, etc.) can also be collected from the structure and incorporated within the DT; however, this paper only examines processing the load data, images, and video collected by UAS. Once the data is collected, it first passes through preliminary analysis. The preliminary data analysis finds: 1) the defects and cracks on the structure through AI and computer vision, 2) the growth rate of the defects, 3) the 3-component (3C) dynamic displacements from the collected videos, and 4) the segmentation of the components of the structure from the collected images through AI. Next, with the preliminary data analysis complete, the model library within the DT is established for performance assessment and prediction. The advantage of a DT is the ability to query and fuse different types of data as inputs to visualization, assessment, and prediction models. There are a variety of model types that can be utilized within a DT: 1) visualization (i.e., 3D point cloud, photorealistic models), 2) AI (i.e., defect detection, component detection models), 3) FE (i.e., high or low-fidelity FE models) and 4) surrogate (i.e., stress intensity factor models). The visualization model is used as a location-based query tool to identify and select data and is the basis for the user to interact with the DT. The additional model types perform simulations and assessments of the structure to facilitate decision-making. Predictions can be made based on the data and additional models to understand future performance and allow for cost/benefit analysis on how to maintain a structure. In the following chapters, each aspect of the DT will be explained in detail on how to build and use the DT. First, the data analytics module is explained with each data analytic tool discussed in Chapter 3. With the preliminary data analysis complete, the model library is presented in Chapter 4. Two case studies are presented in Chapter 5 on a concrete bridge and a steel bridge to show how a DT integrates data and model to provide decision-making support. Lastly, the conclusions and anticipated contributions are discussed in Chapter 6.

Chapter 3

Data Analytics

3.1 Introduction

In this module, the raw data collected by the UAS will be initially processed to be used directly for decision-making or treated as input into the model libraries. There are five blocks within this module to analyze the data. Block #1 first identifies the defects in the structure (Section 3.2). An AI network is proposed to identify the cracks in the images at a pixel level. The network is trained with images of steel cracks. Block #2 identifies the growth of a defect over time if successive inspections occur (Section 3.3). The proposed methodology was tested on concrete cracks in a laboratory setting, but the general methodology is applicable to other defects in various materials as discussed in the section. Block #3 is able to measure the 3C dynamic displacement of the structure. In this block, three different methodologies were developed. In Section 3.4 the 2C dynamic displacements of a blast-loaded cable are measured using a single camera. A field test is conducted with a UAS hovering above a cable as a small explosive is detonated to excite the cable. Next in Section 3.5, using a single sensor that incorporates infrared and optical cameras, the 3C dynamic displacements are measured for close-range measurements and tested in a laboratory setting. The last methodology of Block #3 in Section 3.6 provides a more general approach to measure 3C dynamic displacements of full-sized structures using four optical sensors attached to a UAS. In Block #4, the components of the structure are identified using the collected images (Section 3.7). This block provides a framework to have component-level damage and streamline the FE model compilation discussed later in Section 4.2. An AI network is proposed and is trained using a synthetic dataset of concrete railway viaducts; however, application on real-world structures is also discussed. Lastly, Block #5 proposes an AI network to automatically identify the connections of a structure from the images collected (Section 3.8). The technique to identify the location and type of connection within images of a bridge is proposed. Two networks are trained

on data of steel and concrete connections. Each of the aforementioned blocks is discussed in the following with separate subsections of motivation, objectives, methodology, and results.

3.2 Task 1: Defect Identification

3.2.1 Task 1: Motivation

Early detection and assessment of cracks is a vital step for guiding effective maintenance and repair strategies to prolong a structure's life [24]. A delayed crack detection may lead to the progression to more severe defects, causing structure closure and significant cascading economic loss due to impaired mobility. This is evident by the closure of the Interstate-40 bridge across the Mississippi River near Memphis, Tennessee, in May 2021 due to the failure of early detection of a crack in a steel truss member. The indirect cost was approximately 2.4 million USD per day of closure for the trucking industry due to the longer travel times, and increased traffic load along the detour route [25]. Although a drone video in May 2019 showed the initial crack on the Interstate-40 bridge, the inspector did not report it. This type of human error and its severe consequences highlight the need to develop more reliable early crack detection and structural assessment techniques. This section discusses the use of an AI network to automatically find the pixel-level location of the cracks in steel members. This proposed technique has been published in [26].

The research on leveraging the advances in computational power (with the use of graphical processing units (GPUs) and tensor processing units (TPUs)), machine learning (ML), and remote sensing to enhance civil infrastructure inspection has become very active [27–37]. ML algorithms such as neural networks have been used in the identification of cracks from images of steel and concrete structures to provide more reliable inspections compared to the human-based visual inspection or data analysis [30–37]. For concrete structures, various neural networks (i.e., [34–37]) or computer vision techniques (i.e., black hat transform and canny edge detection) [10, 38–41] have been successfully used to identify cracks at a pixel-level. However, for steel structures, most existing studies only outlined the region of the defect using neural networks; while studies on the pixel-level location of the crack or defect are rare [42]. This is mainly because the pixel-level

identification with neural networks is more challenging for steel structures since steel cracks often have a smaller width (typically on the order of 0.05 - 1.5-*mm*) and images of steel have a greater variation of color of both the background and cracks (i.e., rusted or paint/coating). One should note that the pixel-level identification of cracks is beneficial for steel structures as it allows precise extraction of crack length, kink angle, and location information needed for in-depth assessment of structural condition (discussed in Section 4.3.) Therefore, there is a need to develop reliable ML techniques for pixel-level steel crack identification.

In this context, an accurate and efficient U-Net architecture to enable pixel-level steel crack identification is proposed. The proposed AI architecture to achieve semantic segmentation is a U-Net which consists of an encoder and decoder. The encoder reduces the input, $P(x, y)$, into a latent-space representation by $z = g(P(x, y))$ and the decoder then predicts the outputs by $\tilde{P}(x, y) = f(z)$. However, having only an encoder-decoder network results in a loss of fine-grained information from the dimension reduction in the encoder; therefore, skip connections are added to link the encoder to the decoder at various levels to retain the fine-grained information. With the skip connections, the model combined semantic information from deeper layers with coarser information and appearance information from shallower and finer layers. The U-Net architecture is chosen because 1) it was originally developed for pixel-level segmentation [43], and 2) it has accurate performance and a fast training process. The proposed U-Net in this study is adapted from a specific U-Net that uses the VGG-19 convolutional neural network (CNN) as the basis for the encoder [36] (later referred to as U-Net-VGG-19). The VGG-19 network was originally developed by [44] as a deep CNN for classification and has demonstrated robust performance in crack classification [30–32, 36], satellite image classification [45], and biomedical image classification [46]. Using VGG-19 as the encoder, the U-Net-VGG-19 has achieved enhanced accuracy due to more convolutional layers than the original U-Net and has been successfully used in the segmentation of concrete cracks. However, the development of U-Net-VGG-19 did not focus on processing speed, and the net has never been applied for steel cracks [36]. A recent study used another modified version of the original U-Net to identify steel cracks on a pixel level; however, this

technique requires additional processing time due to the morphological transformation needed in the post-processing [42]. Through a systematic parametric study, the architecture of the original U-Net-VGG-19 is significantly modified to establish a new U-Net suitable for pixel-level steel crack identification. This modified U-Net focuses on achieving good accuracy and high computational efficiency.

The section is organized as follows. First, the development of the U-Net is discussed in detail, including the components and architecture of the network. Next, the data sets used to train, validate, and test the proposed network are discussed. Lastly, the results of the trained networks are presented.

Objectives

1. Identify cracks in steel structures with images collected with a UAS
2. Perform a parametric study to build an efficient AI network
3. Identify the compromise between the efficiency and accuracy of the proposed architecture

3.2.2 Task 1: Methodology

To quickly and accurately identify the cracks in a steel structure, a U-Net network is developed to determine the pixel-level location of the steel cracks from the images. Other crack identification techniques such as morphological transformations, which perform at a much faster processing speed than neural networks, were tested; however, these techniques struggled to distinguish between markings, dirt, rust, and cracks resulting in a poorer performance [10, 38–41]. The architecture of a U-Net is symmetrical with an *encoder* (i.e., convolutional/pooling layers) and *decoder* (i.e., transposed convolutional layers). Layers from the encoder are added to the decoder layers of the same size; therefore, the architecture of the encoder dictates the architecture of the decoder. The encoder reduces the size and increases the depth of the input through convolutions and pooling to extract features in the image. In this U-Net implementation, the encoder is loosely based on the well-researched *VGG-19* CNN classifier [44]. The VGG-19 has 16 convolutional layers and three

fully connected layers; however, since segmentation, and not classification, was the application in this study, the last three fully connected layers were replaced with convolutional layers. Moreover, to decrease the size of the network and achieve a higher computational efficiency, some of the convolutional layers were removed. For practical use of the U-Net, systematic tests on various network architectures were conducted to measure the effect of the hidden layers of the network with an emphasis on evaluating processing speed and accuracy. With an increase in the network size (i.e., the total number of parameters in the network), a higher accuracy of the network is typically expected but at a slower processing speed, and vice-versa. This trade-off will be discussed further in the results section (Section 3.2.3).

U-Net Architecture for Defect Identification

The following will discuss all the components used to build the proposed U-Net. The components (i.e., convolution, transposed convolution, pooling, normalization, etc.) used within the U-Net are not unique and are the typical components one would find in a neural network or CNN; however, the architectural design is original. The *convolution* is the primary process of the encoder within the U-Net. It is a technique to extract features from an image and store the features in feature maps to determine whether a pixel is cracked or un-cracked. In this proposed architecture, the convolutional size is $(3, 3)$, and the stride is set to $(1, 1)$, so there is no size reduction during a convolution. A rectified linear unit (ReLU) is used for the activation function to bound the output of the convolution and eliminate the vanishing gradient problem. *Normalization* of the features was performed after each Convolutional block. Next, *average pooling* was used to reduce the dimensions of the feature map. At the end of the last two convolutional blocks, a *dropout* layer was added to randomly disconnect 40% of the connections to the next layer during training, which helps to prevent overfitting of the network. On the decoder side of the U-Net, the *transposed convolution* technique with a ReLU activation function was used to transform a sparser feature map into a denser feature map. It is the primary process within the decoder. In this proposed architecture, the size, stride, and shape vary depending on the encoder architecture to ensure that the output label is the same size as the input image. Finally, a *SoftMax* layer was added to scale the final output

so that the sum of all the classes of a pixel is equal to 1. The results can be loosely interpreted as a probability distribution over each pixel to which class the pixel belongs (i.e., the binary class of cracked or non-cracked).

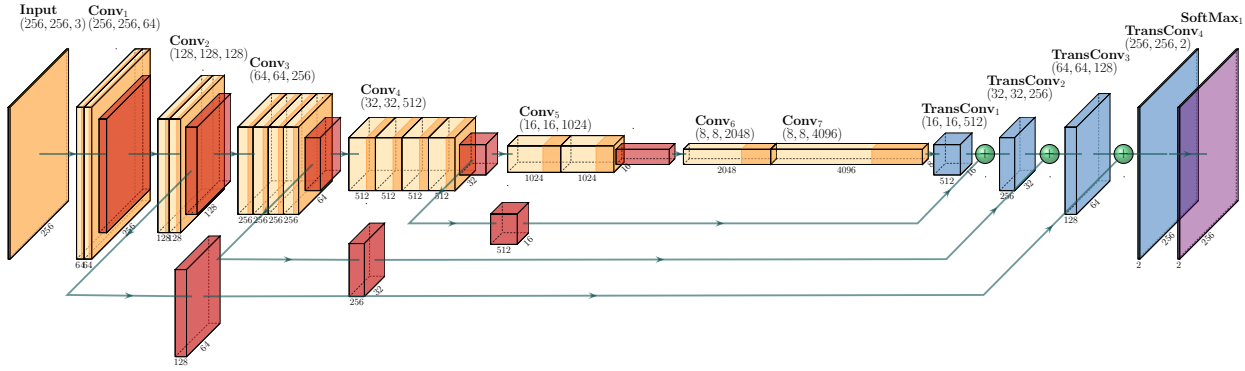


Figure 3.1: Full network architecture of Net_1 (orange layers: convolution operator with a ReLU activation function; red layers: average pooling operator; blue layers: transposed convolution operator with a ReLU activation function; and purple layer: SoftMax operator)

With the components defined, the U-Net architecture is assembled. The initial proposed U-Net architecture is shown in Figure 3.1 and Table 3.1 in the Net_1 column. To further improve the efficiency of the network, a parametric study was conducted to test the accuracy and processing speed of a variety of architectures. The networks with different architectures were built by systematically removing layers of the initial network. The performance of the tested networks on identification accuracy and processing time for a full-sized, 16-*MP* images in the validation set were evaluated against the network size (i.e., the total number of parameters). The initial network has the most layers and number of parameters; the number of layers was subsequently reduced in modified networks. There were a total of 16 networks tested in the preliminary study and the results of five representative nets, Net_2, Net_3, Net_4, and Net_5, were reported in this paper (Table 3.1). Each network was trained using an *Adam* optimizer with the initial parameters set to a He normal distribution [47]. A Python script using the TensorFlow library [48] was used in Google Colab Pro during training with 8 TPUs and 32-*GB* of RAM. The learning rate was set to 0.001, and 201 epochs were performed during the training process with a validation evaluation at every five

epochs. Training for each U-Net took about six *hours*. To further reduce the network size and data storage requirements, gray-scaled images were also tested and evaluated using the same network designs in Table 3.1; however, the initial input layer (i.e., gray-scaled image) would have a shape of $(256, 256, 1)$ instead of $(256, 256, 3)$ used for the color images (Table 3.1). The change of the input layer for processing gray-scaled images slightly modifies the architecture of the U-Net due to the connection between the input layer and the first convolutional layer. Four metrics are defined to determine the accuracy performance of the U-Net: accuracy, recall, precision, and F_1 score. Accuracy is defined as,

$$acc = \frac{\textit{True Positive} + \textit{True Negative}}{\textit{True Positive} + \textit{True Negative} + \textit{False Positive} + \textit{False Negative}} \tag{3.1}$$

Recall is defined as,

$$recall = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}} \tag{3.2}$$

Precision is defined as,

$$precision = \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}} \tag{3.3}$$

Lastly, the F_1 score is defined as,

$$F_1 = 2 * \frac{\textit{precision} * \textit{recall}}{\textit{precision} + \textit{recall}} \tag{3.4}$$

With each metric, a value of 1 is perfect. Next, the processing time is evaluated. The processing time is defined as the total time required to 1) apply a border around a 16-*MP* image, 2) crop the image into 1,014 - 256×256 images, 3) run the cropped images through the U-Net, and 4) restitch the outputs of the U-Net into the full 16-*MP* image. The time evaluation was performed with an Nvidia GeForce GTX 1060 graphics card and 6-*GB* of GPU-RAM. Each image was processed separately, and the average time to process an image was reported.

Table 3.1: Architecture of tested U-Nets (Note, the stride and shape of the *Transposed Convolutional Layers* will vary depending on the U-Net chosen)

Name	Parameter	Size	Stride	Shape	Net_1	Net_2	Net_3	Net_4	Net_5
Input									
Input Image	-	-	-	(256, 256, 3)	X	X	X	X	X
Convolutional Layers									
$Conv_{11}$	64	(3, 3)	(1, 1)	(256, 256, 64)	X	X	X	X	X
$Conv_{12}$	64	(3, 3)	(1, 1)	(256, 256, 64)	X	X	X	X	X
$Norm_1$	-	-	-	(256, 256, 64)	X	X	X	X	X
$Pooling_1$	-	(2, 2)	(2, 2)	(128, 128, 64)	X	X	X	X	X
$Conv_{21}$	128	(3, 3)	(1, 1)	(128, 128, 128)	X	X	X	X	X
$Conv_{22}$	128	(3, 3)	(1, 1)	(128, 128, 128)	X	X	X	X	X
$Norm_2$	-	-	-	(128, 128, 128)	X	X	X	X	X
$Pooling_2$	-	(2, 2)	(2, 2)	(64, 64, 128)	X	X	X	X	X
$Conv_{31}$	256	(3, 3)	(1, 1)	(64, 64, 256)	X	X	X	X	X
$Conv_{32}$	256	(3, 3)	(1, 1)	(64, 64, 256)	X	X	X	X	X
$Conv_{33}$	256	(3, 3)	(1, 1)	(64, 64, 256)	X		X		
$Conv_{34}$	256	(3, 3)	(1, 1)	(64, 64, 256)	X		X		
$Norm_3$	-	-	-	(64, 64, 256)	X	X	X	X	X
$Pooling_3$	-	(2, 2)	(2, 2)	(32, 32, 256)	X	X	X	X	X
$Dropout_1$	40%	-	-	-			X	X	X
$Conv_{41}$	512	(3, 3)	(1, 1)	(32, 32, 512)	X	X	X	X	
$Conv_{42}$	512	(3, 3)	(1, 1)	(32, 32, 512)	X	X	X	X	
$Conv_{43}$	512	(3, 3)	(1, 1)	(32, 32, 512)	X		X		
$Conv_{44}$	512	(3, 3)	(1, 1)	(32, 32, 512)	X		X		
$Norm_4$	-	-	-	(32, 32, 512)	X	X	X	X	
$Pooling_4$	-	(2, 2)	(2, 2)	(16, 16, 512)	X	X	X	X	
$Dropout_2$	40%	-	-	-			X	X	
$Conv_{51}$	1024	(3, 3)	(1, 1)	(16, 16, 1024)	X	X			
$Conv_{52}$	1024	(3, 3)	(1, 1)	(16, 16, 1024)	X	X			
$Norm_5$	-	-	-	(16, 16, 1024)	X	X			
$Pooling_5$	-	(2, 2)	(2, 2)	(8, 8, 1024)	X	X			
$Conv_6$	2048	(2, 2)	(1, 1)	(8, 8, 2048)	X	X			
$Dropout_3$	40%	-	-	(8, 8, 2048)	X	X			
$Conv_7$	4096	(2, 2)	(1, 1)	(8, 8, 4096)	X	X			
$Dropout_4$	40%	-	-	-	X	X			
Transpose Convolutional Layers									
$TransConv_1$	512	(4, 4)			X	X	X	X	
Add_1	$Pooling_4$ + $TransConv_1$				X	X	X	X	
$TransConv_2$	256	(4, 4)			X	X	X	X	X
Add_2	$Pooling_3$ + $TransConv_2$				X	X	X	X	X
$TransConv_3$	128	(4, 4)			X				X
Add_3	$Pooling_2$ + $TransConv_3$				X				X
$TransConv_4$	2	(16, 16)		(256, 256, 2)	X	X	X	X	X
Prediction									
$SoftMax_1$	-	-	-	(256, 256, 2)	X	X	X	X	X

Data Processing

To test the proposed U-Net architectures, three datasets were combined and used for training, validation, and testing (Dataset 1: 120 - 4,928×3,264 resolution images [41, 49], Dataset 2: 50 - 4,928×3,264 resolution images [50], and Dataset 3: 80 - 4,928×3,264 resolution images [49]). The datasets came from publicly available sources. Two of the datasets were pre-labeled; however, the third dataset was only of the steel crack images (i.e., unlabeled) and was, therefore, used for testing purposes. Moreover, since the three datasets came from different sources, with various data capture techniques, cameras, resolutions, and scaling, combining the datasets for use within the U-Net demonstrated the generalizability of the proposed U-Nets.

With the 140 - 4,928×3,264 images and corresponding labels from the 2 labeled datasets, the dataset was ready to be built. The full-sized images and labels were cropped into 256×256 images. Since the crack identification was the purpose of the U-Net, a sliding window was used to search for the cracked location by querying every 10 pixels for a crack label in the center of a 256×256 window. Once a pixel was identified, the image was cropped and randomly shifted up/down and left/right within $\pm$ 110 pixels to ensure that the crack location was not exactly in the center of the image. The sliding window technique for the crack location resulted in 31,072 - 256×256 resolution images of cracks. Because the sliding window technique queried every 10 pixels and the images were randomly shifted $\pm$ 110 pixels, the overlap of the images was reduced to limit providing similar data during training and alleviate overfitting. Next, the background of the images (i.e., where no crack was present) was randomly sampled. Again, a sliding window technique was used; however, the overlap of the images was set to 50%, and only 33% of the images were randomly saved. Additionally, the images were checked to ensure no crack was present. This 50% sliding window technique resulted in 44,484 - 256×256 resolution images without cracks. The resulting dataset was comprised of 41% cracked images and 59% uncracked images. Lastly, the 75,556 images were randomly split into training and validation datasets, with 25% (18,889 images) assigned to validation and 75% (56,667 images) assigned to training. As a note, a similar technique was implemented for cropping the images to a size of 128×128. After

preliminary testing, it was found that the network achieved similar accuracy as the 256×256 data image set; however, because a larger number of the smaller images were required to pass through the network for a full-sized, 16-*MP* image, the resulting processing time was longer. Therefore, the larger, 256×256 images were used for the remainder of the training and data processing.

Lastly, to make the limited dataset apply to a more generalizable application of steel crack detection, the data was randomly altered. The data alteration in this application was not used to increase the number of data points in the dataset; instead, it was used to alter the images after each iteration. Data alteration is typical practice in machine learning and is used to limit overfitting. The first data alteration technique was flipping the images. Fifty percent of the entire training and validation datasets were randomly flipped either left or right. Then, 50% of the entire training and validation datasets were randomly flipped up or down. After the flipping of the images, 50% of the training and validation datasets were color altered. Of the 50% selected for color alteration, 33% had a random and uniform brightness increase or decrease, 34% had a random and uniform contrast increase or decrease, and 33% had a random principal component analysis (PCA) color alteration. PCA color alteration was first used in [51] with the development of the AlexNet for shifting the color balance of the image. The color alterations help to ensure that cracks are robustly identified even when different coatings or lighting conditions are present and help to reduce overfitting.

3.2.3 Task 1: Results

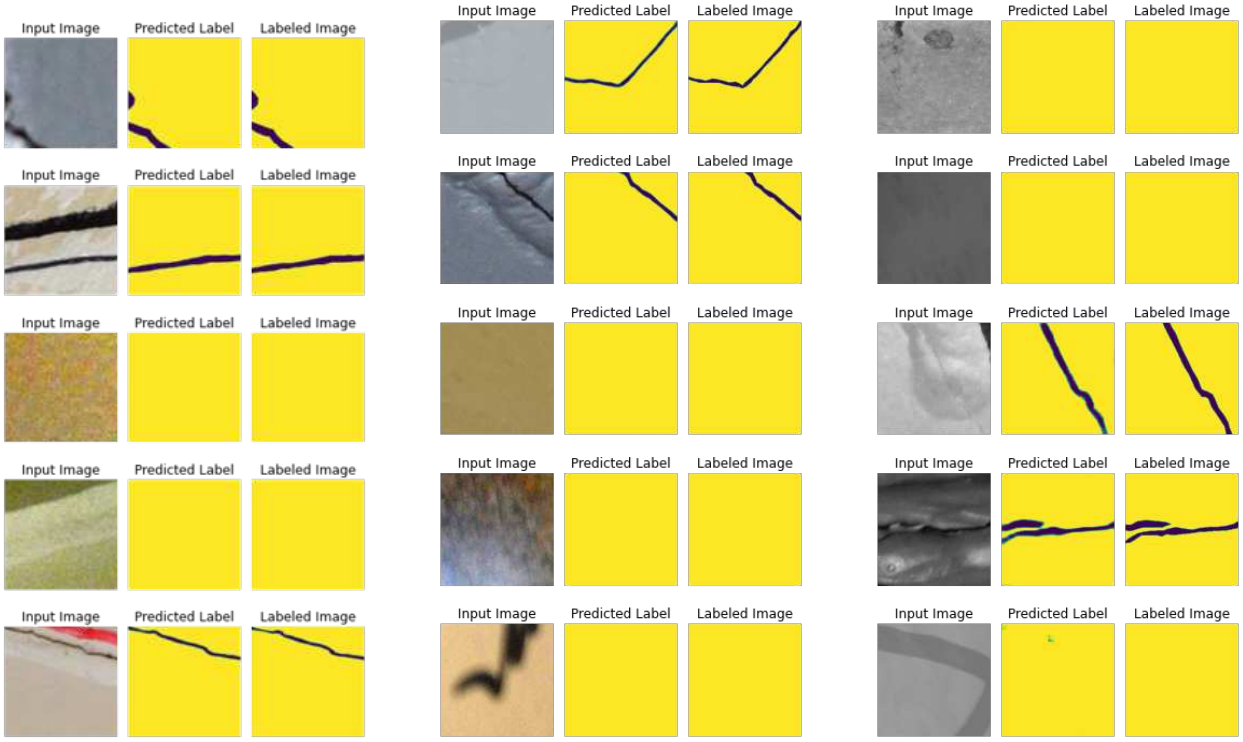
After preparing the data and assembling the components to build the U-Nets, the 10 tested U-Net architectures (i.e., 5 U-Nets with colored inputs and 5 U-Nets with gray-scaled inputs) were trained. The validation results (i.e., accuracy, precision, recall, and F_1 score) of the 10 tested U-Net architectures are presented in Table 3.2. All the networks demonstrated a high accuracy within the validation dataset. The processing time for a 16-*MP* image is plotted against the number of parameters in Figure 3.4. There is a significant decrease in processing time as the size of the U-Net decreases, but the validation accuracy also decreases. A sample of the validation results is shown in Figure 3.2. Additional samples from the best-performing network in the validation dataset, Net_1

with color inputs, are shown in Figure 3.3. From Figure 3.3, there is a higher rate of false-positives over the entire image (identified as red pixels) when the size of the U-Net decreases (i.e., Net_4 and Net_5); however, the cracks are still successfully identified by all the 10 nets. The Net_1 with color inputs (Figure 3.3) showed the best performance with the minimum false positives. A comparison of the proposed U-Net_1 - Color to other machine learning architectures in the literature is shown in Table 3.3. Although there are many other techniques and neural networks to identify cracks in concrete and asphalt, only networks trained to identify cracks in steel members are included in the comparison. All the networks were run using a batch size of 14 images. Note that the batching process ultimately increases the processing time required for all the networks. However, the estimated processing time for all the networks has the same amount of delay due to batching; therefore, the comparison among them is fair. For the pixel level identification, the proposed U-Net_1 - Color outperformed the network in [42] in both accuracy and efficiency. For a UAS-enabled bridge survey, between several hundred to several thousand images can be captured depending on the size and level of detail of the bridge inspection [10,52]. Using the largest and most accurate U-Net (Net_1 with color inputs) to process 3,000 16-*MP* images would take an estimated 194-*min* with 3.88-$^{sec}/_{image}$. Running the smaller and least accurate tested U-Net (Net_5 with gray-scaled inputs) on 3,000 16-*MP* images would take an estimated 106-*min* with 2.11-$^{sec}/_{image}$. With a time-savings of about 1.5-*hours* for a large dataset processing, the trade-off between the higher false-positive rate in the smaller networks with faster processing speeds must be decided.

3.3 Task 2: Damage Growth Tracking

3.3.1 Task 2: Motivation

Under current bridge inspection practices, identifying a defect's growth rate is done in a few different ways: 1) comparing previous inspection notes, 2) comparing previous inspection images, or 3) identifying markings on the structure from previous inspections. These three techniques all provide somewhat qualitative and subjective metrics. On the other hand, if a bridge's inspection is performed with UASs, the image sets from these successive inspections are comparable and



(a) *Net*$_1$ *- Color* **(b)** *Net*$_3$ *- Color* **(c)** *Net*$_5$ *- Gray-Scale*

Figure 3.2: Sample validation results of 256×256 images

Table 3.2: Validation results of the 10 tested U-Net architectures

	Image Color	Total Parameters	Validation Results Accuracy	Precision	Recall	F_1 Score
Net_1	Color	77,776,578	0.9983	0.9984	0.9982	0.9983
	Gray	77,775,426	0.9975	0.9977	0.9973	0.9975
Net_2	Color	71,417,922	0.9982	0.9980	0.9977	0.9979
	Gray	71,416,770	0.9981	0.9982	0.9989	0.9981
Net_3	Color	17,012,290	0.9985	0.9984	0.9981	0.9924
	Gray	17,011,138	0.9983	0.9984	0.9984	0.9982
Net_4	Color	11,112,514	0.9979	0.9982	0.9975	0.9977
	Gray	11,111,362	0.9977	0.9980	0.9973	0.9960
Net_5	Color	2,785,986	0.9964	0.9969	0.9960	0.9964
	Gray	2,784,834	0.9963	0.9970	0.9955	0.9963

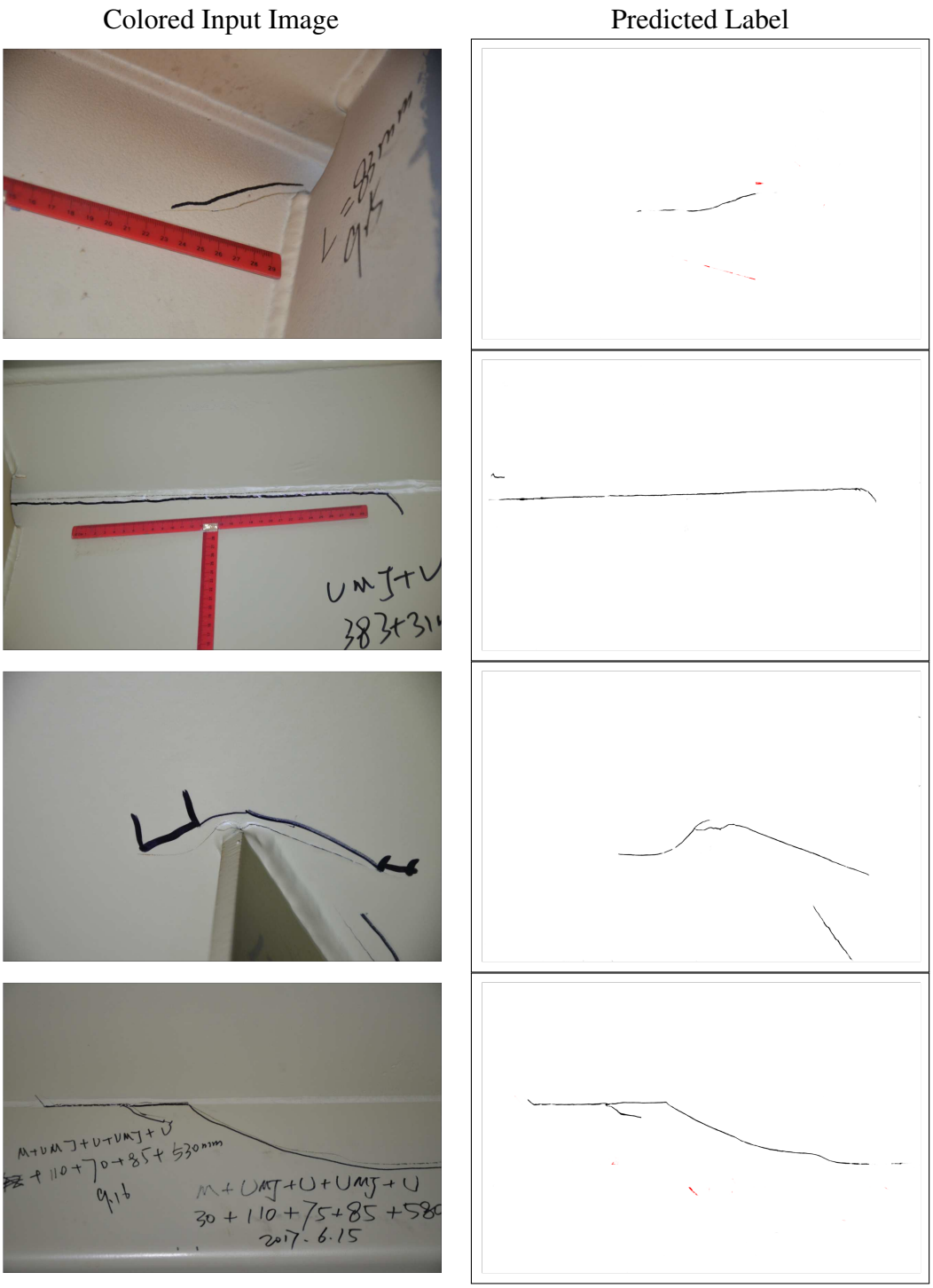


Figure 3.3: Sample outputs from the testing dataset of the best performing U-Net architecture: Net_1 - *Color* (red pixels identify false-positives)

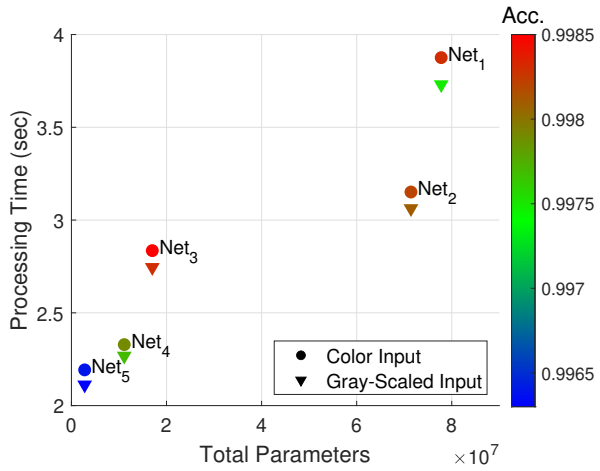


Figure 3.4: Total processing time of a 16-*MP* image as a function of the total parameters and validation accuracy of the 10 tested U-Net architectures

Table 3.3: Comparison of time and metrics of proposed U-Net to other literature references[a]

Study	Pixel-Level	Same Dataset	Network Type	Input Size	Acc. (%)	P (%)	F_1 (%)	Time per image[b] (sec)
[53]	No	No	CNN	(64×64)[c]	98.00	-	55.2	**3.116**
[41]	No	Yes	FCNN	(64×64)[c]	93.90	-	94.70	3.167
[42]	Yes	Yes	U-Net	(512×512)	-	45.80	42.80	7.736
Proposed Net_1 - Color	Yes	-	Modified U-Net	(256×256)	**99.83**	**99.84**	**99.83**	3.616

[a] To run all the networks in the same manner on a GPU with 6-*GB* of RAM, a batch size of 14 was used during prediction

[b] The time includes loading a 16-*GB* image, cropping to input size, running the cropped images through the network, and stitching the cropped images back together. The time does not include any post-processing that may be needed.

[c] The authors in these studies proposed a 50% overlap of the cropped input images to provide a finer-grained crack location prediction.

can be used to facilitate the determination of a defect's growth rate quantitatively and objectively. Therefore, extending the developed defect detection algorithm in the previous section, a module to automatically track the change of a defect over time is developed. This module is especially beneficial to bridge managers, as the growth of a defect often indicates more significant problems in a structure. As a note, this technique was tested and performed using a defect detection technique based on morphological transformations; however, the same methodology can be performed using the output from the AI defect detection technique discussed in Section 3.2. The proposed damage growth tracking technique has been published in [10]. The following will discuss the methodology to track the defect's growth over time and the results from a laboratory experiment of the proposed methodology.

Objectives

1. Provide an automated approach to measure the growth of a defect over time
2. Develop a generalizable methodology to be usable on a variety of defect detection techniques

3.3.2 Task 2: Methodology

To track defect growth through image comparison, the two images need to be perfectly aligned. However, the images collected at different times may have inconsistent geometric distortions (i.e., they are not perfectly aligned) due to dissimilar camera angles, scales, bridge displacements over time, etc. This introduces an additional challenge of comparing images and tracking changes. To tackle this challenge, an automated Affine Transform algorithm is implemented to align the previous image to the current image so that images taken from different angles can be readily compared. To perform this alignment, the (scale invariant feature transform) SIFT algorithm is utilized [54]. SIFT keypoints and their related descriptors are identified and matched across the two images. Once the pixel locations of the key-point matches are found, an Affine Transform, which is a linear mapping method typically used to correct for geometric distortions or deformations that occur with non-ideal camera angles, is performed to warp the previous image to align its shape to the current image; the current image stays constant throughout the algorithm. By knowing

the location of matched keypoints in the previous and current images, the linear transformation is defined as,

$$\mathbf{T} = \mathbf{M} \times \mathbf{X}; \qquad \mathbf{X} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.5}$$

where, $\mathbf{T}$ is a matched keypoint from the current image, $\mathbf{X}$ is a matched keypoint of the previous image, and $\mathbf{M}$ is the linear transformation matrix. $\mathbf{M}$ is firstly solved using the known matches of the keypoints, and then the linear transformation is applied over the entire image.

After the alignment, the morphological crack detection algorithm (published in [10]) is performed on both the previous and current images; however, the algorithm is stopped after the Black Hat Transform. If using an AI defect detection technique, the binary segmentation map will be used. Then, the resulting binary image of the current state is subtracted from the previous image (i.e., the difference between the two images, namely the "difference image" in the following), which indicates the damage growth. If the morphological technique published in [10] is used, the Gaussian Blur convolution, thresholding, and outlining steps (i.e., the remaining steps of the morphological technique) are performed on the "difference image" to highlight the areas of crack growth, thus, the extent of crack growth is quantified. If using the AI defect detection technique, the Gaussian Blur convolution, thresholding, and outlining steps are not needed and the "difference image" directly identifies the growth of the defect.

To test the proposed algorithm, concrete crack growth data need to be collected over time. Due to a typically slow rate of crack growth in reinforced concrete structures, data collection from real structures may take several years. To accelerate the algorithm development, the crack growth data are obtained from split cylinder tests in the laboratory per American Society for Testing and Materials (ASTM) C496 [55]. In the test, a cured concrete cylinder is laid on its side, and a compressive force is applied uniformly over the two points of contact through two compression plates of a universal testing machine (Figure 3.5). This compressive force creates tension in the

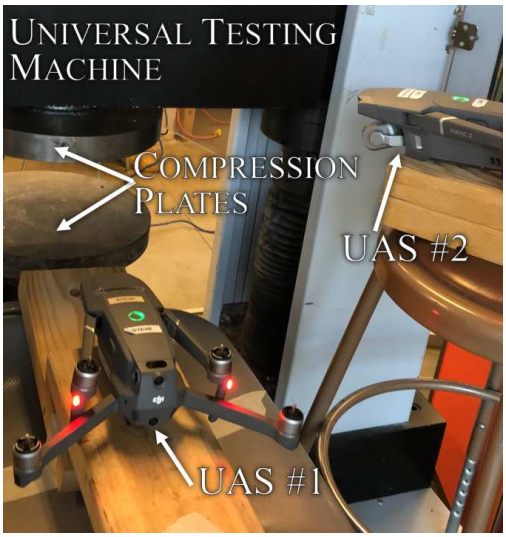


Figure 3.5: Set-Up of experimental data collection

center of the cylinder, which causes a crack to develop gradually within the specimen. To record the crack growth, two UASs (DJI Mavic 2 Zoom (UAS #1 in Figure 3.5) and DJI Mavic 2 Pro (UAS #2 in Figure 3.5)) were placed in stationary positions (as they were unable to hover within the confined space of the testing set-up), pointing in the direction of the universal testing machine to capture images from different angles. The use of the two UASs in the experiment intends to simulate the real-world scenario, where one UAS might take different paths during the two inspections or different UASs, with different cameras, may be used in the two inspections.

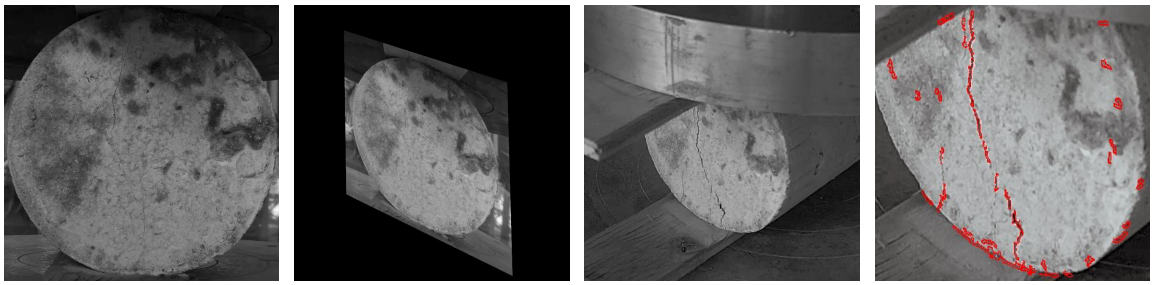
3.3.3 Task 2: Results

Data were collected from two split cylinder tests and used for validation of the proposed methodology. The first test used two images (Figure 3.6a and Figure 3.6b) taken by the same UAS (UAS #1 in Figure 3.5) at the same angle. A human visual inspection of the two pictures shows the growth of about 10 cracks on the side of the cylinder. After performing the Affine Transform and finding the difference as explained earlier, Figure 3.6c shows the resulting "difference image," where the newly grown cracks were successfully identified. While the first test represents the ideal scenario where the two pictures are taken at the same angle by the same camera, the second test aims to evaluate the efficacy of the proposed method for a more realistic scenario, where the previ-

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

Figure 3.6: Test 1 results: (a) previous image, (b) current image, and (c) "difference image" (crack growth highlighted)

ous and current images are taken at different angles by different UASs during different inspections. In the second test, the initial image (Figure 3.7a) was still obtained from the UAS #1 (Figure 3.5), while the UAS #2 (Figure 3.5) was used to collect the image with the progressed cracks over time (Figure 3.7c). The Affine Transform was applied to the initial image (Figure 3.7a), resulting in the warped image (Figure 3.7b) that can be readily compared to the current image (Figure 3.7c). Figure 3.7d shows the identified crack growth. These two tests show that the proposed method can effectively identify the changes in the cracks between the two images taken at different times, highlighting the new cracks grown during the period. This automated identification result of crack growth is consistent with a human visual inspection. The critical step for tracking the changes of cracks over time is the alignment of the images collected from different angles, which relies on matching SIFT keypoints. SIFT keypoint matching has been used to create 3D point clouds and proved to be effective in real-world applications. Therefore, it is expected that the developed crack growth tracking technique is generalizable out of the laboratory and across bridges.



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

Figure 3.7: Test 2 results: (a) previous image, (b) affine transformed previous image, (c) current image, and (d) "difference image" (crack growth highlighted)

3.4 Task 3: 2C Dynamic Displacement Measurements of Blast-Loaded Cables

3.4.1 Task 3: Motivation

Cables are essential elements for cable-stayed bridges, cable-supported roofing systems, communication tower supports, and electrical distribution systems. Measuring the dynamic properties of cables can be cumbersome or impractical if using contact-based sensors such as accelerometers and Linear Variable Differential Transformers (LVDTs) sensors because of the difficulties associated with running wires, placing the sensors, and the deleterious effects on the mass distribution and aerodynamics of the cable. Moreover, the dynamic displacement is difficult to measure using accelerometers because of the loss of low-frequency information. To overcome these limitations, stationary camera-based remote sensing has been used in the displacement measurement of cables [56–59]. However, stationary camera systems may have limitations on the locations of camera placement; and the ideal viewing angle might not always be achievable. Therefore, a more versatile sensing system still needs to be explored. Recently, uncrewed aerial system (UAS)-based platforms have been implemented. [60] measured the displacements of a cantilever beam using a UAS system equipped with active stereo-vision cameras. [61] measured the relative displacements of a cable using a UAS system, but only in pixel units instead of engineering units (e.g., *mm* or *in*).

Alkharisi and Heyliger recently studied the behaviors of power line cables under forced support motion [62]. As an extension of this work, a unique blast experiment was designed to excite the cable vibration. Since a blast is ignited to excite the cable, traditional techniques are not viable options for measuring the displacement of the cable. For example, stationary cameras were tested for measuring displacement but were easily knocked down by the blast loading in the preliminary tests. Additionally, the most advantageous viewing angles may be difficult to obtain with a terrestrial camera. In this context, a novel UAS-enabled dynamic displacement measurement technique for cables is proposed, where a single camera instrumented on the UAS is used.

As the UAS hovers, there is inherent random drift of the system. To minimize the influence of this drift on the measured displacements, a stationary reference target is tracked, and the measured displacement of the stationary reference is subtracted from the recorded signal of cable movement. To improve the quality of the measurement, the change of elevation of UAS during the flight is also considered. From the measured dynamic displacement, the fundamental dynamic properties of the cable can be identified including but not limited to the natural frequency, damping, and mode shape. This task presents the experimental setup, methodology of the proposed new displacement measurement technique using computer vision, and key results. The key findings and best practices are also summarized. This methodology and experimental results are published in [63].

Objectives

1. Perform an initial experiment to explore the feasibility of using optical cameras to measure dynamic displacements

2. Find limitations of the optical sensors

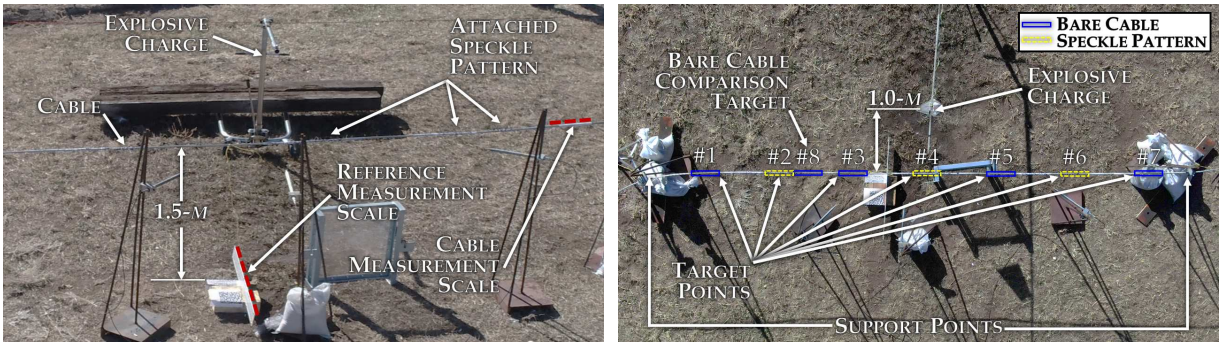
3.4.2 Task 3: Methodology

Experimental Design

The cable used in the experiment was an aluminum conductor steel reinforced (ACSR) six-stranded cable shown in Figure 3.8. This cable had a core radius of 2.25-*mm*, an outer radius of

2-*mm*, and a total area of 91.30-mm^2. The density per unit length is 0.35-*kg/m* and the cable has an elastic modulus of 74.2-*GPa* and a shear modulus of 37.1-*GPa*. The cable was pre-tensioned to a final span length of 7.7-*m* with a mid-span sag of 4-*cm*. During the experiment, the cable was subjected to a point charge generated by a 30.5-*cm* length of det cord wrapped around 39-cm^2 of C1 sheet explosive. This charge was placed 1-*m* from the cable mid-span and directed with a water bag perpendicular to the cable, which was then allowed to free vibrate after the initial impulse. An overview of the complete cable experimental setup is shown in Figure 3.9.

Figure 3.8: The ACSR cable used in the experiment



(a) **(b)**

Figure 3.9: Setup of blast loading on the cable: (a) Perspective view and (b) Plan view from UAS

A DJI Mavic 2 Zoom was used in this test as it provides high-quality video resolution and a stable flight. The UAS was flown about 6-*m* above the pretensioned cable and about 7.5-*m* above ground level. From this height, the ground-sampling distance (GSD) is 2.2-*mm/pixel*. The video was captured at 30-*frames/sec*. The view from the UAS before the charge was detonated is

shown in Figure 3.9(b). A speckle pattern was placed on the ground to provide the most robust and straightforward stationary reference point to track the drift of the UAS. The speckle pattern provides high contrast and texture, ensuring robust keypoints are detectable. To test whether a speckle pattern is required to track the movement of the cable, both the natural features of the cable (i.e., bare cable) and an artificial target (i.e., speckle pattern) were tested. Eight target locations were identified and are highlighted with bounding boxes in Figure 3.9(b). The dashed line and solid line bounding boxes indicate target locations with and without a speckle pattern, respectively. Targets #1-#7 are approximately equidistant apart along the length of the cable. Target #8 was chosen as a comparison to Target#2 as Target #8 was bare cable and Target #2 had an attached speckle pattern. The speckle pattern was printed on paper and wrapped around the cable. Since the paper was thin, lightweight, and matched the shape of the cable, it was assumed that any error imposed by the artificial speckle pattern by stiffening or restricting the movement of the wire was negligible.

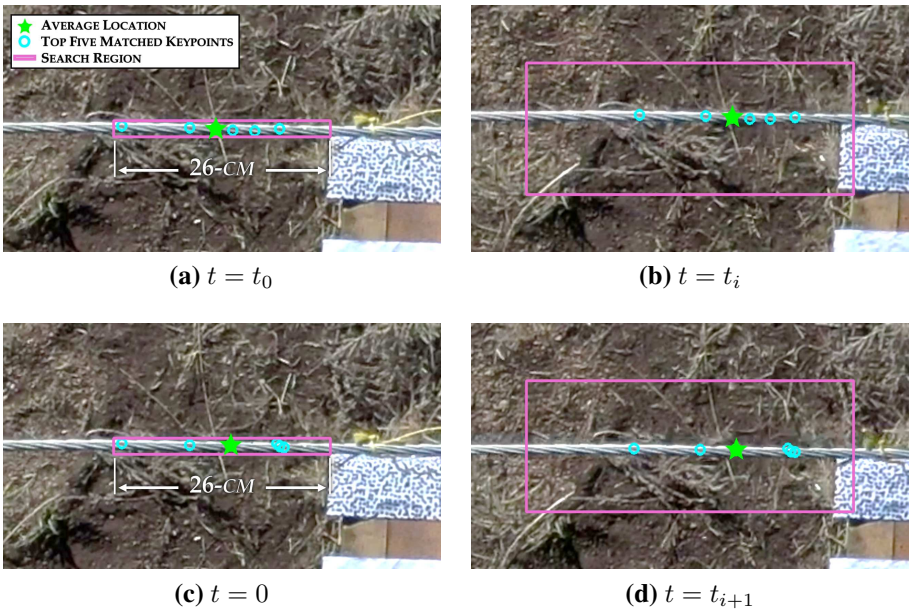
Cable Displacement Measurements using Computer Vision

To measure the displacements of the cable with a portable, non-contact sensor, a UAS-based 2-D computer vision measurement technique was implemented. The workflow of the 2-D measurements first begins with identifying and tracking keypoints over time and applying a scale factor to relate the pixel units to engineering units. The motion of the UAS was also tracked and compensated for in the measurement of the cable. The proposed technique requires minimal user interaction. A user only needs to select the bounding box of the stationary reference and the section of the cable to track.

Keypoints and their associated descriptors are first identified within a small region of interest (ROI, i.e., bounding boxes in Figure 3.9(b)) on the cable through Scale-Invariant Feature Transform (SIFT) [54]. SIFT is adopted as it has proven to be robust in remote sensing [10, 64]. Extracting SIFT keypoints relies on the difference of Gaussians (DoGs) algorithm to extract features within the ROI. The DoGs algorithm performs a Gaussian blur convolution over the ROI and then subtracts the original image from the blurred image. This process is repeated in multiple iterations

performing a Gaussian blur on the previous image and then taking the difference. The local extrema in the DoGs of the different layers are the resulting identified keypoints given by sub-pixel coordinates. The number of SIFT keypoints within an ROI depends on the object's size, resolution, and texture. To further improve the robustness of SIFT, the image was scaled up using linear interpolation to double the number of pixels within the image. The bounding box of the ROIs (bolded rectangles in Figure 3.10 (b, d)) contained twice the number of pixels than originally taken by the camera and provided a larger sampling area. SIFT keypoints were detected on a frame-by-frame ($t = t_i$) basis of the collected video within the ROI. The keypoints in the i^{th} frame ($t = t_i$) that match those detected in the initial frame ($t = t_0$) were then identified based on *brute force matching* between the feature descriptors. Brute force matching checks every identified keypoint in the initial frame ($t = t_0$) against every identified keypoint in thei^{th} frame ($t = t_i$). There were about 150 total keypoints matched between the i^{th} frame ($t = t_i$) and the initial frame ($t = t_0$). The keypoints associated with the top five best matches based on the *distance* metric of the SIFT matched were identified (circle points Figure 3.10) to ensure that the keypoints are robustly matched. The locations of the top five keypoints are then averaged to give the pixel location of the cable at the i^{th} frame ($t = t_i$) (star points in Figure 3.10). The top five matched keypoints may not be the same from frame to frame (as seen by comparing the matched keypoints in Figure 3.10(a-b) with Figure 3.10(c-d)). However, since the selected ROI is relatively small (with a length of 26-*cm*, or about $1/20^{th}$ the length of the cable), it is assumed that the average location of the keypoints within the ROI provides the displacement of the cable at the ROI's center point. This same process was also repeated for the stationary reference target on a frame-by-frame basis to track the displacements of the UAS.

With the pixel location of the cable identified, the pixel units are then transformed into engineering units using an adaptive scale factor. Measurement scales are placed within the frame to relate real-world measurements, highlighted in Figure 3.9(a) as dashed bold lines, to pixels. The measurement scale has a known length and, therefore, is able to relate the pixel lengths to engineering units. To account for the vertical drift of the UAS during flight, an adaptive scale factor



(a) $t = t_0$ **(b)** $t = t_i$

(c) $t = 0$ **(d)** $t = t_{i+1}$

Figure 3.10: Matched keypoints for Target #3: (a-b) matched keypoints between initial frame and frame $t = t_i$, (c-d) matched keypoints between initial frame and frame $t = t_{i+1}$

is updated on a frame-by-frame basis. To measure the pixels of the measurement scales in each frame, *optical flow* is used to track the pixel locations of the measurement scales. The pixel distance between the ends of the measurement scales is then determined. Here, optical flow is used instead of SIFT as it is more efficient in this application. Optical flow tracks features using a simpler *Harris Corner Detector* and only requires one point location at the end of the measurement scale [65–67].

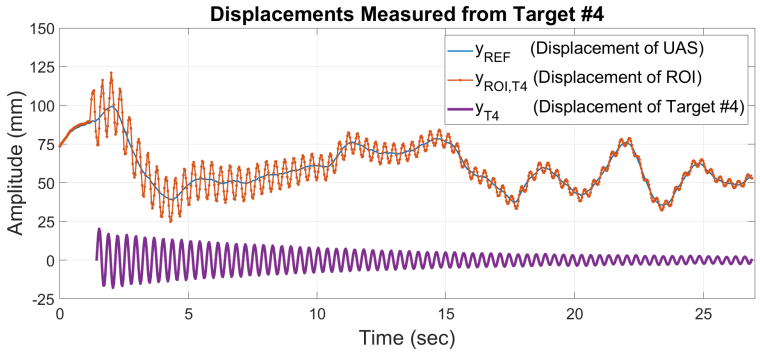
To compensate for the drift of the UAS in the horizontal directions (i.e., parallel to the ground), a stationary reference target is used. A stationary target (e.g., ground point or tower) within the field of view of the camera is identified. The cable's motion is measured with respect to the stationary target. Since the stationary target is on the ground, it is assumed that the movement of the stationary target is due only to the movement of the UAS. Therefore, the drift of the UAS is recovered with respect to the reference target and subtracted from the measurements of the ROI. The true movement of the ROI is found as

$$y_T = y_{ROI,T} - y_{REF} \tag{3.6}$$

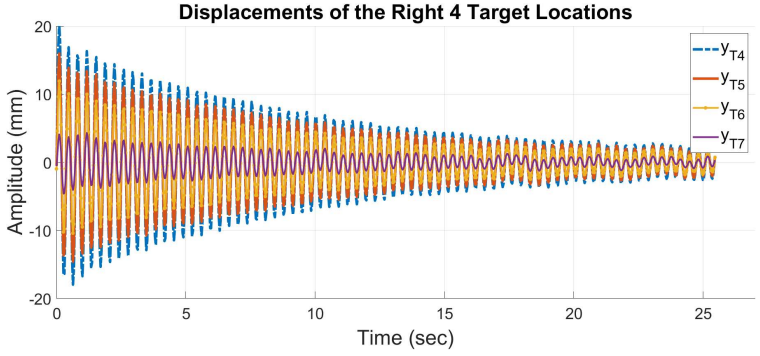
where, y_T is the displacement of the cable at a target location, $y_{ROI,T}$ is the displacement of the cable at a target location relative to the UAS, and y_{REF} is the displacement of the reference target relative to the UAS.

3.4.3 Task 3: Results

A sample of the displacements at Target #4 (Figure 3.9(b)) measured using the proposed technique is shown in Figure 3.11(a). To remove some of the low- and high-frequency measurement noises, a *High Pass* (cutoff frequency: 1-*Hz*) and a *Low Pass* Butterworth filter (cutoff frequency: 8-*Hz*) were applied to y_T. The resulting filtered displacements of Targets #4, #5, #6, and #7 are shown in Figure 3.11(b).



(a)



(b)

Figure 3.11: Filtered measured displacements: (a) All measured displacements of Target #4 and (b) Filtered y_T displacements of the right 4 Target locations

A comparison study was performed to test whether the attached artificial pattern was necessary for the cable displacement measurements. The displacement measurements at Target #2, with an attached speckle pattern, and Target #8, with bare cable (Figure 3.9(b)), are compared in Figure 3.12. The use of bare cable without an attached artificial pattern successfully captures the dynamic properties of the cable and good agreement between the two curves is shown. However, the bare cable data does have slightly more high-frequency measurement errors, as seen from the minor fluctuations in the envelopes of the free decay curve. After performing the aforementioned Butterworth filter, the majority of the high-frequency measurement error was removed. The displacement of the bare cable was slightly larger than that with the speckle pattern. This is expected since the target location of the bare cable section (i.e., Target #8) was slightly closer to the center of the cable. This comparison study has shown that using artificial patterns for displacement measurements of bare ACSR cables in future applications is unnecessary. In this study, since the speckle patterns had been attached in several locations, the measured displacements from the seven target locations containing sections with and without speckle patterns were later used to identify the system parameters.

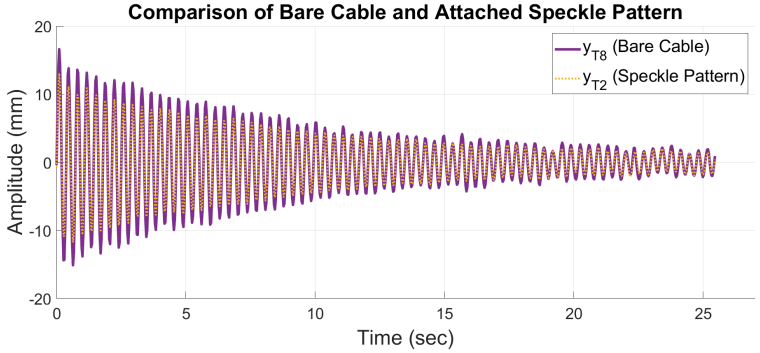
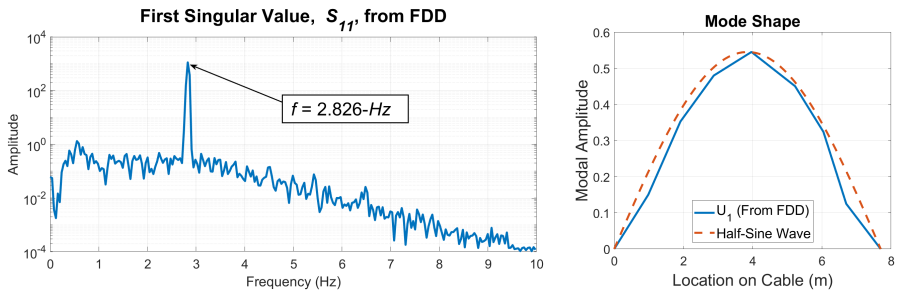


Figure 3.12: Comparison of the dynamic displacements of the cable between an attached speckle pattern (y_{T2}, dotted line) and bare cable (y_{T8}, solid line) near Target #2.

A Fourier analysis showed that only the first mode dominates the cable response. Hence, the simple *Logarithmic Decrement Technique* (LDT) was applied for system identification using the measured displacements from the seven target locations (Figure 3.9(b)). Using signals within a



(a) **(b)**

Figure 3.13: System identification of cable: (a) First Eigenvalue from FDD and (b) First mode shape using FDD

moving window with window sizes varying between 20-39 *peaks*, the natural frequency and the damping ratio averaged over all windowed signals were 2.83-*Hz* and 0.00523, respectively.

The mode shape was identified using Frequency Domain Decomposition (FDD). Specifically, the cross-power spectral density matrix, $\mathbf{S_{ij}}(\omega)$ was estimated by

$$\mathbf{S_{ij}}(\omega) = \hat{f}\left(y_i\right) * \hat{f}\left(y_j\right)^* \tag{3.7}$$

where $\hat{f}$ is the Fourier Transformation and * is the conjugate, y_i and y_j are the response signals from ith and jth target location, ω is the frequency. Then, singular value decomposition was performed, defined as

$$\mathbf{S_{ij}}(\omega) = \mathbf{USV}^H \tag{3.8}$$

where ^{H} is the conjugate transpose. The singular values (i.e., the diagonal elements of $\mathbf{S}$) and singular vectors, (e.g., $\mathbf{U_i}$) were obtained. The first singular value is shown in Figure 3.13(a). From the figure, the natural frequency is identified as 2.83-*Hz*, which is in good agreement with that obtained from LDT. The mode shape is the first singular vector corresponding to the natural frequency, shown in Figure 3.13(b). Note that the damping ratio was not identified using FDD, as the signal was only about 25-*sec* long and the frequency resolution of the power spectral density was relatively low, which may result in amplified bandwidth and an overestimation of the damping ratio.

The last step was to compare the system identification result using measured displacement by the proposed computer vision-based technique to the results of both analytical and finite element models. The frequencies of sagged cables have been investigated in analytic detail by Irvine and Caughey [68]. The lowest natural frequency of a sagged cable is the motion of the cable in a direction perpendicular to the original plane in which the undeformed cable rests. This frequency can be computed analytically when there is no axial-torsional coupling of the cable displacements and is a function of the cable span, the mass density, and the sag. However, for braided wires, there can be slight adjustments to the frequencies based on the levels of coupling where the analytical formulas may not hold [62, 69]. In this case, a finite element (FE) model can be used where a geometrically nonlinear quasi-static analysis is first performed on the cable under gravitational load, and then a free vibration analysis about this equilibrium configuration is completed. Both scenarios were used to compute the dominant frequency of 2.77-Hz, indicating that the coupling does not play a significant role for this mode. This frequency is in excellent agreement with the experimental value of 2.83-Hz with a 2.1% difference. In addition, the exact and finite element mode shapes for the fundamental mode are the half-sine wave, which is also in excellent agreement with the FDD results.

Conclusions

A new technique was presented to directly measure dynamic displacements of cables using a single camera attached to a UAS. The methodology was tested in a real-world environment where a pre-tensioned cable was excited with a small explosive charge. The displacement time histories of seven target locations were obtained, from which the natural frequency, damping, and mode shape were identified.

The primary findings of this study are as follows:

1. The natural frequency matched an FE simulation with a 2.1% difference and the UAS-measured mode shape is in excellent agreement with the numerical result.

2. The proposed technique has the ability to extract and identify dynamic properties of power lines within an electrical grid or cables attached to a structure or tower.

3. The bare, ACSR six-stranded cable provided enough texture and contrast to not require any additional artificial targets on the cable. An operator is therefore able to quickly and more easily fly to a location with a UAS without the need to attach a special target.

In addition to these findings, there are two key considerations for successfully implementing this technique:

1. In the present experiment, the cameras were not calibrated prior to the blast test. It was assumed that since the majority of the target locations were within the center of the frame, the effect of camera distortion is minimal; however, it is recommended to perform a camera calibration before taking the dynamic displacement measurements in general applications.

2. A stationary target must be within the camera's frame to track the movement of the UAS during its flight. Although the displacement is minimal, without accounting for the UAS drift, the errors could be significant. A stationary target could be a background point, the cable's connection point, or a support tower/pier.

3.5 Task 4: 3C Dynamic Displacement Measurements: Small-Scale

3.5.1 Task 4: Motivation

Understanding the dynamics of structures is critical for evaluating long-term structural performance and decision-making regarding the maintenance and operation of the structure. Quantifying the displacement of structural vibration is an important means to evaluate the dynamic performance of structures under various dynamic loading such as winds, traffic, impact loading, etc. Once the full three-component (3C) dynamic displacement of structures is measured, system identification methods can be applied to estimate the dynamic properties of the structure. If the time series is

stationary, the conventional system identification (SI) methods (e.g., random decrement technique, frequency domain decomposition, stochastic subset identification) can be used [70–73]. On the other hand, if the time series are non-stationary, more advanced SI techniques (e.g., time-frequency domain approaches) can be used [74–76]. Additionally, the 3C displacement measurements can also be useful for the development of various data-driven approaches for SHM [77–81]. Lastly, the identification of 3C displacements is also a vital step for the validation of FE models and digital twins. Currently, these measurements rely on accelerometers attached to the structure which only measure acceleration; displacement measurements are not possible due to the double integration required. Logistical issues impede the implementation of accelerometer sensors on a wide scale (i.e., wiring, time synchronization if wireless, placement, and inability to relocate).

With the recent advancements of UASs, research on using optical sensors on UASs to track the displacements of a structure in either the two-component (2C) planar directions (the plane that is perpendicular to the camera) or the one-component (1C) depth direction (the out-of-plane direction or the distance from the camera to the structure) has been explored. With the assumption that the out-of-plane displacement of civil infrastructure is minimal, 2C planar displacements of structures can be measured with optical sensors on UASs [82, 83]. In Section 3.4, the 2C displacements of a cable were measured using a UAS and applying an adaptive scale factor to relate pixel units to engineering units. Kalaitzakis et al. implemented a painted speckle pattern on a concrete girder to measure the strain during a four-point loading test with an additional painted speckle pattern on a stationary reference [83]. Other studies can measure the out-of-plane displacement of structures [84–86]. Garg et al. used a laser Doppler vibrometer, which accurately measures the reflection of a high-frequency wave, to calculate the out-of-plane displacement of an object [84, 85]. Catt et al. implemented passive stereo vision by mounting two cameras at a known distance apart to measure the out-of-plane deformations of a deformable board with a random speckle pattern [86]. The proposed algorithm processes the speckled pattern placed on the board to measure the 1C depth; however, the drift of the UAS was not compensated resulting in the 2C displacement of the deformable board not being realized. Although the aforementioned techniques

using painted speckle patterns can provide accurate measurements within the speckled area (such as [86] or [83]), they do not take advantage of the entire field of view (FOV) of the camera, and implementing these techniques in the real-world may be challenging due to the large-scale of the structures and workforce required to install the detailed patterns. Yoon et al. proposed a 2C planar UAS-enabled measurement methodology without a speckle pattern by identifying, matching, and tracking feature points in the background of the video to estimate the camera's pose and recover the 3C translation and rotation of the UAS; however, the proof of concept still required LED lights as targets [87]. UASs have also been used to identify the vibration modes of a structure without correcting for the movement of the UAS [88]. The existing studies have not attempted to measure the full 3C displacements of a structure using UAS-enabled sensors without the aid of a painted speckle pattern, which can be critical for certain types of structures such as long-span bridges and cables.

This task is a preliminary exploration of using stereo vision on a UAS to collect 3C dynamic displacement measurements. The purpose was to conduct a small-scale proof of concept and explore the limitations and challenges of UAS-enable stereo vision measurement. Although the technique in this task was successful, the methodology is not scaleable for measuring distance (i.e., the distance from the structure to the camera) longer than about 1.5-*meters*. These limitations and challenges are discussed at the end of this section and Task #5 (Section 3.6) explores a scalable and updated methodology to measure full-scale structures. The results of this small-scale proof-of-concept which are published in [60] are presented. Then, the limitations of the preliminary framework are discussed.

Objectives

1. Explore the practicality of using optical and infrared sensors with an infrared projector within the RealSense D435 to measure 3C dynamic displacements

2. Explore using the RealSense D435 on a UAS and find limitations

3. Explore the challenges and limitations of developing a methodology to perform full-sized 3C dynamic displacement measurements

3.5.2 Task 4: Methodology

The computer vision algorithms of the proposed double-faceted 3C displacement measurement technique will be introduced in this section. An initial overview of the workflow is presented in Figure 3.14.

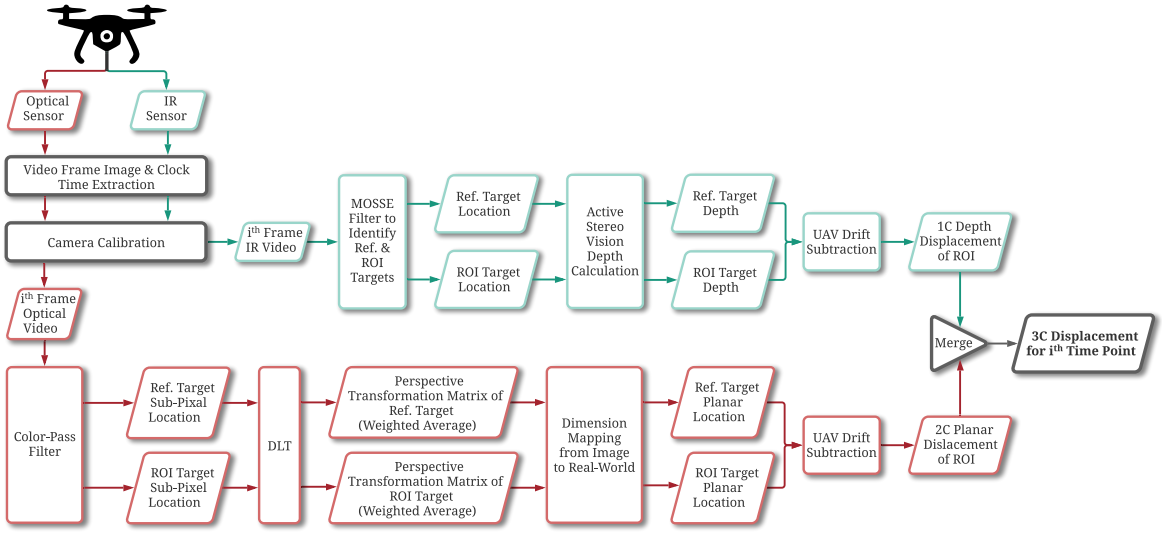


Figure 3.14: Proposed methodology overview

2C Planar Measurement

To measure the 2C planar movement of the ROI, first, the camera needs to be calibrated to ensure that the physical dimensions measured by the cameras are free of tangential and radial distortion errors caused by the lens. Camera distortion correction is a well-studied problem within the computer vision field [89–93], and many off-the-shelf software/toolboxes are available. In this study, both a proprietary calibration software developed specifically for the RS optical and IR sensors included in the Realsense SDK 2.0 and an open-source MATLAB camera calibration toolbox [94] were tested. A grid pattern target and a total of 14 pictures were used in the calibration.

Both tools achieved similar performance. In the following displacement measurement study, only the calibration results using Realsense SDK 2.0 were used. Once the camera is calibrated, it can be used for multiple surveys; however, if the camera is jarred hard enough so that some structural components of the lens or camera are moved or affected, the camera will need to be re-calibrated. Also, note that changing temperatures can cause errors in the camera calibration. For instance, heating and cooling the camera by 10°C can cause up to a pixel image drift with larger pixel drift from larger temperature changes; therefore, it is recommended that the camera stays at a relatively constant temperature after calibration or re-calibration is needed [95].

With the calibrated sensors, the UAS hovers next to a structure/object, and the optical sensor records a video of the object moving in the *x*-*y* plane (Figure 3.15). In this study, the optical sensor inside the RS sensor is used. This optical (or RGB) sensor is a 2.729 $\times$ 1.550-*mm* Complementary Metal Oxide Semiconductor (CMOS) sensor with a rolling shutter capable of capturing 1920 $\times$ 1080-*pixel* at 30-*fps* (or a faster sampling frequency at a lower resolution), and a 1.93-*mm* focal length; the horizontal angular FOV of the camera is 69.4° (Table 3.4). The camera attached to a hovering UAS captures image frames that contain the targets of both the ROI on the vibrating object (e.g., a bridge under wind excitation) and that of the stationary reference target (Figure 3.15). The maximum distance between the stationary reference and the moving object that allows both in the same frame is dictated by the FOV of the optical camera, the UAS flight height, as well as the level of UAS horizontal drift. The UAS flight height is controlled by the desired level of accuracy for depth measurement which will be discussed later in this section, while the drift of UAS is dependent on a particular UAS system and wind conditions. The motion of the UAS is recovered by tracking the relative movement of the stationary reference target with respect to the UAS; while the relative motion between the UAS and the ROI is extracted by tracking the movement of the ROI. By adding the movement of the ROI with respect to the UAS, $\mathbf{X}_{ROI}$, to the movement of the UAS with respect to the ground, $\mathbf{X}_{UAS}$, the true movement of the ROI, $\mathbf{X}_{True}$, is found, as shown in Equation 3.9.

$$\mathbf{X}_{True} = \mathbf{X}_{ROI} + \mathbf{X}_{UAS} \tag{3.9}$$

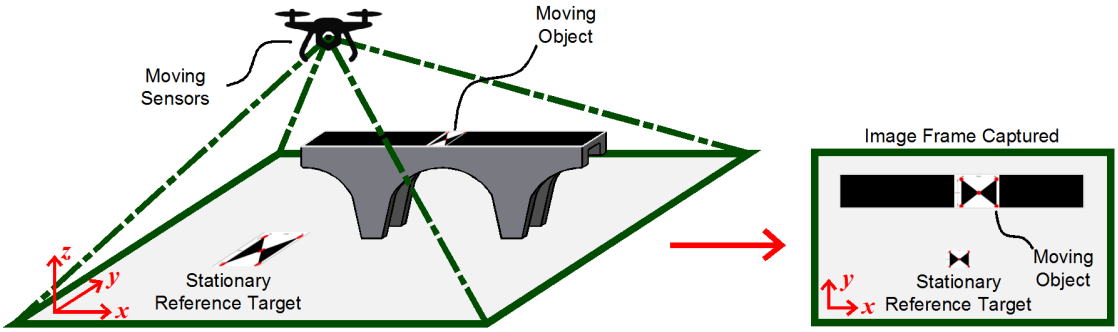


Figure 3.15: Concept of 2C planar measurement

Table 3.4: Camera parameters

	Sensor Type	Sensor Dimensions (*mm*)	Shutter	Focal Length (*mm*)	Horizontal FOV (*deg.*)	Video Resolution ($pixel \times pixel \times fps$)
RealSense Optical	CMOS	2.729×1.550	Rolling	1.93	69.4	$1280 \times 800 \times 30$ $848 \times 480 \times 90$ $840 \times 100 \times 100$
RealSense Infrared	CMOS	3.90×2.45	Global	1.93	86.0	$1920 \times 1080 \times 30$ $960 \times 540 \times 60$

A robust target identification algorithm is developed to simultaneously track the targets on the ROI and those on the stationary reference target. To promote the fast and simple implementation of the proposed framework, a color detection-based technique was developed to robustly and accurately find the center points of the red-circle-target (Figure 3.16a) used in the experiments. The red-circle-target also provides a scale to help accurately transform the image coordinates into real-world coordinates in the next step. Note that it is not necessary to use this specific type of reference target; only measurable targets and identifiable reference points are needed. The idea of this approach is to first detect the target using its color (red in this case). To accomplish this, the image is converted from *RGB* to a *Hue*, *Saturation*, *Value* (HSV) color model. The *Hue* represents the specific color value ranging from 0-255, with 0, 85, and 170 representing red, green, and blue, respectively, while the *Saturation* and *Value* represent how close the color is to black or white. Using the HSV color model is more advantageous for color identification in the images compared to using an RGB color model because only one value (versus the three values in the RGB model),

e.g., the Hue value, represents the color present. Moreover, since the reference target is on a white background (Figure 3.16a), the red color is filtered easily using a small range of Hue values that encompass the scale points on the reference. Since certain Hue values are the only color allowed to pass, the filter is called a *color-pass filter*. The color-pass filter results in the binary image shown in Figure 3.16b. Then, the contours in the binary image are found by a contours detector developed in [96]. The center is estimated by taking an average of the pixel locations inside the contour. Knowing the general location of each reference point in the image (e.g., upper left corner, bottom right corner, etc.), a search was performed in each quadrant of the image to distinguish the different reference points. The result is shown in Figure 3.16c. The major advantages of the proposed color-pass filter include 1) it provides subpixel detection accuracy and 2) it is more computationally efficient over other target detectors such as template matching that requires a reference image and an exhaustive search and Hough Transforms that needs additional higher-level processing. The proposed color-pass filter technique is generalizable for field application when applied in conjunction with other object-tracking techniques. For example, one may first use an object tracker, such as the MOSSE filter [97] or CRST [98] to track the target's general location. Then, the background is masked out, and the color-pass filter is applied to the masked image to detect the reference points with high accuracy. Note that when used solely, object trackers like the MOSSE filter do not provide the subpixel accuracy needed for displacement measurements [99], which highlighted the value of the proposed color-pass filter.

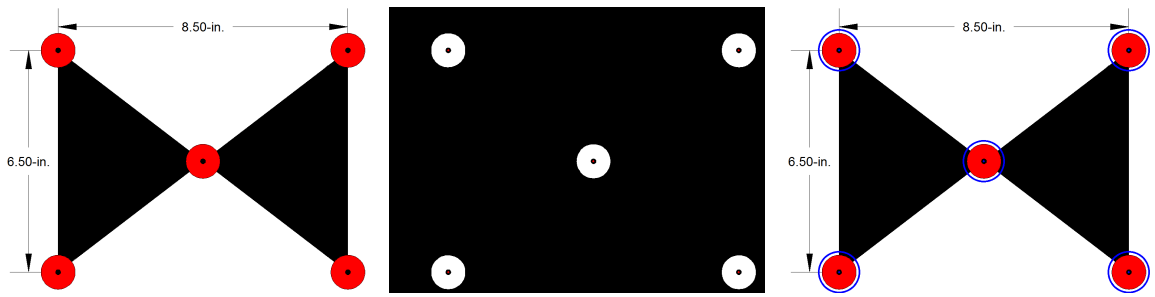


Figure 3.16: Reference target: (a) Reference and red scale points used; (b) Red-pass color filter of targets; and (c) Identified red targets

After the reference targets are successfully found, the actual amplitude of the displacement needs to be recovered by mapping the pixels on the image to the real-world dimensions through a direct linear transformation (DLT). Implementing DLT ensures that 1) image warping due to non-perpendicular camera angles is eliminated to enhance measurement accuracy in the plane, and 2) each pixel is scaled to engineering units (e.g., in millimeters). Item 1) is needed to ensure there is no warping of the image. In real-world applications, the video can be captured at a nearly perpendicular angle; however, small misalignment typically exists due to the random drifts of the UAS. The DLT corrects the image for non-perpendicular (e.g., the small misalignment) and skewed camera angles to achieve a high measurement accuracy. On the other hand, item 2) allows real-world measurements to be taken directly from the image. To begin the DLT, the relationship (or perspective transformation matrix) between a known reference point in the real world in space coordinates, $\mathbf{X} = [X, Y, 1]^{\mathrm{T}}$, and its location in pixel points in image coordinates, $\mathbf{x} = [x, y, 1]^{\mathrm{T}}$, must be known. Note that the image observation points (x, y) were obtained from the pre-processed image with radial and tangential distortions corrected in the camera calibration process. By knowing both image and space coordinates, the DLT is implemented to solve for the perspective transformation matrix, $\mathbf{A}$, with respect to the linear system,

$$\mathbf{X} \sim \mathbf{A}\mathbf{x} \tag{3.10}$$

where $\sim$ denotes equality up to a scale. The resulting A is used to transform every pixel of the image by the following,

$$\mathbf{P}_{DLT}(X, Y) = \mathbf{P}_{Original}\left(\frac{A_{11}x + A_{12}y + A_{13}}{A_{31}x + A_{32}y + A_{33}}, \frac{A_{21}x + A_{22}y + A_{23}}{A_{31}x + A_{32}y + A_{33}}\right) \tag{3.11}$$

where, $\mathbf{P}_{DLT}(X, Y)$ is the transformed image with real-world coordinates at point (X, Y) and $\mathbf{P}_{Original}(x, y)$ is the original image in image coordinates. As a note, the transformation is only viable on the z-direction plane where the reference targets are placed; if the z displacement (the distance from the camera to the ROI) is significant, errors can occur in the measurement. Therefore, for the most accurate measurements, it is best that DLT is performed on a frame-to-frame

basis to eliminate the impact due to camera displacement in z-direction. However, updating the perspective transformation matrix, $\mathbf{A}$, with every frame introduces noise in the measured signal due to the subpixel location errors of the reference points. To reduce this noise, it is recommended to take a weighted average of $\mathbf{A}$, for each frame of the video with,

$$\mathbf{A}_i^* = \eta \mathbf{A}_i + (1 - \eta)\mathbf{A}_{i-1} \tag{3.12}$$

where $\mathbf{A}_i^*$ is the perspective transformation matrix used in the DLT for the i^{th} frame, $\mathbf{A}_i$ is the calculated perspective transformation matrix of the i^{th} frame, and $\mathbf{A}_{i-1}$ is the previous perspective transformation matrix. Through testing, it was found that $\eta = 0.125$ produced smooth and accurate results. After the raw images are transformed through the process of camera calibration and DLT, the distances between points on the image represent the real-world dimensions, and the amplitude of the displacement in the real world is finally recovered. One should note that the DLT is very sensitive to the exact subpixel location that is used for scaling. Therefore, it is important to ensure that each subpixel location extracted from the image corresponds to the exact location of the reference point on the real-world scale for accurate measurements. As previously mentioned, the color-pass filter developed in this study robustly identifies the subpixel locations in the image.

1C Depth Measurement

The 1C depth measurement is also realized by the RS sensor. Similar to the 2C measurement facet discussed in the previous section, an algorithm is developed to compensate for the movement of the UAS to obtain the true motion of the object. In the following, the concept of the proposed RS sensor-based 1C displacement measurement is introduced in detail.

The RS depth sensor used in this study has a 360-*mW* average IR laser projector with a nominal wavelength of 850-*nm*$\pm$10-*nm*. This projector emits a virtual speckle pattern, which is not visible to the human eye, onto the FOV of the camera. Two IR 3.90 $\times$ 2.45-*mm* CMOS sensors with a global shutter and wavelength range of 400 - 865-*nm* are also included in this sensor set. Each IR camera can capture video at 1280 $\times$ 720-*pixel* at 30-*fps* (or a faster sampling frequency at a

lower resolution) and 1.93-*mm* focal length; the horizontal FOV of the sensor is 86.0° (Table 3.4). The baseline between the two IR cameras is 50-*mm* and is stiffened to allow for more accurate measurements; the baseline for the RGB sensor and IR projector is also known. A schematic of the sensor set-up is shown in Figure 3.17a. By projecting a seemingly random, but known, IR speckle pattern onto the objects (Figure 3.17b), each speckle is able to be matched between the two IR cameras. With the matched speckles and a known baseline between the two IR cameras, the angle from each camera to the speckle is calculated, and the distance from the speckle to the camera is found. Using active stereo vision, every available pixel in the image is transformed to the depth measurements in real-time, and the depth from the RS sensor to the ROI is found. This calculation is embedded onboard the RS sensor.

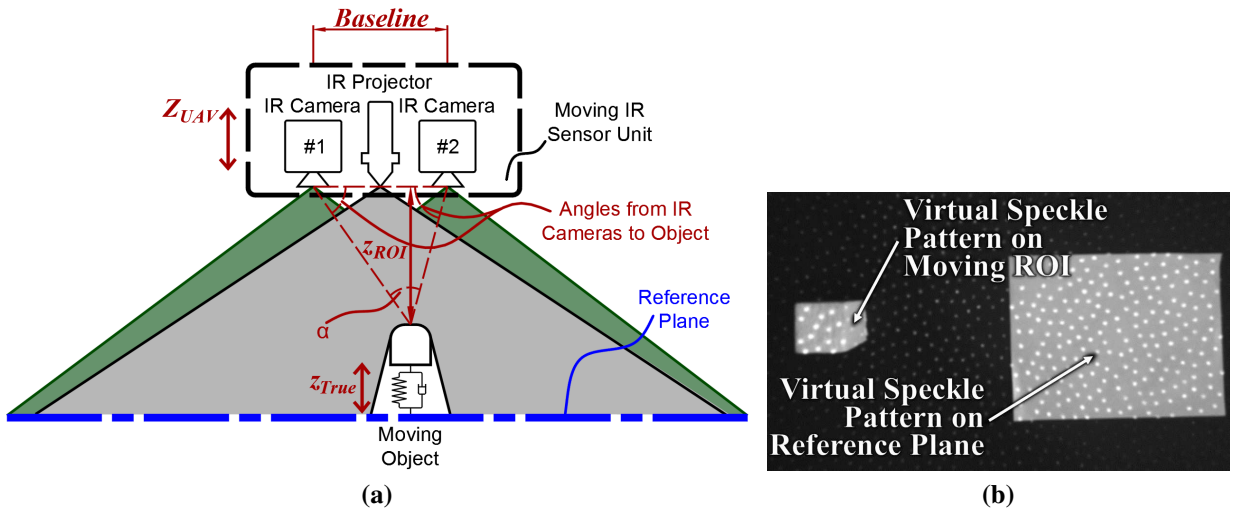


Figure 3.17: Concept of 1C depth measurement: (a) Schematic plot; and (b) View of virtual speckle pattern from RS IR camera

Since the IR projector covers the entire FOV of the camera, the depth throughout the entire image is solved, allowing the full-field displacement measurement. To enhance the measurement accuracy and reduce depth measurement noise, a small area of pixels (within a one cm^2 range) on the ROI can be selected, averaged, and considered as a point to represent the ROI. Since both the object and the UAS are moving in the two horizontal directions (x- and y-directions), the ROI and the stationary reference must also be tracked in the IR video to allow for the same point measure-

ment. To accomplish this, a MOSSE filter [97] was used to track the areas of interest in the *x*- and *y*-directions. Although a MOSSE filter is not recommended to be used solely in the 2C planar measurement due to the lack of subpixel precision, it is admissible for the 1C depth measurement since the average depth for speckle points in a small area is taken. As a note, shadow effects of the IR projector or cameras may limit the performance of full-field measurement, which is a common drawback to stereo vision; measuring the displacement from the top surface of a structure, like in this study, alleviates this issue. Using this approach, the relative dynamic displacements between the UAS and the object, as well as that between the UAS and the reference plane are measured. Then, the true motion of the object is extracted by removing the motion of the UAS itself.

Description of Data Sets

In the experiments, the dynamic displacement of the structure in horizontal directions is produced by a small shaking table with a sine wave input. The signal of the displacement used as the ground truth in the validation was constructed based on the sine wave function (Equation 3.13). The amplitude, A, was found by manually measuring the diameter of the spinning motor of the shaking table. The frequency, f_0, of the true displacement, was found through a Power Spectral Analysis of the accelerometer data using Fast Fourier Transform. Lastly, the phase angle, φ, was determined by minimizing the RMS error between d (Equation 3.13) and results from the measuring techniques tested, as direct data synchronization between the input signal of the shake table and the measurement devices was not an option.

$$d = A \cdot \sin\left(2\pi f_0 \cdot t + \varphi\right) \tag{3.13}$$

Based on the analysis of the previous study [100], the typical natural frequencies of a variety of civil structures and bridges are between 0.088-*Hz* (for the world's tallest building, Burj Khalifa, in the United Arab Emirates) to 7.6-*Hz* (for a 15-*m* beam bridge), and the average of the reported natural frequencies (without the beam bridges which have a relatively large natural frequency) is 0.585-*Hz* [100]. According to the classic Nyquist–Shannon sampling theorem, the sampling

frequency needs to be at least twice the maximum frequency of the vibration to faithfully record the dynamic response. Thus, a sampling frequency greater than 20-*Hz* is required for monitoring civil structures. The maximum sampling frequency (e.g., video frame recording rate or inverse of temporal resolution) for typical low-cost consumer-level cameras is about 30-90-*Hz* (fps). So, the requirement in temporal resolution for structural monitoring can be easily achieved using low-cost consumer-level cameras. The input frequencies of the excitation of the shaking table are set at 0.5, 1.0, and 2.0-*Hz* to test the applicability of the proposed techniques for civil structures. The preliminary 2C and 1C experiments tests were conducted at a sampling frequency of 30-*Hz* (or 30-*fps*) and the maximum frequency of vibration is 2-*Hz*. The proof of concept experiment was conducted at a sampling frequency of 60-*Hz* (or 60-*fps*) and the maximum frequency of vibration is 4.5-*Hz*. Therefore, the assigned sampling frequencies (or *fps* camera recording rate) is sufficient. Note that for the applications that require a higher sampling rate, a maximum rate of 90-*fps* can be achieved with the RS sensor by reducing the resolution of the video.

3.5.3 Task 4: Results

The proof of concept experiment was designed to test the proposed methodology in a small-scale laboratory setting. The double-faceted technique was used to measure the full 3C displacement of the model structure. In this section, preliminary analyses of 2C and 1C measurements are presented, respectively. Then, a proof of concept test on a cantilever structure is demonstrated.

Analysis of 2C Planar Measurement

First, to ensure the accuracy of the 2C planar measurement technique, an accuracy assessment of the camera calibration and DLT was performed on the RS optical sensor. A grid pattern was photographed at an inclined angle, and four measurements of the grid corners were taken. Knowing the dimensions of the grid to be 127-*mm* on each side, the DLT was applied to transform the image, and the distances on the grid were measured. Figure 3.18a is the original distorted image; the error in millimeters in each direction is labeled on the figure. Figure 3.18b has been corrected for the tangential and radial distortion only and Figure 3.18c has only the DLT applied without correcting

the lens distortion. Lastly, Figure 3.18d shows the image that was firstly corrected for distortion and then transformed with the DLT (the methodology used within the proposed technique). From the comparison of sub-figures in Figure 3.18, the overall least error is achieved from the undistorted, and DLT transformed image (Figure 3.18d). The image with only the DLT applied (Figure 3.18c) has a negligible error in the *y*-direction, but a relatively larger error in *x*-direction. On the other hand, the error from the undistorted and DLT-transformed image is consistent in both directions. This remaining error in Figure 3.18d is caused by the small errors in the subpixel location of the four corner reference points since the DLT used to transform the image is sensitive to this subpixel location. Nevertheless, the measurement accuracy in Figure 3.18d is within a sub-millimeter range with a percent error of less than 0.02%.

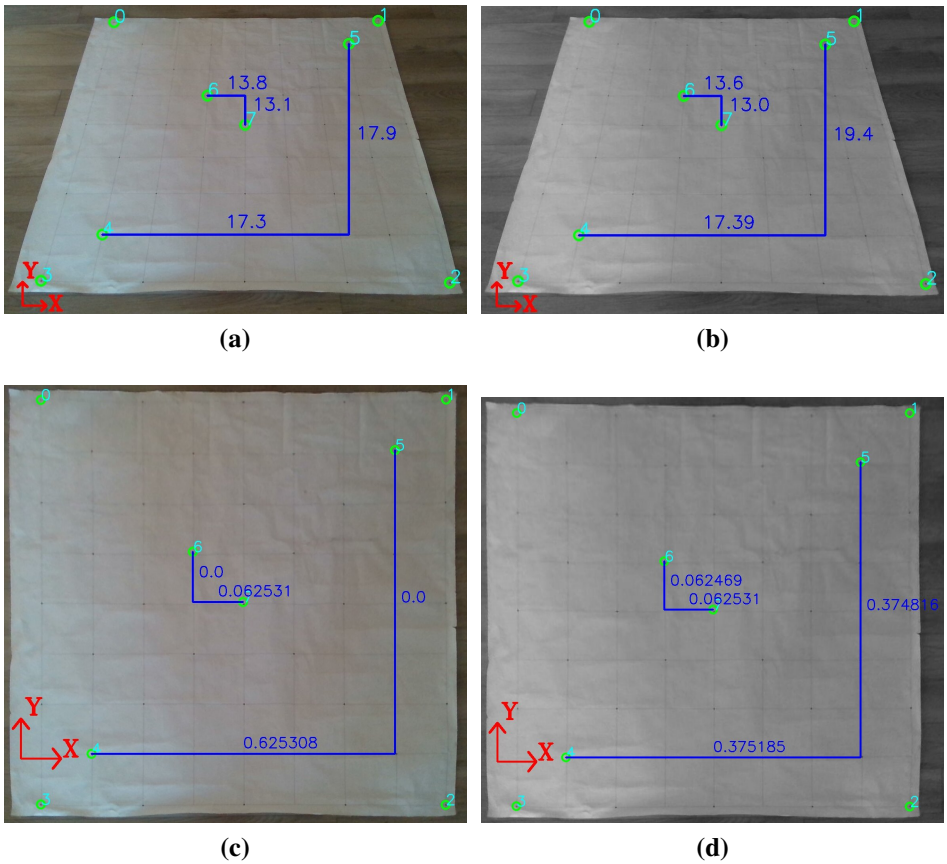


Figure 3.18: Camera calibration results of the RS optical camera (the blue numbers are the error from the calculated distance to the ground truth distance in *mm*): (a) Original image; (b) Undistorted image corrected in post-processing, (c) Image with only DLT applied, and (d) Image with both distortion correction and DLT applied

With the Camera Calibration and DLT showing minimal error, the efficacy of the proposed 2C planar measurement technique was tested. The RS sensor was suspended above the shaking table and moved slowly (on the order of 0.5-m) to simulate the sensor movement when equipped on a hovering UAS as shown in Figure 3.19. The target to be measured was secured onto the shaking table. A stationary reference target was attached to the ground to track the movement of the RS sensor. The input frequencies of the shaking table were 0.5, 1.0, and 2.0-Hz and the amplitude varied between 15.7-mm and 35.6-mm. The video with the appearance of both the target of interest and ground reference was recorded with 480 $\times$ 640-*pixel* at 30-*fps*.

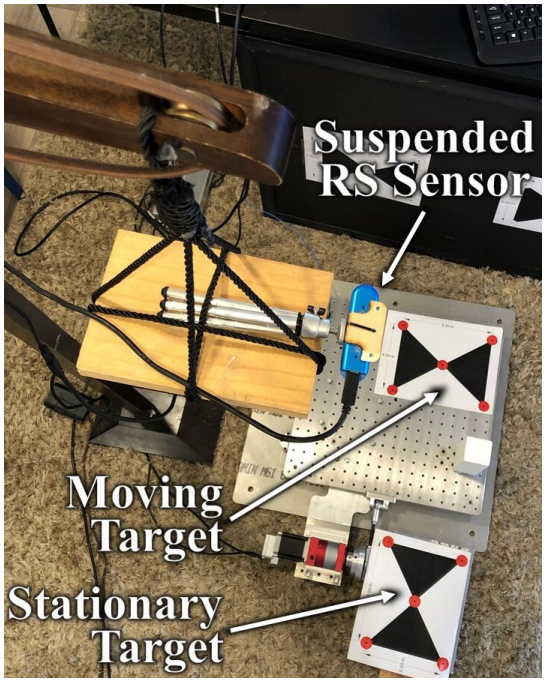


Figure 3.19: Overview of non-stationary 2C planar measurement met-up

The results of the tests are listed in Table 3.5 with the sample time history of two measurements plotted in Figure 3.20. The RMS error of vibration amplitude is on the order of a couple of millimeters when compared to the ground truth and is consistently small throughout the three input frequencies and two input amplitudes tested. The frequencies of the signals measured by the proposed technique are very close to those of the ground truth. The time-varying absolute error shown in Figure 3.20 is caused by the randomness of the perspective transformation matrices of

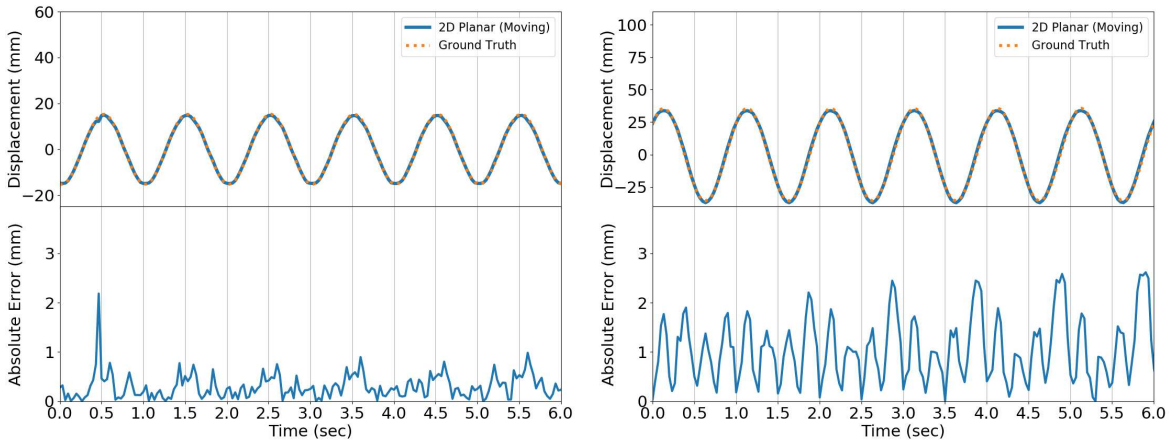


Figure 3.20: Sample displacement measurement results from *non-stationary* RS optical sensor with time history at top and absolute error at bottom: (a) $f_0 = 1.00$-*Hz* and A = 15.7-*mm* and (b) $f_0 = 1.00$-*Hz* and A = 35.6-*mm*

the DLT (as the DLT is sensitive to the sub-pixel location of the reference target). Note that the results shown are the measurement in x-direction. The measurement accuracy in the y-direction is similar, while the results are omitted for conciseness.

Analysis of 1C Depth Measurements

The measurement accuracy of the 1C dynamic displacement using the RS depth sensor placed in a *stationary* position was firstly tested to lay the foundation for 1C displacement with the *non-stationary* RS sensor equipped on a UAS. The experimental setup is shown in Figure 3.21. The amplitude of the displacement remained the same in this test. The RS sensor was set up perpendicular to the rigid structure at a distance of approximately 0.5-*m*. The IR image resolution was 1280 $\times$ 720-*pixel* at 30-*fps*. The three frequencies of the test found through a Power Spectral Analysis of the time series are compared with those obtained from the accelerometer and the results are listed in Table 3.6, showing an exceptionally good agreement. Additionally, the result of measured time history, absolute error (e.g., the absolute value of the difference between the measured displacement and the ground truth), and power spectral density from two sample tests with different excitation frequencies are shown in Figure 3.22. It is seen from the results that the error is at a

Table 3.5: Results from 2C non-stationary planar validation test

Ground Truth						
Frequency (*Hz*)	0.495	0.495	1.004	1.004	2.007	2.007
Amplitude (*mm*)	15.7	35.6	15.7	35.6	15.7	35.6
1C Depth Measurement						
Number of Samples	361	232	186	203	211	255
Total Time (*sec*)	12.01	7.70	6.17	6.74	7.00	8.47
Sampling Frequency (*Hz*)	30.06	30.11	30.14	30.13	30.12	30.10
Frequency Measured (*Hz*)	0.501	0.502	0.978	1.040	2.008	2.014
Validation Against Ground Truth						
RMSE (*mm*)	2.280	2.661	1.353	2.286	2.500	1.553
Mean Absolute Error (*mm*)	1.389	2.116	0.699	1.856	1.841	1.318
R^2 Score	0.998	0.997	0.998	0.998	0.998	0.995
Error of Frequency	1.20%	1.40%	2.62%	3.52%	0.005%	0.35%

relatively constant level throughout the varying input frequencies. The mean absolute error of vibration amplitude is in the sub-millimeter range. This result suggests that a stationary RS sensor can achieve a high displacement measurement accuracy.

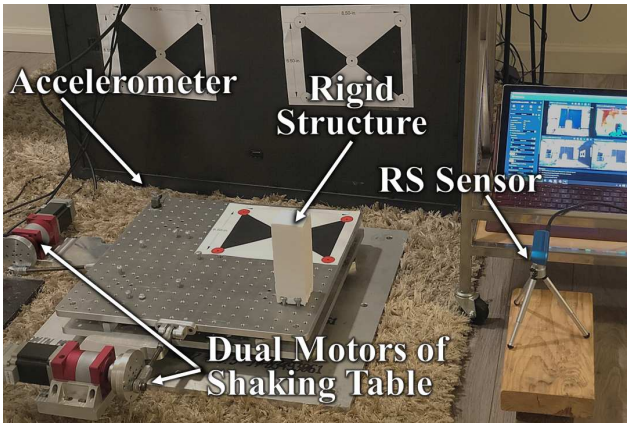
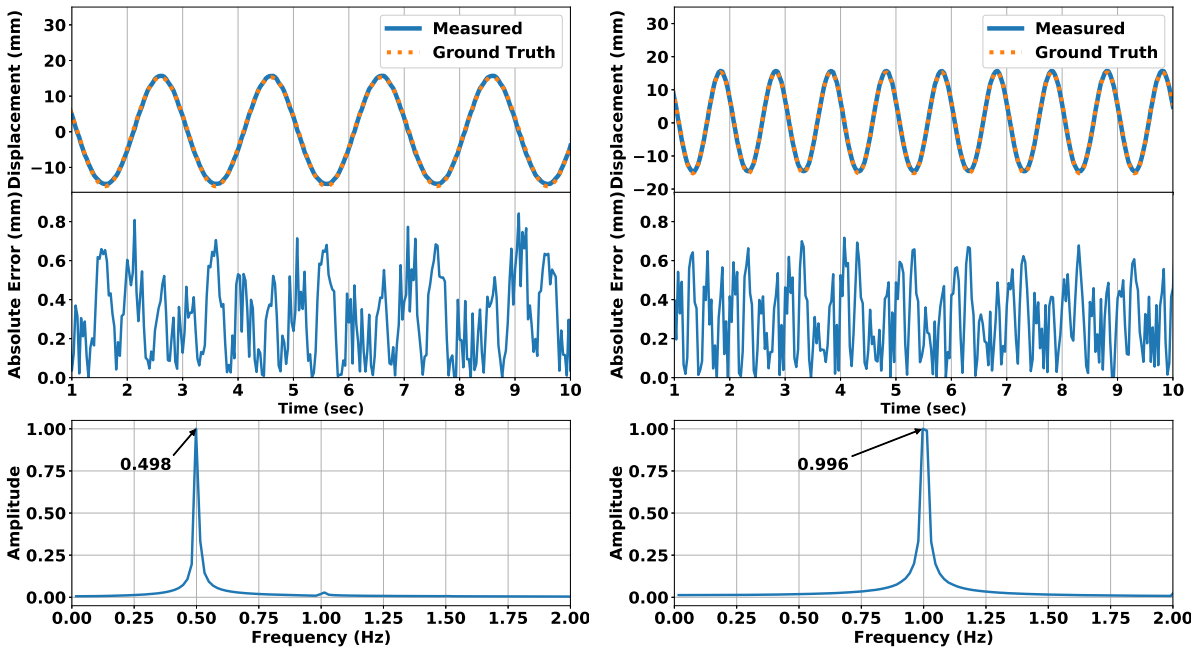


Figure 3.21: Set-up of *stationary* 1C validation test

Inherently with the RS depth sensor, there is measurement noise that increases as the distance from the ROI to the sensor increases. The sensor works best at close range, and an experiment was conducted to test the noise level and accuracy of the sensor at various distances. In the Realsense SDK 2.0, Intel provides a *Depth Quality Tool* to measure the noise level of each measurement



(a) (b)

Figure 3.22: Sample displacement measurement results from *stationary* 1C RS depth sensor with time history at the top, absolute error in *mm* in the middle, and power spectral density at the bottom: (a) $f_0 = 0.50$-*Hz* and (b) $f_0 = 1.00$-*Hz*

Table 3.6: Results of stationary 1C RS depth sensor

Ground Truth			
Frequency (*Hz*)	0.495	1.004	2.007
Amplitude (*mm*)	15.7	15.7	15.7
1C Depth Measurement			
Number of Samples	1749	1749	1749
Total Time (*sec*)	58.27	58.27	58.27
Sampling Frequency (*Hz*)	30.07	30.01	30.01
Frequency Measured (*Hz*)	0.498	0.996	2.009
Validation Against Ground Truth			
RMSE (*mm*)	0.3788	0.3584	0.4243
Mean Absolute Error (*mm*)	0.315	0.300	0.356
R^2 Score	0.999	0.999	0.998
Error of Frequency	0.402%	0.800%	0.100%

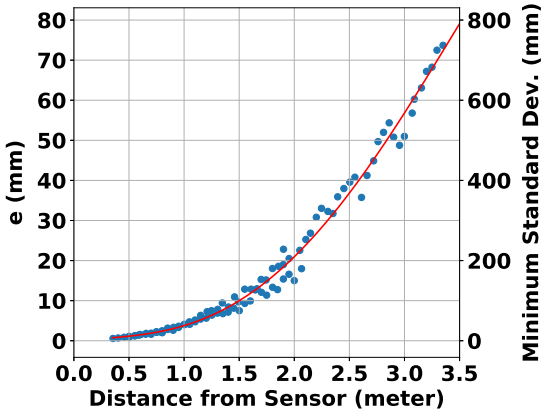


Figure 3.23: Noise floor of the RS depth sensor

point. By pointing the sensor at a smooth, flat surface, the depth quality tool fits the measured depth points in the ROI to a plane (which gives the mean value of the depth measurements) and calculates the RMS error of the measurements. The result is plotted versus the measurement distance in Figure 3.23 with the required minimum standard deviation of the signal (measured as the standard deviation of the vibration) of an ROI to achieve a 10% noise-to-signal ratio (defined as the standard deviation of noise over the standard deviation of the signal). To achieve less noise and higher accuracy from stereo vision for longer distances, cameras with higher resolution are needed, and the baseline between the two cameras needs to be increased.

Next, to test the proposed technique using a moving (non-stationary) RS Depth Sensor for structural displacement measurements, the rigid prismatic structure was secured onto the shaking table as before; however, the RS sensor was placed on a rolling cart, which allowed for sensor motion in the depth direction (on the order of 0.5-m) to simulate the drift of the UAS in the z-direction. The input frequencies and amplitudes of the shaking table were the same as those for the stationary 1C depth measurement analysis. The IR video with the appearance of both the rigid structure and reference plane was recorded with 480 $\times$ 640-*pixel* at 30-*fps*. The results of the tests are listed in Table 3.7 with the sample time histories of two measurements plotted in Figure 3.24. The RMS error of vibration amplitude is on the order of a few *mm* and is consistently

small throughout the three input frequencies and two input amplitudes tested. The time-varying absolute error shown in Figure 3.24 is inherent with the RS depth sensor, and its magnitude is a function of the small baseline and α angle (Figure 3.17a). The frequencies of the signals measured by the proposed technique agree well with those of the ground truth.

Table 3.7: Results from 1C non-stationary depth validation test

Ground Truth						
Frequency (*Hz*)	0.495	0.495	1.004	1.004	2.007	2.007
Amplitude (*mm*)	15.7	35.6	15.7	35.6	15.7	35.6
1C Depth Measurement						
Number of Samples	613	744	693	856	333	673
Total Time (*sec*)	20.41	24.82	23.08	29.32	11.07	22.48
Sampling Frequency (*Hz*)	30.03	29.98	30.02	29.20	30.07	29.94
Frequency Measured (*Hz*)	0.491	0.484	0.998	0.990	1.993	2.005
Validation Against Ground Truth						
RMSE (*mm*)	0.552	2.101	1.068	1.446	0.456	2.855
Mean Absolute Error (*mm*)	0.455	1.621	0.907	1.204	0.353	2.229
R^2 Score	0.997	0.993	0.990	0.997	0.998	0.985
Error of Frequency	0.811%	2.25%	0.599%	1.404%	0.700%	0.100%

Proof of Concept: An Experiment on 3C Measurement

This experiment was designed to test the proposed methodology in a small-scale laboratory setting. The double-faceted technique was used to measure the full 3C displacement of the model structure. The UAS was equipped with the RS sensor, and both the optical sensor and the depth system of the RS sensor were employed to measure the full 3C displacement of the ROI from about 0.75-*m* above the structure. In the test, a 252.4 $\times$ 609.6-*mm* cantilever structure constructed of balsa wood was placed on the shaking table with the 3-axis accelerometer placed at the end of the cantilever. Reference targets were placed on the cantilever to allow for DLT and a stationary reference target was placed 18-*cm* from the staking table. The corresponding horizontal measuring distance is 0.735-*m* for the optical and 1.53-*m* for the IR image. Sample images are shown in Figure 3.25. Since the optical sensor has a smaller FOV compared to the IR sensor, the FOV of the

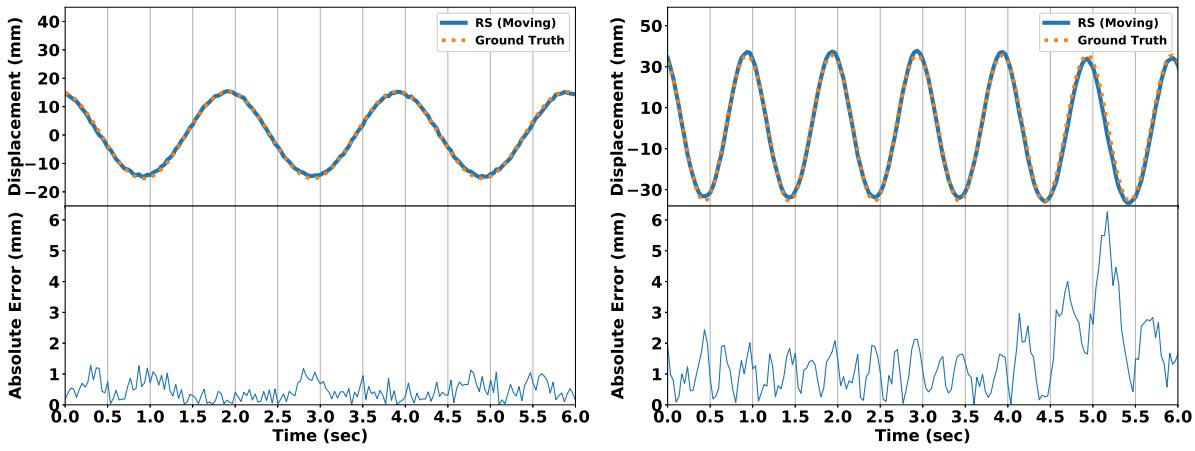
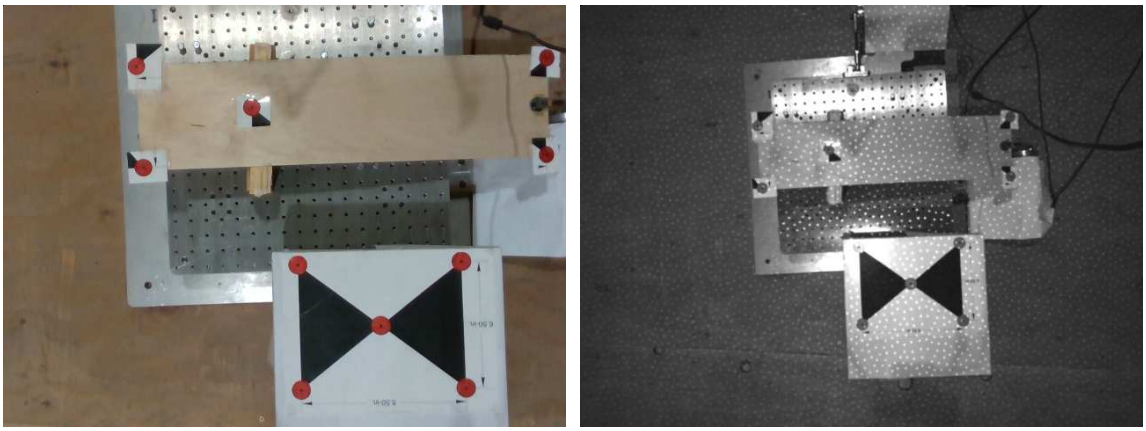


Figure 3.24: Sample displacement measurement results from *non-stationary* RS depth sensor with the time history at the top and absolute error at the bottom: (a) $f_0 = 0.50$-*Hz* and A = 15.7-*mm* and (b) $f_0 = 1.00$-*Hz* and A = 35.6-*mm*

optical sensor controlled the height of the UAS in this experiment. The z-direction of the structure was excited by flicking the cantilever. Additionally, some small ambient load was generated by the wind induced by the thrust of the UAS's propellers. The shaking table was used to generate the rigid body motion of the structure in x- and y-directions. In such a way, the displacements in all three directions were produced. The video was recorded at 640 $\times$ 480-*pixel* at 60-*fps*. The inputs of the shaking table for this test were 0.5-*Hz* with an amplitude of 35.6-*mm* in the x-direction and 1.0-*Hz* with an amplitude of 15.7-*mm* in the y-direction. Given the set-up requirements of the experiment, the flight was performed indoors with a DJI Mavic 2 Pro. This UAS was chosen because of 1) the omnidirectional obstacle avoidance system built into the UAS (including sonar, visible sensors, and a dedicated CPU for stability processing) to ensure a safe flight in the GPS-deprived location and 2) it was extremely stable in-flight with random drifts of about 0.15-*m*. A DJI Inspire 2 was also tested but proved too unstable in the indoor environment with a significant drift of more than 0.5-*m*. This greater drift forced the DJI Inspire 2 to fly higher above the cantilever in order to keep the reference and the ROI in the video, which caused a significant error in the 1C depth measurement due to the increased distance between the camera and the structure. To



(a) **(b)**

Figure 3.25: FOV of RS sensors from 0.75-*m* above ground level: (a) Optical sensor and (b) IR sensor

quickly test the concept of the proposed framework, the RS sensor was wired to a ground control station with a 9-*m* USB 3.1 cable. The setup for this experiment is shown in Figure 3.26. In a future implementation, the RS sensor could be self-sufficient without the need for the wire when secured to the UAS with an implementation of a microcontroller and a dedicated battery source. Additionally, to provide smoother images, the RS Sensor could be gimballed to the UAS.

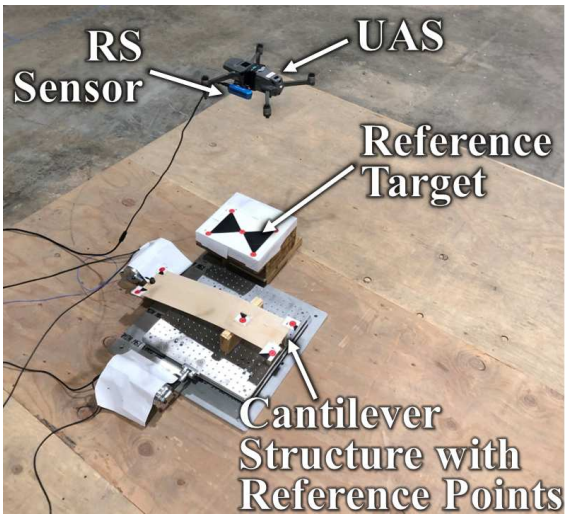


Figure 3.26: Set-up of 3C measurement experiment using a UAS equipped with an RS sensor

The results of the experiment are listed in Table 3.8. The ground truth data for *x*- and *y*-directions were obtained with the shake table using Equation 3.13 and are compared in the table.

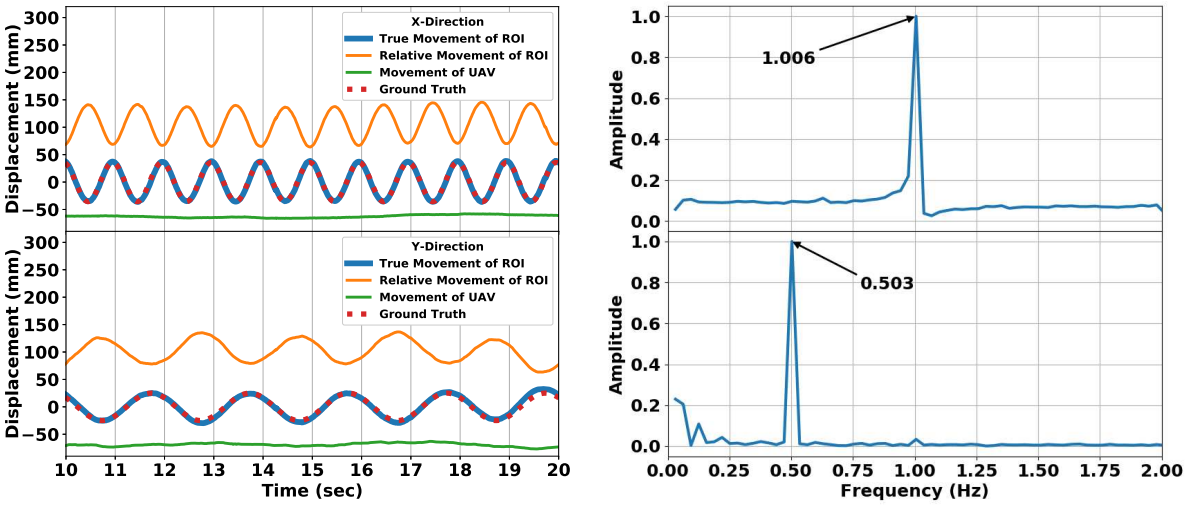


Figure 3.27: Displacement in x- and y-directions measured by UAS equipped with RS sensor: (a) Time history and (b) Power spectral density

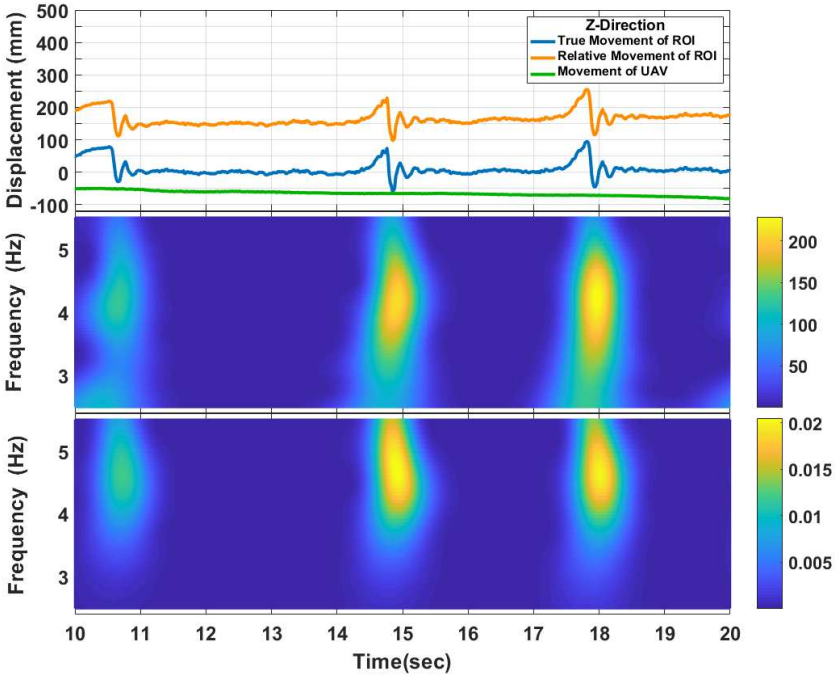


Figure 3.28: Displacement in z-direction measured by UAS equipped with an RS sensor with the measured time history at the top, the wavelet scalogram of the measured displacement in the middle, and the wavelet scalogram for the acceleration measured by accelerometer at the bottom (for comparison)

Table 3.8: Results from the 3C measurement experiment using a UAS equipped with an RS sensor

	X	*Y*	*Z*
Ground Truth			
Frequency (*Hz*)	0.504	1.007	4.673*
Amplitude (*mm*)	35.6	15.7	-
Damping Ratio	-	-	0.0928*
Measurement			
Number of Sample	1910	1910	1910
Total Time (*sec*)	31.83	31.83	31.83
Sampling Frequency (*Hz*)	60.02	60.02	60.02
Frequency Measured (*Hz*)	0.503	1.006	4.431
Damping Ratio Measured	-	-	0.0970
Validation Against Ground Truth			
RSME (*mm*)	1.648	3.352	-
Mean Absolute Error (*mm*)	1.342	2.514	-
R^2 Score	0.996	0.964	-
Error of Frequency	0.199%	0.100%	5.308%
Error of Damping Ratio	-	-	4.367%

* Obtained from Acceleration Using Wavelet Analysis

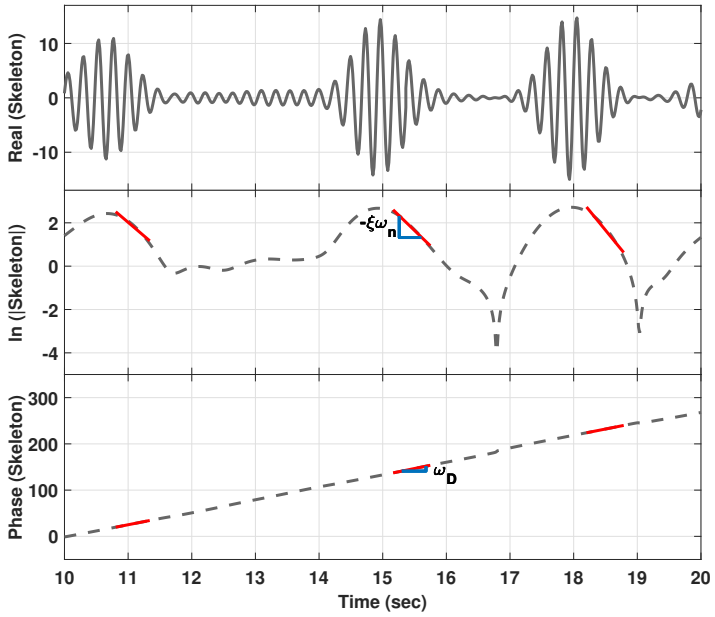


Figure 3.29: Wavelet analysis of the *z*-direction displacement

The time histories of the displacements in the x- and y-directions are shown in Figure 3.27 along with the Power Spectral Densities. It is seen from the results that there is a good agreement between the measured data collected through the proposed double-faceted technique and the ground truth data with a mean absolute error of amplitude less than 2-mm and frequency error less than 0.2% in the x- and y- directions. For the z-direction displacement, transient vibration responses were generated by finger flicking at the tip of the cantilever (Figure 3.28). Due to the transient and non-stationary nature of the response, the time-frequency analysis using continuous Morlet Wavelet Transform was performed to analyze the frequency contents in the signal (Figure 3.28) and identify the natural frequency and damping ratio of the vertical mode [74, 76, 101]. The wavelet skeleton, e.g., wavelet coefficients along the ridge in the wavelet scalogram, was extracted. This skeleton is the analytic signal at the dominant frequency, from which the instantaneous amplitude and phase angles were obtained. The damping ratio was obtained by taking the logarithm of the instantaneous amplitude and fitting a linear line for the free decay signal (Figure 3.29). The natural frequency was found by fitting a linear line to the phase angle of the wavelet skeleton (Figure 3.29). The instantaneous frequencies and damping ratios for the three transient signals were identified and averaged values are presented in Table 3.8. Since there is no ground truth data for displacement in the z-direction, the wavelet scalogram, as well as the natural frequency and damping ratio identified from the displacement data are compared to those obtained by the acceleration measured by the accelerometer. It is seen from Figure 3.28 that the dominant frequency contents from acceleration are slightly higher than those of the displacement, which is expected as the frequency response function (FRF) for acceleration equals the FRF of the displacement multiplied by the frequency squared. The agreement of the natural frequency and damping ratio identified from displacement and those obtained from acceleration is good (Table 3.8), considering the very short duration of the signal and inherent larger uncertainties in system identification. A comparison of the preliminary proposed technique to other techniques in the literature is presented in Table 3.9.

Table 3.9: Comparison of the proposed technique to other literature references

Method	***2C Planar RMSE***	***2C Freq. Error***	***1C Depth RMSE***
Proposed			
UAS-Enabled 3C Displacement Measurement	1.65-*mm* (4.6%)	0.10%	2.10-*mm* (5.9%)
Single Stationary Camera (2C)			
Markers on Target Locations [102]	-	0.80%	-
Markers on Target Locations [103]	0.5-*mm*	-	-
LED Markers on Target Locations [104]	-	0.78%	-
Painted Speckle Pattern [105]	2.7-12.6%	-	-
Stationary Stereo Vision (3C)			
Markers on target locations [106]	0.52-*mm*	-	0.35-*mm*
UAS-Enabled Single Camera (2C)			
High-Pass Filtering & Adaptive Scale-Factor [88]	-	1%	-
High-Pass Filtering [82]	-	1.6%	-
Feature Tracking of Background and ROI [87]	2.14-*mm*	-	-
UAS-Based Laser Doppler Vibrometer (1C)			
Vibrometer on UAS [84]	-	-	5%

Discussion

Despite its high accuracy in the proof of concept, the preliminary framework may not be practical in a real-world application. Below, the limitations are discussed.

1. **Measurement accuracy.** To apply this technique to field application, a longer working distance between the UAS and the structure is desired. To achieve a similar level of accuracy, the ground sampling distance of both the IR and optical sensors needs to be similar or better, and the angle between the two IR cameras should be equal to or greater than in the current experiments. This requires using cameras with higher resolution and longer baseline (i.e., distance between the cameras), and a laser projector with a longer working distance.

2. **Reference target size and location.** The size of the reference target is dependent on the ground sampling distance of the optical camera. In this proof of concept, the diameter of the red targets was 2.54-*cm*. If the ground sampling distance of the optical camera is similar to the proof of concept test, a similarly sized reference target could be used. In the laboratory

experiments, the reference target was located close to the structure so that it was in the same image frame as the ROI. In large-scale applications, it might not be feasible to have the reference and the structure in the same frame due to the constraints of the measurement field of the cameras.

3. **Lighting condition.** The virtual speckle pattern projected by the IR projector is less visible in direct sunlight. However, the proposed technique can be used in shadowed areas (e.g., under a bridge or on the shaded side of a building) or after sunset. The current bridge inspection may occur during the nighttime to alleviate traffic interruptions, providing a more favorable lighting condition for applying the proposed technique.

3.6 Task 5: 3C Dynamic Displacement Measurements: Large-Scale

3.6.1 Task 5: Motivation

Measuring the 3C dynamic displacement of full-sized structures is a vital step to validate FE models and more generally understand the performance of a bridge within a DT. Previously, in Task #4, (Section 3.5), a successful technique was developed to capture 3C measurements; however, a variety of limitations impeded measurements of real-world structures including the close distance to the region of interest, lighting conditions, and the small horizontal measuring distance of the sensors [60]. This task developed a technique to overcome these shortcomings and impediments to real-world applications. To facilitate 3C displacement measurements with sensors attached to a UAS in the field without the use of reference targets, the proposed framework with an upgraded sensor suite will be tested, analyzed, and validated with a real-world example. The literature review for this task has been completed in Section 3.5 and is omitted here for conciseness.

Considering the advantages of UAS technology, lessons learned, and limitations of the previous studies, this paper proposes to scale up the concept of 3C displacement measurement first proposed in [10] for application to full-scale structures, where the stereo-vision technique is uti-

lized to measure the 3C displacement of the structure and compensate for the motion of the UAS by tracking a stationary reference target. Despite the straightforward concept, significant challenges are associated when scaling up this technique for large-scale applications.

Challenge #1: Calibration of four cameras in dual stereo vision pairs. In large-scale applications, it is difficult to capture both the region of interest (ROI) on structures and the reference target in the same image due to the 3D layout of the structure, which necessitates the use of two pairs of cameras with one pair tracking the ROI and the other pair tracking the reference target. Although camera calibration (i.e., solving internal camera parameters) and stereo vision rectification (i.e., solving external camera parameters between a camera pair) have been well studied, calibrating four cameras together for dual stereo vision measurement is very challenging since the four cameras facing different directions cannot capture the same calibration targets. This type of dual stereo vision calibration has not been reported in the literature. To tackle this challenge, this study proposes a unique dual stereo vision calibration technique.

Challenge #2: Precisely accounting for the 6-degree-of-freedom (DOF) motion of a UAS. Large-scale applications require a longer working distance from the cameras to a structure for safety and accessibility considerations (the working distance is defined in Figure 3.30). To achieve high measurement accuracy at a further distance, a longer baseline (Figure 3.30) between two cameras in each pair is needed. With long camera baselines, the geometric measurement errors due to the 3-DOF rotation of the UAS system will be magnified significantly. Thus, compensating for the rotation of the UAS becomes a critical yet challenging step due to the multiple DOFs that are involved. Although some earlier studies have explored compensating for the translational motions of UAS [60, 83, 85], there have been no existing methodologies to compensate the 3-DOF rotation of UAS for 3C dynamic displacement measurements using computer vision in the literature. To overcome this challenge, this study proposes a mathematically elegant method to simultaneously measure both the 3-DOF rotation and 3-DOF translational motions of the UAS.

In addition to the above two challenges, tracking the natural features of structures is another important consideration for practice. Attaching artificial patterns or targets to a structure might be

impractical in practice [60, 83, 86]. To the author's knowledge, there is no successful implementation using only natural features to directly measure the 3C displacements of a structure with a non-stationary camera. Although there is a slight tradeoff between measurement accuracy and ease of implementation, this study will explore how to track natural features in concrete with minimal user input.

In the proposed technique, four total optical sensors are used: two cameras pointed at the ROI to record the dynamic displacement of the structure and two additional cameras pointed at a stationary reference to track the movement of the UAS. Each camera pair implements passive stereo vision to record the dynamic displacements without the need for an artificial target or laser projector. The 6-DOF displacements of the UAS are measured and compensated for in the ROI measurements. This dual passive stereo vision framework is novel and provides a portable system for structural surveillance compared to stationary non-contact sensors while still being able to measure the 3C dynamic displacement using the full FOV of the camera. The proposed technique has been published in [107].

The methodology section directly presents the techniques to address the three major challenges. Next, experimental results are shown to explore the effectiveness of the methodologies. The lab test is specifically designed to stress-test the three challenges, while a second test explores the practicality of the system with the cameras attached to a UAS. Lastly, there is a discussion on the advantages, limitations, and lessons learned from the system, with a conclusion summarizing the results.

Objectives

1. Develop methodology considering the lesson learned from the RealSense D435 sensor
2. Overcome the limitations of Task #4 for full-scale 3C dynamic displacement measurements
3. Account for all the 6-degree-of-freedom motion of the UAS as to hovers and collects data
4. Perform a field test with the camera rig attached to a UAS

3.6.2 Task 5: Methodology

This section will discuss the methodology of the dual stereo vision system. First, the design of the camera rig, which holds the four cameras, and the UAS used will be discussed. Next, the methodology to calibrate the four cameras to find the relative location of the cameras is explored. Lastly, this section will discuss the technique to measure the ROI and reference point (REF) displacements and correct for the UAS movement.

UAS-Based Camera Platform

Four GoPro Hero 10s (hereafter, GPs) were chosen because of their compact size, sufficient battery, and high-quality imagery. The GPs had a superior ground sampling distance (GSD) due to the increased sensor size and the number of captured pixels than those used in the previous study [60]. From a 5-m distance, the GPs had a GSD of 1.99-*mm/pixel*. Additionally, from a 5-m distance, the cameras capture an area of 5.9-$m \times 10.6$-m. With a 1/2.3" sensor (6.17-$mm \times$ 4.55-*mm*), the GPs were able to capture 16-*MP* images at a sampling rate of 60-*frames/sec (fps)*. The GPs were attached to a DJI Matrice 600 Pro UAS. This UAS can carry a 5.5-*kg* payload while providing about 20-*min* of flight time.

The two camera pairs were placed at a 90° angle from each other with one pair tracking a moving object (structure) and the other pair tracking a stationary target, as shown in Figure 3.30. As a note, the stationary reference is not required to be under the UAS; likewise, the ROI is not required to be in front of the UAS as shown in Figure 3.30. Rather, the stereo camera pairs can be placed at any angle from each other according to the geometries of the structure of interest and available reference in field applications. The configuration of the four GPs and the UAS are shown in Figure 3.31. The entire camera rig was designed to be as rigid as possible to ensure that the four cameras moved in tandem (i.e., there is no relative motion among cameras), which is vital for accurate measurements. Each camera was bolted to a square steel tube, and the bolted connections were secured with a permanent thread lock. The rigid steel tube restricted the movement between the GPs (i.e., consistent baseline). Next, to reduce the camera's vibration while it is attached to the UAS, rubber washers, rubber grommets, and rubber rings were attached to the connections

between the UAS and the steel square tube. The steel tubing and bolted cameras are referred to as the *camera rig*. The camera rig was removed from the UAS during calibration and lab testing for easier handling. The gimbaled camera originally attached to the Matrice 600 was still operable during the flight for easier navigation and control.

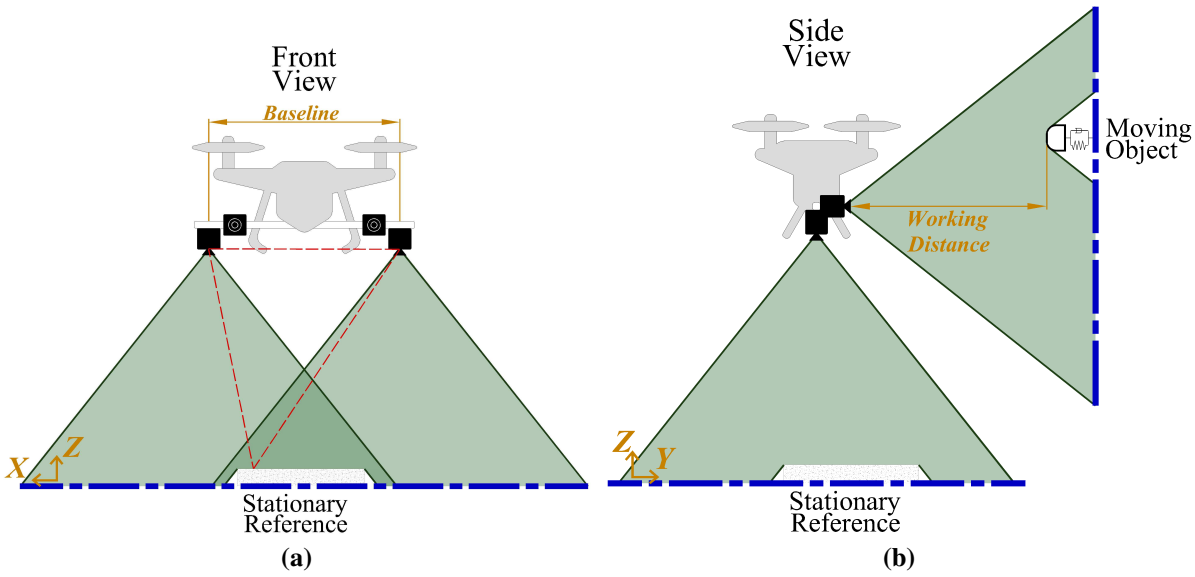


Figure 3.30: Overview of dual stereo vision camera setup: (a) Front view of the UAS system with the overlap of the stereo cameras shown; and (b) Side view of the UAS system with the layout of the dual stereo vision system

Dual-Pair Camera Calibration

Before taking measurements with the cameras, they need to be calibrated. The dual stereo vision process requires a three-step calibration process:

1. Individual camera calibration to undistort images
2. Stereo rectification to solve for the extrinsic parameters of a camera pair and virtual intrinsic parameters
3. Dual stereo pairs calibration to find the location of the camera pairs with respect to each other and to unify measurements to one local coordinate system

(a) **(b)**

Figure 3.31: Dual stereo vision camera rig attached to a DJI Matrice 600 Pro UAS: (a) Detached camera rig used during calibration and rectification; and (b) Matrice 600 Pro with attached camera rig

Step 1: Individual camera calibration. The dual stereo pairs calibration is a novel technique, while individual camera calibration and rectification have been well-researched previously. Typical camera calibration identifies the *Intrinsic Matrix*, **I**, (Equation 3.14) and distortion coefficients, which removes distortions in the image [92]. **I** consists of the camera's focal lengths in the x- and y- directions (f_x and f_y, respectively), the skew factor, s, and the principal points of the image (c_x and c_y). In this implementation, the skew factor, s, was assumed to be 0, a typical assumption for modern cameras. The camera parameters were found using MATLAB's built-in *Camera Calibration App* with a checkerboard pattern. A high-quality checkerboard print was printed on thick paper and glued to a sheet of glass to ensure a smooth flat surface, which is important for accurate calibration and rectification. The results were then exported for later use. As a note, the camera calibration must be performed on every camera individually (i.e., four total sets of calibration parameters); however, the parameters remain constant for each individual camera unless there is a significant knock to the camera or a large temperature shift [95].

$$[\mathbf{I}]_{3\times4} = \left[\begin{array}{ccc|c} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \tag{3.14}$$

Step 2: Stereo vision rectification. Once the cameras are calibrated, and the distortions are removed from the images, each camera pair must be rectified. Rectification ensures that epipolar lines exist between the two cameras (i.e., the vertical pixel location of the ROI is the same between the two cameras) and that the image planes of the camera are on the same plane. Stereo rectification solves for the rotation, $\mathbf{R}$, and translation, $\mathbf{T}$, of the second camera (i.e., right camera in this implementation) with respect to the first camera (i.e., left camera). The rotation matrix and translation vector can then be combined to build the *Extrinsic Matrix*, $\mathbf{E}$, (Equation 3.15), which defines the transformation from the sensor of the left camera to the sensor of the right camera. This process was performed with MATLAB's built-in *Stereo Camera Calibrator App* with a checkerboard pattern glued to a glass plane. After rectification, the two image planes are assumed to be on the same plane. This transformation changes the focal length, principal point, and baseline found in the camera calibration procedures. Therefore, a virtual focal length, principal point, and baseline are estimated based on the rectification parameters. These virtual parameters are used in subsequent steps to find the 3D distances. The camera pair rectification must be performed for each camera pair (i.e., two times total) but does not need to be re-performed unless the cameras are moved or rotated with respect to each other from their original position. During the camera calibration and rectification procedures, the camera rig was suspended above the ground with a tripod with the calibration plane placed on the ground. The cameras were triggered remotely using a proprietary remote for the GPs. This remote is capable of triggering the GPs to start capturing videos at roughly the same time with an error within a couple of frames. Although this synchronization error is too large for measuring dynamic displacement, here, it is not an issue for calibration and rectification since the cameras and calibration board are in a stationary position for each video capture.

$$[\mathbf{E}]_{4\times4} = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{T} \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3.15}$$

Step 3: Dual stereo pairs calibration. The dual pairs calibration is performed to measure the position of each camera pair with respect to the other camera pair. A novel dual stereo pairs calibration technique is developed to measure the camera's pose and transform the 3C distances measured by each camera pair in their unique local camera coordinate system into the same local coordinate system. The camera pose is the camera's position and rotation at a given time point. This effort addresses *Challenge #1* of the dual stereo vision technique. For this process, only the left camera poses are used (the transformations from the right camera to the left camera are known through $\mathbf{E}$ in Equation 3.15, found during camera pair rectification). Since two pairs of cameras are facing different directions and cannot see the same ROI, a unique 3D calibration target is created. Specifically, a checkerboard calibration pattern with numbered squares is attached to two flat surfaces normal to each other. Two sheets of glass were attached to the wall and the ground and leveled to ensure they were perpendicular. A long checkerboard pattern was printed on thick paper and glued to the glass panes for a smooth surface. Next, the camera rig is suspended above the 3D calibration target, and a short, 30*sec* video is recorded while the camera rig is randomly rotated and translated by hand in all 6-DOF. The set-up of this calibration process is shown in Figure 3.32. Since the distances between the corners of the checkerboard are known (i.e., each square has a width of 1.651-*cm*), the relationship between the two camera pairs is found. First, the transformation to convert the points measured in the local coordinate system of the camera to global coordinates $\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$, is found by

Figure 3.32: 3D calibration target set-up for dual stereo pairs calibration with the camera rig suspended above the target

$$\underbrace{\begin{bmatrix} i & j & k & 1 \end{bmatrix}^T}_{\textit{ROI Cam. Coord.}} = \begin{bmatrix} \mathbf{A}_{w\to c}^{ROI} \end{bmatrix} \underbrace{\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T}_{\textit{Global Coord.}} \tag{3.16}$$

$$\underbrace{\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T}_{\textit{REF Cam. Coord.}} = \begin{bmatrix} \mathbf{A}_{w\to c}^{REF} \end{bmatrix} \underbrace{\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T}_{\textit{Global Coord.}} \tag{3.17}$$

where $\begin{bmatrix} i & j & k & 1 \end{bmatrix}^T$ are the identified points in the local camera coordinates of the *ROI* cameras, $\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T$ are the identified points in the local camera coordinates of the *REF* cameras, and $\begin{bmatrix} \mathbf{A}_{w\to c}^{ROI} \end{bmatrix}$ and $\begin{bmatrix} \mathbf{A}_{w\to c}^{REF} \end{bmatrix}$ are the transformation matrices from global coordinates to local camera coordinates for the *ROI* and *REF* cameras, respectively. The transformation matrices are composed of a 3×3 rotation matrix and 3×1 translation vector.

$$\begin{bmatrix} \mathbf{A}_{w\to c}^{ROI} \end{bmatrix} = \left[\begin{array}{c|c} \mathbf{R}_{w\to c}^{ROI} & \mathbf{T}_{w\to c}^{ROI} \\ \hline 0 & 1 \end{array}\right]_{4\times 4} \tag{3.18}$$

$$\left[\mathbf{A}_{w\to c}^{REF}\right] = \left[\begin{array}{c|c} \mathbf{R}_{w\to c}^{REF} & \mathbf{T}_{w\to c}^{REF} \\ \hline 0 & 1 \end{array}\right]_{4\times 4} \tag{3.19}$$

Since the coordinates of checkerboard points are known in both the local camera coordinates and the global coordinates, the transformation is solved using singular vector decomposition. First, the covariance matrix, $\mathbf{H}$, is defined for each transformation matrix given a set of n points in a frame as

$$\mathbf{H}^{ROI} = \left[\begin{bmatrix} i \\ j \\ k \end{bmatrix}_{3\times n} - \begin{bmatrix} \bar{i} \\ \bar{j} \\ \bar{k} \end{bmatrix}_{3\times 1}\right] \times \left[\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{3\times n} - \begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \end{bmatrix}_{3\times 1}\right]^T \tag{3.20}$$

$$\mathbf{H}^{REF} = \left[\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{3\times n} - \begin{bmatrix} \bar{x} \\ \bar{y} \\ \bar{z} \end{bmatrix}_{3\times 1}\right] \times \left[\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{3\times n} - \begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \end{bmatrix}_{3\times 1}\right]^T \tag{3.21}$$

where the barred coordinates represent the average, and *n* is the number of identified points in one frame. After the covariance matrices for the *ROI* and *REF* points are found, singular value decomposition of these covariance matrices is performed such that $\mathbf{H} = \mathbf{U} \times \mathbf{S} \times \mathbf{V}^T$. By multiplying the right singular vectors, $\mathbf{V}$, by the transpose of the left singular vectors, $\mathbf{U}$, the rotation matrix, $\mathbf{R}_{w\to c}$ is found.

$$\left[\mathbf{R}_{w\to c}^{ROI}\right]_{3\times 3} = \left[\mathbf{V}^{ROI}\right]_{3\times 3} \times \left[\mathbf{U}^{ROI}\right]_{3\times 3}^T \tag{3.22}$$

$$\left[\mathbf{R}_{w\to c}^{REF}\right]_{3\times 3} = \left[\mathbf{V}^{REF}\right]_{3\times 3} \times \left[\mathbf{U}^{REF}\right]_{3\times 3}^T \tag{3.23}$$

Lastly, the translation vectors are found by

$$\left[\mathbf{T}_{w\to c}^{ROI}\right]_{3\times 1} = \left[\bar{X} \quad \bar{Y} \quad \bar{Z}\right]_{3\times 1}^T - \left[\mathbf{R}_{w\to c}^{ROI}\right]_{3\times 3} \times \left[\bar{i} \quad \bar{j} \quad \bar{k}\right]_{3\times 1}^T \tag{3.24}$$

$$\left[\mathbf{T}_{w\to c}^{REF}\right]_{3\times 1} = \left[\bar{X} \quad \bar{Y} \quad \bar{Z}\right]_{3\times 1}^T - \left[\mathbf{R}_{w\to c}^{REF}\right]_{3\times 3} \times \left[\bar{x} \quad \bar{y} \quad \bar{z}\right]_{3\times 1}^T \tag{3.25}$$

With the solved transformations, the transformation from the local coordinate system of the *ROI* camera to the local coordinate system of the *REF* camera, $\left[\mathbf{A}^{ROI\rightarrow REF}\right]$, is found by converting the *ROI* points to global coordinates and then to the local *REF* coordinate system with the following transformation,

$$\left[\mathbf{A}^{ROI\rightarrow REF}\right] = \left[\begin{array}{c|c} \mathbf{R}_{w\rightarrow c}^{REF} & \mathbf{T}_{w\rightarrow c}^{REF} \\ \hline 0 & 1 \end{array}\right] \left[\begin{array}{c|c} \mathbf{R}_{w\rightarrow c}^{ROI} & \mathbf{T}_{w\rightarrow c}^{ROI} \\ \hline 0 & 1 \end{array}\right]^{-1} \tag{3.26}$$

Therefore, the points in the local camera coordinate system of the *ROI* camera are converted to the local camera coordinates of the *REF* camera by

$$\underbrace{\begin{bmatrix} x & y & z & 1 \end{bmatrix}^T}_{\textit{REF Cam. Coord.}} = \left[\mathbf{A}^{ROI\rightarrow REF}\right] \underbrace{\begin{bmatrix} i & j & k & 1 \end{bmatrix}^T}_{\textit{ROI Cam. Coord.}} \tag{3.27}$$

With the above-defined transformation matrix, $\left[\mathbf{A}^{ROI\rightarrow REF}\right]$, the points measured from the *ROI* camera pair (in engineering units in the *ROI* local camera coordinates) are then transformed into the local camera coordinates of the REF camera pair. With a short video, a rotation matrix is solved by averaging the rotation matrices for each frame. Specifically, the rotation matrix of each frame is converted into quaternions. Then, the quaternions are averaged and converted back into a rotation matrix, $\left[\mathbf{R}^{ROI\rightarrow REF}\right]$. The translation vector is also averaged over the frames, $\left[\mathbf{T}^{ROI\rightarrow REF}\right]$. $\left[\mathbf{A}_{w\rightarrow c}^{REF}\right]$ and $\left[\mathbf{A}_{w\rightarrow c}^{ROI}\right]$ can vary from frame to frame depending on the movement of the cameras; however, $\left[\mathbf{A}^{ROI\rightarrow REF}\right]$ remains constant since the position of the camera pairs relative to each other remains constant. With all the measured points in the same local coordinate system, the measured distances are related and later transformed into global coordinates for displacement measurements. This process is done only once for the two camera pairs and remains constant if the cameras are not shifted (i.e., during transportation of the camera rig).

Dual Stereo Vision Measurements

To provide high accuracy at a farther measuring distance, the baseline between the cameras within a camera pair is increased to about 80-*cm*. Theoretically, with an extended baseline, the accuracy of the measurements increases; however, there are limits to the baseline length in implementation. For example, occlusions of the ROI, size of camera rig, inability to match keypoints, etc. could be factors limiting further increases of the baseline. The cameras used had a superior GSD due to the increased sensor size and the number of captured pixels than those used in the previous study [60].

Each stereo vision camera pair measures the x-, y-, and z-coordinates of a matched keypoint in the local camera coordinate system of the left camera. To find these measurements, first, the keypoints between the cameras are found and matched between the two cameras. To avoid the need to use artificial patterns, natural features intrinsic in the material are found and tracked. Keypoints are initially manually identified within the image frame where measurements are needed. This minimal user input only requires the user to select matching keypoints in each camera pair for the first image frame. For instance, if the user would like to track the motion of a bolt, the user will manually select a keypoint on the bolt in the left camera and then manually select the corresponding keypoint in the right camera of a camera pair to create a matching keypoint set. The procedure can be aided by zooming in on the bolt extremely closely to find a matching keypoint between the two cameras. An advantage of manually initializing and matching keypoints over other automatic techniques is that measurements can be taken at any location a keypoint can be tracked instead of relying on the identified keypoint location with other algorithms. In this implementation, five keypoints were matched between the two cameras in each pair. Then, *optical flow* is implemented to track the selected keypoints frame-to-frame. Optical flow tracks keypoints using a *Harris Corner Detector* and then searches within a small window in the next frame for a similar point [65–67]. As a note, alternatively, the keypoints can be automatically matched by other techniques such as block matching, which is typically used to generate depth maps of the entire FOV of the cameras in stereo vision applications [108]. However, manually matching the keypoints initially

and implementing sub-pixel optical flow across the frames produced more precise results than using the block matching technique, which has made tracking natural features possible. Using the existing automated matching techniques, tracking the natural features can be very unreliable. After the keypoints between the two cameras are found, the 3D locations of the keypoints are determined using stereo vision.

With a matched set of identified keypoints, the 3D loations of the keypoints are found. The variables for the stereo vision calculations are defined in Figure 3.33. The image plane in Figure 3.33 is the rectified image plane. Correspondingly, f (which is the same for both cameras) and B are the virtual focal length and baseline, respectively. u^L_{pix} and v^L_{pic} are the pixel distances from the virtual principal point in the local x- and y-directions for the left camera, while u^R_{pix} is the pixel distance from the virtual principal point in the local x-direction for the right camera. The star on the target is an identified and matched keypoint, and x, y, and z are the 3D coordinates in the local camera coordinate system. Using similar triangles, the 3D geometry is defined by,

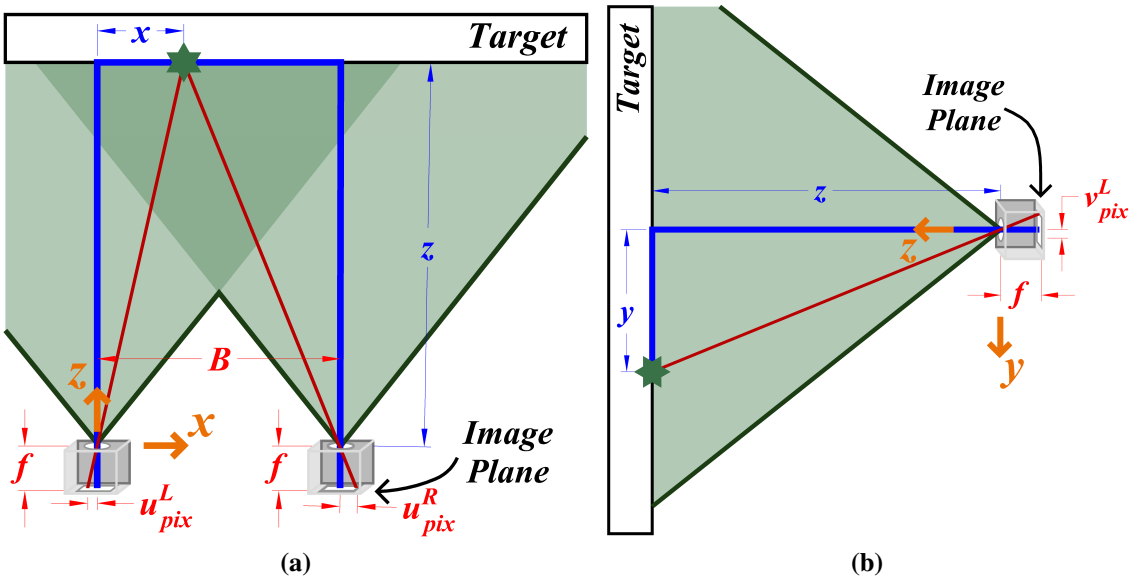


Figure 3.33: Schematic diagram of the general stereo vision geometry (plotted based on REF cameras): (a) Solving for the x- and z-directions; and (b) Solving for the y-direction

$$\frac{f}{u_{pix}^{L}} = \frac{z}{x}; \qquad \frac{f}{u_{pix}^{R}} = \frac{z}{(x-B)}; \qquad \frac{f}{v_{pic}^{L}} = \frac{z}{y} \tag{3.28}$$

$$x = \frac{B \cdot u_{pix}^{L}}{\left(u_{pix}^{R} - u_{pix}^{L}\right)}; \qquad z = \frac{B \cdot f}{\left(u_{pix}^{R} - u_{pix}^{L}\right)}; \qquad y = \frac{B \cdot v_{pic}^{L}}{\left(u_{pix}^{R} - u_{pix}^{L}\right)} \tag{3.29}$$

where $\left(u_{pix}^{R} - u_{pix}^{L}\right)$ is defined as *disparity*, which is the pixel distance between a pixel in the left camera to the corresponding pixel in the right camera.

Lastly, to address *Challenge #2*, the translation and rotation of the UAS are measured on a frame-by-frame basis. The rotation and translation measurement of the UAS follows a similar mathematical principle to that stated in [109] for tracking of windborne debris in space using stereo vision. The proposed technique directly solves for the transformation matrix from the global coordinate system to the local coordinate system of the left REF camera. Using this transformation matrix, the 6-DOF motion of the UAS is recovered by tracking points in a reference plane. The directional unit vectors, $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{w}}$, and the center point, $\mathbf{C}$ are defined using the reference plane in Figure 3.34. First, a plane of best fit of the reference points, whose locations are measured with respect to the local coordinate system of the left REF camera, (i.e., $\mathbf{Q1}$ through $\mathbf{Q5}$) is found using least squares. This fitted plane is the reference plane in the real world. The reference points are split into two groups: group 1 consists of $\mathbf{Q1}$, $\mathbf{Q2}$, and $\mathbf{Q3}$, and group 2 consists of $\mathbf{Q3}$, $\mathbf{Q4}$, and $\mathbf{Q5}$. The centroids of the two groups are found and projected onto the fitted reference plane (i.e., $\mathbf{C}$ and $\mathbf{D}$ for group 1 and group 2, respectively). The directional vector, $\hat{\mathbf{w}}$, is found by taking the unit normal of the fitted plane. Next, the directional vector, $\hat{\mathbf{u}}$, is defined as the unit vector from $\mathbf{C}$ to $\mathbf{D}$: $\hat{\mathbf{u}} = \frac{\mathbf{CD}}{|\mathbf{CD}|}$. Lastly, the directional vector $\hat{\mathbf{v}}$ is defined as the cross product of $\hat{\mathbf{u}}$ and $\hat{\mathbf{w}}$: $\hat{\mathbf{v}} = \frac{\hat{\mathbf{u}} \times \hat{\mathbf{w}}}{|\hat{\mathbf{u}} \times \hat{\mathbf{w}}|}$. Point $\mathbf{C}$ is defined as the origin of the global coordinate system, while $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$, and $\hat{\mathbf{w}}$ define the mutually perpendicular coordinate axis. As a note, only three non-collinear points in the reference plane are required; however, it was found that using two groups of three points produced a more stable global coordinate system as it averaged out random fluctuations of the point measurements. The transformation matrix from global points to the local camera coordinates of the REF camera

is defined by,

$$\left[\mathbf{F}\right] = \begin{bmatrix} \mathbf{R} & \mathbf{C} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{u}} & \hat{\mathbf{v}} & \hat{\mathbf{w}} & \mathbf{C} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \hat{u}_1 & \hat{v}_1 & \hat{w}_1 & C_1 \\ \hat{u}_2 & \hat{v}_2 & \hat{w}_2 & C_2 \\ \hat{u}_3 & \hat{v}_3 & \hat{w}_3 & C_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

The inverse of $\left[\mathbf{F}\right]$, $\left[\mathbf{F}^{-1}\right]$, defines the transformation from the local camera coordinates of the REF camera to the global coordinate system,

$$\left[\mathbf{F}^{-1}\right] = \begin{bmatrix} \hat{\mathbf{u}}^{\mathbf{T}} & -\hat{\mathbf{u}}^{\mathbf{T}}\mathbf{C} \\ \hat{\mathbf{v}}^{\mathbf{T}} & -\hat{\mathbf{v}}^{\mathbf{T}}\mathbf{C} \\ \hat{\mathbf{w}}^{\mathbf{T}} & -\hat{\mathbf{w}}^{\mathbf{T}}\mathbf{C} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \hat{u}_1 & \hat{v}_1 & \hat{w}_1 & -(\hat{u}_1 C_1 + \hat{u}_2 C_2 + \hat{u}_3 C_3) \\ \hat{u}_2 & \hat{v}_2 & \hat{w}_2 & -(\hat{v}_1 C_1 + \hat{v}_2 C_2 + \hat{v}_3 C_3) \\ \hat{u}_3 & \hat{v}_3 & \hat{w}_3 & -(\hat{w}_1 C_1 + \hat{w}_2 C_2 + \hat{w}_3 C_3) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

$\left[\mathbf{F}^{-1}\right]$ allows the transformation of points in the local coordinate system of the REF camera into global coordinates. Since the reference points do not move, the reference coordinate system remains constant. This transformation is solved on a frame-by-frame basis to compensate for the random drift and rotation of the UAS.

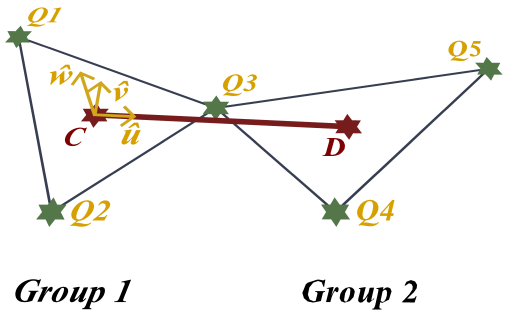


Figure 3.34: Reference plane and the global coordinate system

With all the transformation matrices defined, the points measured in the local camera coordinate system of the ROI cameras ($\begin{bmatrix} i & j & k & 1 \end{bmatrix}^T$) are directly converted into the constant global

coordinates ($\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$), where the 6-DOF of UAS motion is compensated for via $\begin{bmatrix} \mathbf{F}^{-1} \end{bmatrix}$. Specifically, the ROI points in the local ROI camera coordinate system are firstly converted into the local REF camera coordinate system through $\begin{bmatrix} \mathbf{A}^{ROI \rightarrow REF} \end{bmatrix}$. The points in the local REF camera coordinate system are then transformed into global coordinates through $\begin{bmatrix} \mathbf{F}^{-1} \end{bmatrix}$. Finally, the measured 3C motion of ROI points is represented in the constant global coordinates by

$$\underbrace{\begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T_{4\times n}}_{\textit{ROI Points in Global Coord.}} = \underbrace{\begin{bmatrix} F^{-1} \end{bmatrix}_{4\times 4}}_{\textit{Updated Frame-by-Frame}} \underbrace{\begin{bmatrix} A^{ROI \rightarrow REF} \end{bmatrix}_{4\times 4}}_{\textit{Constant}} \underbrace{\begin{bmatrix} i & j & k & 1 \end{bmatrix}^T_{4\times n}}_{\textit{ROI Points in Local Coord.}} \tag{3.32}$$

3.6.3 Task 5: Results

Validation of the Proposed Dual Stereo Vision Technique with Nonstationary Cameras

The results of the validation of the proposed technique are presented in this section. The full calibration process was performed once and assumed to be constant throughout the rest of the experiments. Each camera pair was calibrated and rectified using MATLAB's built-in *Stereo Camera Calibrator App*. The pixel error of this calibration process was less than 0.5-*pixels* for both camera pairs. After calibration and rectification, the ROI and REF camera pairs underwent the dual stereo pairs calibration procedure to find the transformation between the coordinate systems of the two camera pairs. Then, the transformation from the local coordinate system of the REF camera to the global coordinate system is solved for each frame. To test the efficacy of the entire workflow, an experiment is conducted to measure the global position of stationary points on the 3D calibration target (Figure 3.32) while the camera rig is moving. Since the remote triggering of the cameras cannot guarantee perfect synchronization, the cameras were first synced through post-processing using the autocorrelation function of the sound waves generated by hand clapping. The synchronization using the auto-correlation might still be slightly off, as it cannot recover the sub-frame time offset. It was observed from many repetitive experiments that the sub-frame time offset may cause significant measurement errors. Thus, a second step is taken to achieve more

accurate synchronization, where the frames are further synced by interpolating the pixel location of an identified keypoint between the frames. In this experiment, the 3D location of the tracked stationary points was measured as the camera rig was significantly rotated and translated by hand above the 3D calibration planes. In a perfect result, the 3D locations of the points would be stationary since only the camera rig (and not the points) move; however, due to tracking errors and imperfect calibrations, there is a slight error. The plot of the estimated *X*, *Y*, and *Z* location of a stationary point is shown for 1,200 frames (about 20*sec*) in Figure 3.35. The translation and rotation of the camera rig during this test are also plotted in Figure 3.36. Note the significant translation and rotation of the camera rig (on the order of ± 50-*mm* and $\pm 15^{\circ}$) during this test. The root-mean-square (RMS) error to the ground truth is 0.61-*mm*, 0.89-*mm*, and 0.63-*mm* for the *X*-, *Y*-, and *Z*-directions, respectively. The sub-millimeter RMS error shows that the mathematical concept of dual camera pairs calibration, stereo vision measurement, and compensation of 6-DOF motion of moving cameras is valid in practice.

UAS-Based Measurement Results

After the proposed technique was validated in a lab setting with high accuracy, the camera rig was placed on a UAS and flown to test the efficacy of measuring dynamic displacements. A shake table that simulated a sine wave (Equation 3.13) in two directions (*X*- and *Y*-directions) was used to provide a dynamic ROI with known motion (as discussed in Section 3.5). The inputs of the sine wave (i.e., the amplitude and the frequency) are controllable and, therefore, provide the ground truth motion. Two concrete beams were placed on top of the shake table to simulate natural features that would be present in a concrete structure. The frame rate of the cameras is 60-*fps*. The UAS was flown approximately 5-*m* from the shake table. This distance is believed to be sufficient for field applications. The experimental setup is shown in Figure 3.37. Checkered patterns were placed on both the ground under the UAS and on the concrete on the shake table for initial testing of the workflow. However, these patterns were not used in the final data processing. The tracked points from natural features used in the final test are shown in Figure 3.38.

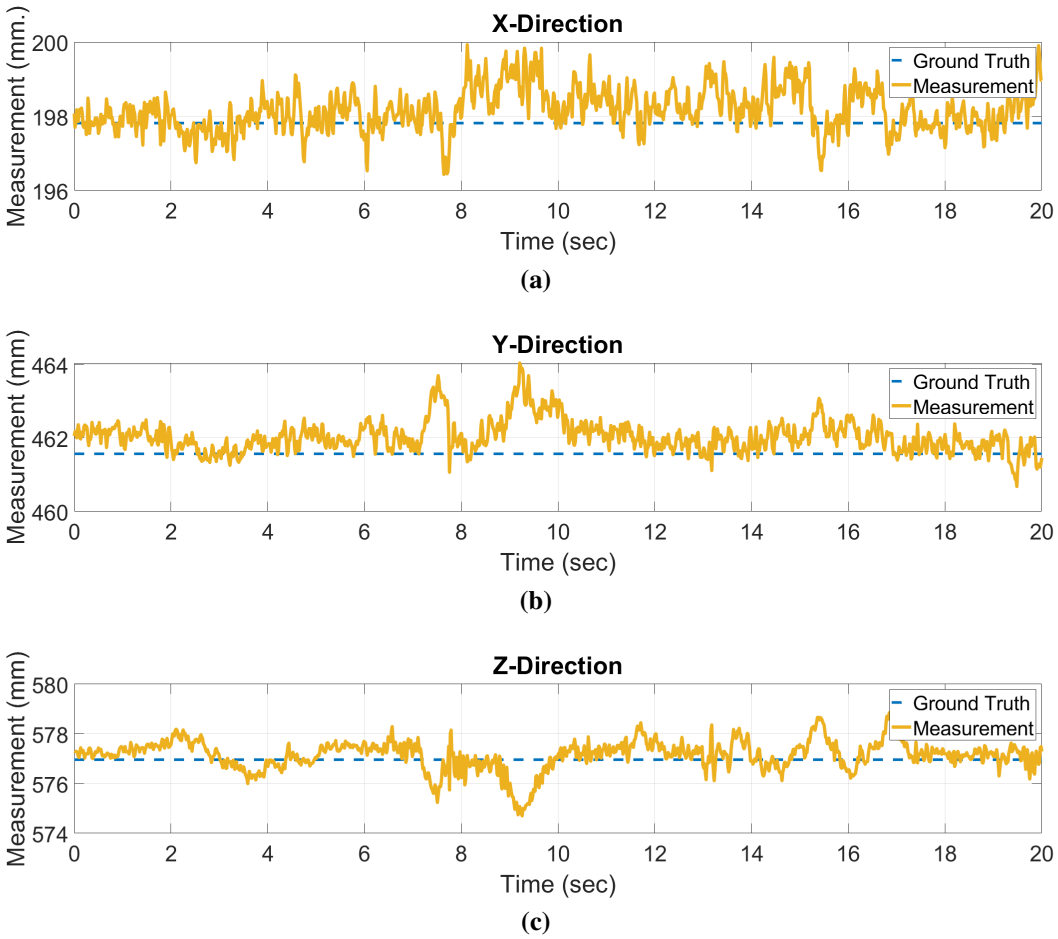


Figure 3.35: Measured location of a stationary point using moving dual stereo camera pairs in the global (a) *X*-; (b) *Y*-; and (c) *Z*- coordinates

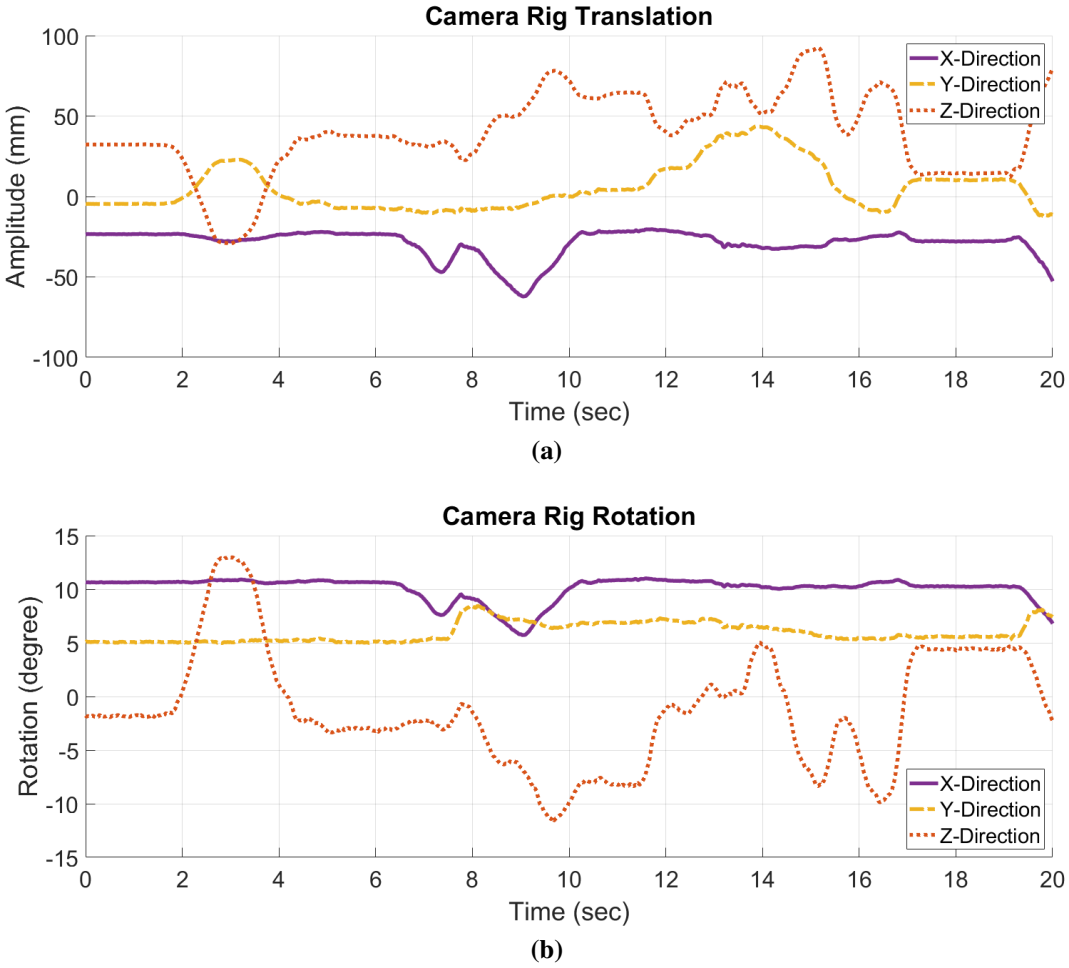


Figure 3.36: The captured motion of the camera rig during the station point test: (a) 3-DOF translation of the camera rig; and (b) 3-DOF rotation of the camera rig

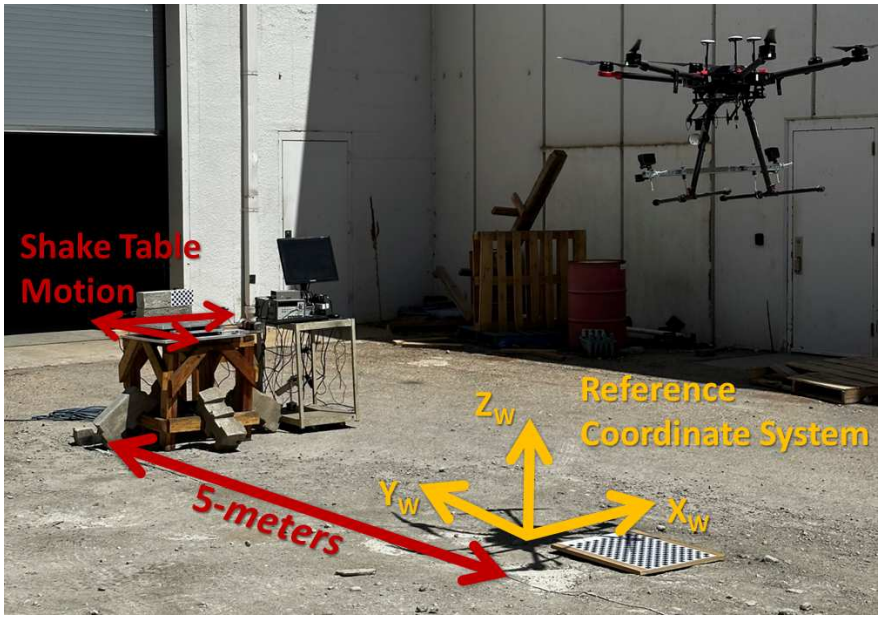
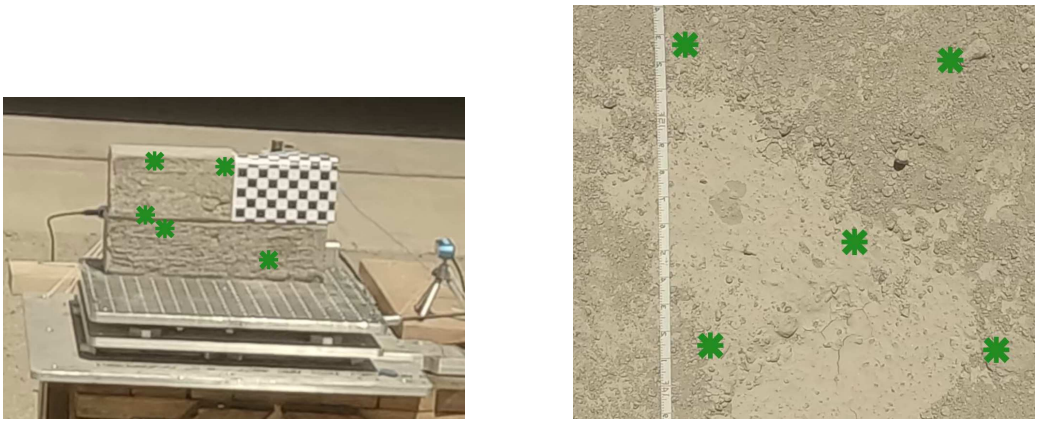


Figure 3.37: UAS-based lab experiment with shake table motion



(a) **(b)**

Figure 3.38: The tracked natural features for the: (a) ROI measurement area; and (b) REF measurement area

The directions of motion of the shake table did not perfectly align with the reference coordinate system on the ground (Figure 3.37). To be able to compare the ground truth from the shake table with the motion measured with respect to the reference coordinate system, the reference coordinate system was transformed to align with the motion of the shake table. The transformation matrix was solved by finding the dominant direction of motion of the measured data through singular value decomposition such that $\mathbf{G} = \mathbf{U} \times \mathbf{S} \times \mathbf{V}^T$, with

$$\mathbf{G} = \left[\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{3\times n} - \begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \end{bmatrix}_{3\times 1} \right] \times \left[\begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{3\times n} - \begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \end{bmatrix}_{3\times 1} \right]^T \tag{3.33}$$

The dominant direction is described by the right singular vector, $\mathbf{V_1}$, corresponding to the largest singular value, S_1. Knowing the dominant direction, the transformation matrix is solved directly using Equation 3.30. Specifically, the dominant direction forms $\hat{\mathbf{u}}$ in Equation 3.30, while $\hat{\mathbf{v}}$ and $\hat{\mathbf{w}}$ are determined using the same approach described in the "Dual Stereo Vision Measurements" section. Lastly, a high-pass filter and low-pass filter were applied to the measurements. These two filters removed the high-frequency variations due to the slightly inaccurate subpixel tracking and the low-frequency variations due to the non-compensated UAS motion. The results for the test are shown in Figure 3.39 with the ground truth signal and tabulated in Table 3.10. Note that the shake table did not move in the *Z*-direction. The RMS error of the measured motion and the ground truth is 2.54-*mm* and 5.77-*mm* in the global coordinate system in the *X*- and *Y*-directions. In addition, the 3-DOF translational motion and 3-DOF rotation of the UAS are plotted in Figure 3.40. Note that the measured dynamic displacement (Figure 3.39) and the captured motion of the UAS (Figure 3.40) show little correlation demonstrating the motion of the UAS are removed successfully from the measured displacements. It is also worth noticing that the magnitude of the UAS motion is quite significant, on the order of ± 250-*mm* and $\pm 5^\circ$, while the magnitude of motion of the concrete beam is only 10 to 50-*mm*. Being able to measure the relatively small amplitude of displacements with the significant UAS motion has demonstrated the prowess of the proposed technique. The

power spectral densities of the measured 3C displacements are shown in Figure 3.41. The percent difference of the estimated frequencies to the ground truth is shown in Table 3.10. Note that due to the relatively short signals (less than 10-*sec*), the errors in frequency estimation are partially attributed to the short window effect on the Fourier transform.

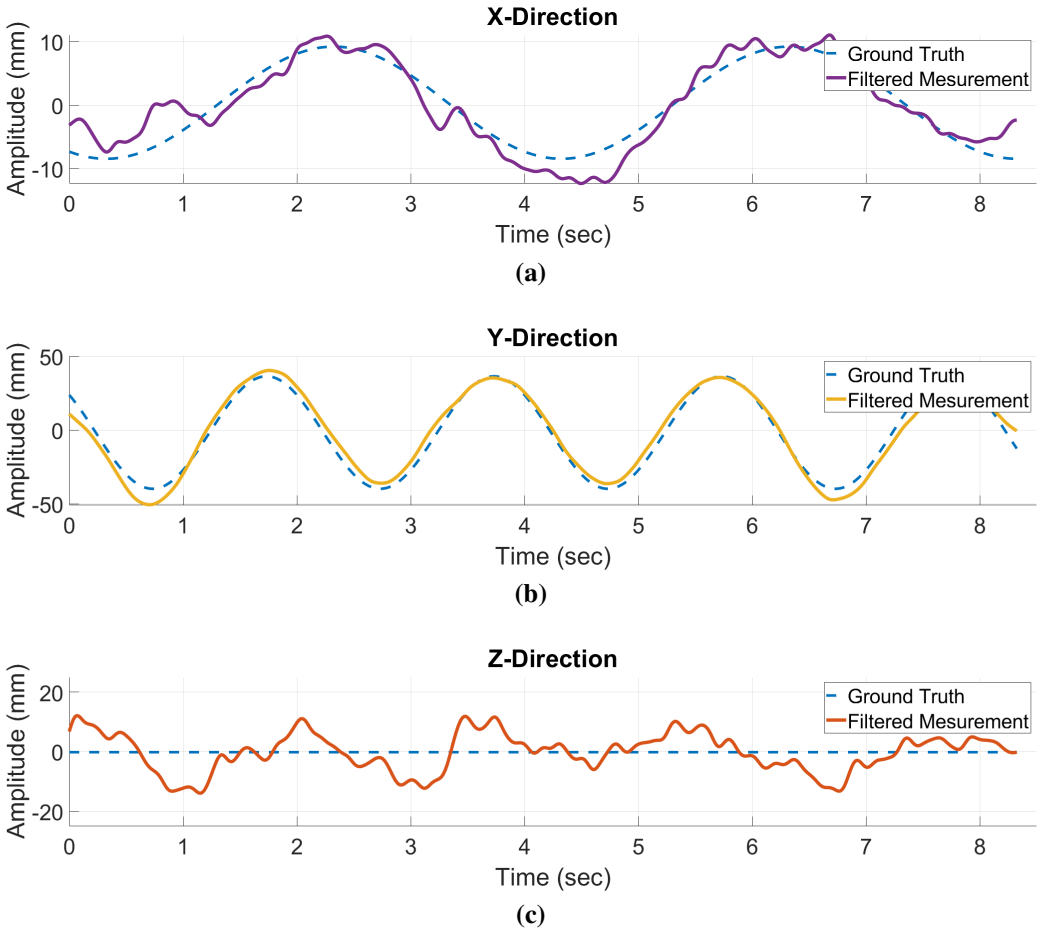


Figure 3.39: Comparison of the measured motion of the concrete on the shake table against the ground truth measurements in the (a) *X*-direction; (b) *Y*-direction; and (c) *Z*-direction

Discussion

Measurements of 3C dynamic displacements were taken of a concrete ROI using cameras attached to a UAS. The proposed system demonstrated sub-millimeter accuracy, but errors were slightly increased when the camera rig was flown farther from the ROI in the field test. This in-

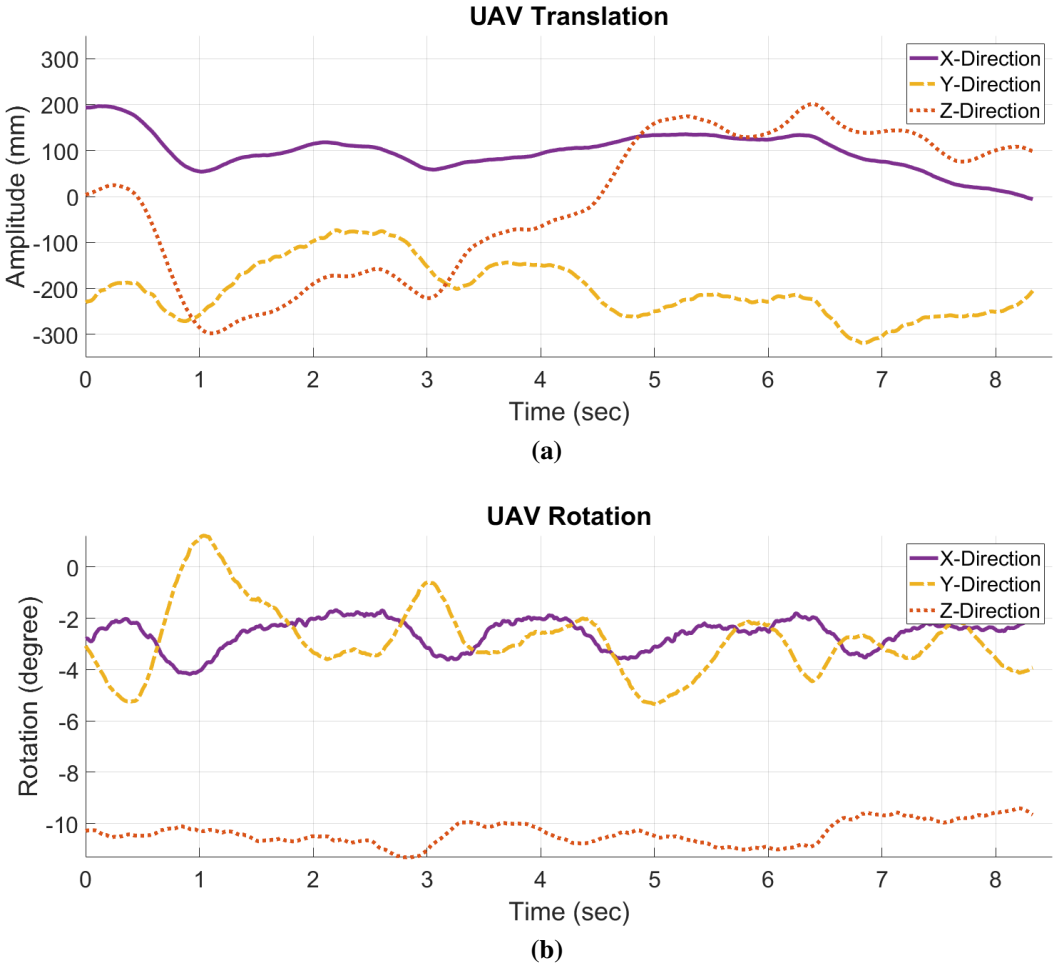


Figure 3.40: The captured motion of the UAS during flight: (a) 3-DOF translations; and (b) 3-DOF rotations

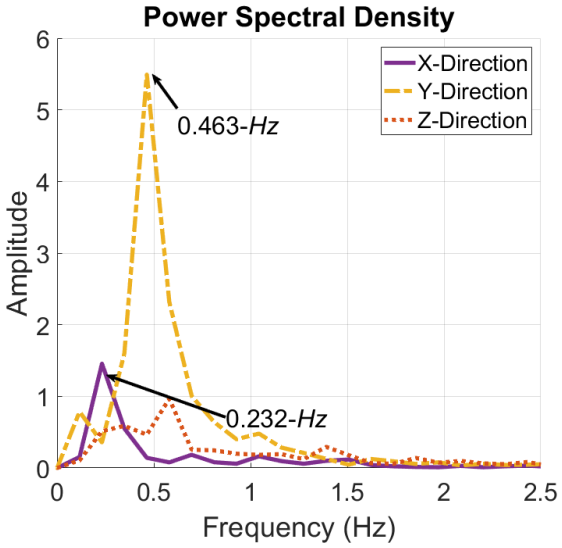


Figure 3.41: Power spectral density of the UAS-enabled 3C dynamic displacement measurements

Table 3.10: Results of UAS-based 3C dynamic displacement measurements

	Dual Stationary Point Lab Measurement RMS Error (mm)	UAS-Based Measurement RMS Error (mm)	UAS-Based Frequency Percent Different
X-Direction	0.635	2.54	5.86%
Y-Direction	0.898	5.77	5.67%
Z-Direction	0.607	6.40	-

crease in error is likely attributed to two related factors: inaccurate subpixel tracking of natural features and reduced GSD at greater distances. The theoretical minimal error based on GSD is about 2-*mm*, and the proposed technique was able to achieve errors close to this benchmark. Based on the lessons learned from this study, there are some considerations for future improvements and applications:

1. The measurement errors in engineering units are extremely sensitive to the pixel errors of the entire system. Therefore, the performance of the cameras (in terms of the GSD) and the tracking technique directly affect the accuracy of displacement measurements.

2. It is crucial to ensure that the four-camera rig attached to the UAS is rigid. Slight movements of the cameras during flight or transport may lead to the change of relative locations of four cameras and de-validate the camera calibration, therefore, causing significant errors in the measurements.

3. Synchronization of the cameras is critical. The errors are significant if the cameras are out of sync by half of a frame (about 0.010-*sec* in this study). Extra caution is needed when using software synchronization. Whenever possible, it is recommended to implement hardware synchronization of the cameras.

4. It is important to estimate the GSD of the camera systems before any application, as the GSD dictates the theoretical minimum error and the longest distance from the camera to the structure that leads to acceptable measurement errors.

Conclusions

This study presents a novel dual stereo vision technique to directly measure the dynamic displacements of a structure with a UAS platform. With two pairs of cameras attached to a UAS, the 3C dynamic displacement of the structure and the 6-DOF motion of the UAS are simultaneously measured using a mathematically elegant workflow. The following conclusions are drawn from this study.

1. A novel pair calibration technique is developed to precisely determine the relative locations of the four cameras for the challenging condition where the same keypoints are not visible in all four cameras.

2. Using two pairs of cameras allows flexibility in choosing a stationary reference. The two camera pairs can be oriented in different directions to provide the best vantage points for both the ROI and reference, greatly enhancing the feasibility of applying the new technology in various field environments.

3. The 3-DOF translation and 3-DOF rotation motion of the camera rig is successfully recovered by finding a transformation matrix from the global coordinate system to the local coordinate system of REF cameras.

4. Natural features can be identified using minimal user input and tracked providing the ability to take measurements anywhere without the use of artificial targets, making the proposed technique viable for difficult-to-access structures.

5. The efficacy of the proposed technique was demonstrated via experiments, where the RMS errors of the measured displacement time histories at a 5m distance from the cameras to the structure are 2.54mm and 5.77mm in the X- and Y-directions of the global coordinate system, which corresponds to a mean average percent error of 22% and 11%, respectively. This accuracy level was achieved when measuring displacements with amplitude in the order of 10 to 50mm under UAS motion in the order of $\pm 250mm$ (translation) and $\pm 5^{\circ}$ (rotation). The

ability of this technique to measure small-amplitude dynamic displacement with significant UAS motion allows the applicability of this UAS-based remote sensing technique to civil structures under challenging environmental conditions such as high winds.

6. The 5m camera-to-structure distance achieved through this study is considered a significant improvement compared to that of the existing study (0.75m) [60]. The area of the ROI captured by the cameras at this 5*meters* distance is 5.9m $\times$ 10.6m. This working distance and camera field of view have demonstrated the feasibility and applicability of the proposed technique on a large scale for many civil structures.

3.7 Task 6: Component Identification

3.7.1 Task 6: Motivation

With segmented components (i.e., bridge deck, girder, piers, etc.) within images collected with terrestrial or UAS-enabled photography, bridge managers will not only be able to facilitate FE modeling by streamlining the materials selections and geometry but also link damage on a component-level and better use predictive modeling to predict the bridge's performance in the future. To find the components of a bridge, previous research has explored using human-in-the-loop machine learning with clustering techniques (i.e., K-means clustering, agglomerative clustering, Gaussian mixture models, etc.) to segment out the components of a bridge directly from the 3D point cloud with a high rate of accuracy [10]. This technique was tested on two concrete bridges. Although the proposed technique requires only 10-30-*min* of time is required to obtain a fully segmented point cloud, there is still user input needed to provide the initial points and supervise the clustering techniques. Moreover, additional processing time is required to label the segments. A more general framework without the need for significant user input still needs to be developed and tested.

To achieve a generalized approach, semantic segmentation from the images is proposed. Semantic segmentation involves classifying each pixel within an image with semantic labels and has been researched in a variety of applications such as crack identification [10, 36, 42], biomedical

image segmentation [43], and scene segmentation for autonomous vehicles [110]. Although training a model for pixel-level semantic segmentation is more difficult, it is advantageous over using other segmentation techniques such as object segmentation where only a bounding box is drawn over an identified object [111]. The basic architecture of the semantic segmentation network used is the U-Net and is used to identify steel fatigue cracks in Section 3.2. The original U-Net design proposed was first presented in [43] for the segmentation of nuclei in biomedical images. Although the general architecture is similar, the proposed U-Net is much different from the original design in that it is deeper and has two encoders that allow for the fusing of the color and depth map images (one encoder to handle the color inputs and one encoder to handle the depth maps). To take advantage of previous research, transfer learning is implemented. A pre-trained classifier network, the Inception ResNet Version 2 (IRNV2), is used as the encoder for the colored images. The IRNV2 was originally developed to classify images from the ImageNet dataset which contained 1,000 image classes [112]. When originally presented, the IRNV2 was a state-of-the-art network to achieve the highest accuracies in the ImageNet dataset and is still one of the best-performing networks. An advantage of the IRNV2, which was exploited in the proposed network, is that it has multi-scale convolutions simultaneously which the authors call "Inception Residuals" which capture features of different sizes, which is an advantage when trying to identify the components of a structure. The skip connections are drawn from layers within the IRNV2 and connected to the decoder. Moreover, the proposed architecture fuses the color image data with depth map images to improve the predictions. The depth map images are generated during the point cloud generation and provide the distance from the camera to pixels within the image. By fusing these two data together, the AI network is able to better predict to which class each pixel belongs. Lastly, once the components are labeled and segmented in the images, they can also be mapped onto a point cloud to provide a component-wise segmented point cloud in addition to the segmented images. This process of mapping the images onto the point cloud can be accomplished by utilizing the camera's poses generated during the point cloud creation using structure-from-motion [10].

This section will first examine the existing literature and techniques to segment a point cloud directly and state-of-the-art semantic segmentation. Next, the proposed AI network is discussed, and the data is presented. Lastly, the results of the training are examined to evaluate the efficacy of the proposed AI network.

Objectives

1. Develop a methodology to allow for component-wise damage information and facilitate the creation of FE models using the 3D point cloud generated with structure-from-motion

2. Identify the performance of an AI network to semantically segment the images collected with a UAS

3. Discuss the applicability of using the network (which is trained with synthetic data) on real-world structures including acquiring and using the depth map images

3.7.2 Task 6: Methodology

Dual U-Net Architecture

All the components used to build the proposed U-Net (i.e., convolution, transposed convolution, pooling, normalization, etc.) have been discussed in Section 3.2 and will not be discussed here for conciseness. The overall network architecture is shown in Figure 3.42. This section will discuss the architecture of each section of the U-Net separately (i.e., color image encoder, depth map image encoder, and decoder). First, the color image encoder is the IRNV2 (dark blue box in Figure 3.42). The weights of the network were initialized as the pre-trained weights from the ImageNet dataset. The last fully connected layers of the IRNV2 were not used since segmentation instead of classification was the purpose of this network. Additionally, the skip connections from the encoder to the decoder (red dashed lines in Figure 3.42) were placed just before the size reduction of the features; therefore, there were 5 skip connections from the encoder. Second, the encoder of the depth map images (bottom left section of Figure 3.42) was a more standard convolutional architecture with a series of convolutions, normalizations, and ReLU activations followed by a max

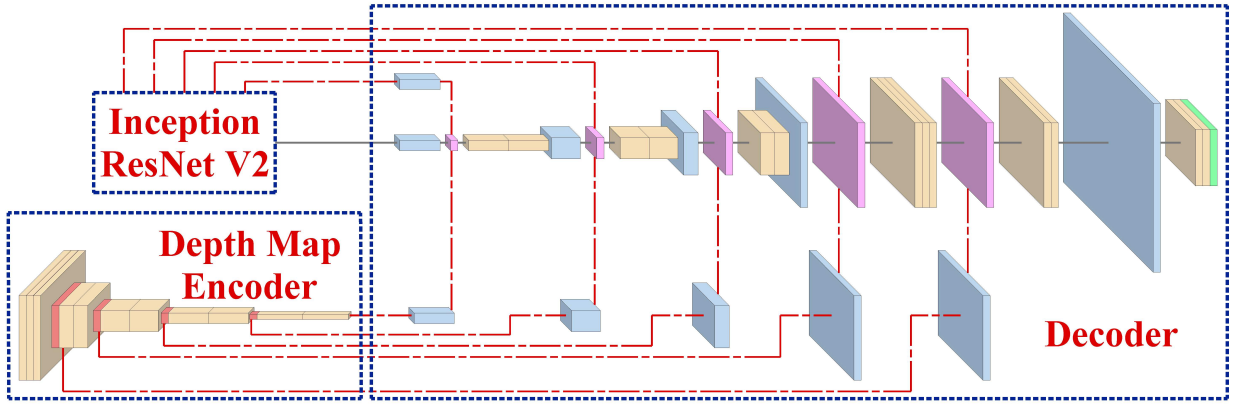


Figure 3.42: U-Net architecture for component detection (tan box: convolution, normalization, ReLU activation; red box: max pooling; blue box: transposed convolution; magenta box: concatenation; green box: softmax activation)

pooling operator. The weights were randomly initialized using a He normal distribution [113]. The skip connections in the depth map encoder were placed just before the max pooling operators and were linked to the decoder through a transposed convolution (red dashed lines in Figure 3.42). The transposed convolution between the depth map encoder and the decoder was needed due to the size difference between the color images and the depth map images (i.e., the depth map images were one-third the size of the color images). Lastly, the decoder of the network gradually up-scaled from the encoder by concatenating the skip connections from the color image encoder, depth map encoder, and previous layer (the previous layer connections are shown as solid gray lines in Figure 3.42). After the concatenation, a series of convolutions, normalizations, and ReLU activations were performed followed by a transposed convolution. Just before the final softmax layer, the feature maps were upscaled larger than the output, and a convolutional layer with a stride of $(3, 3)$ was used to reduce the feature map's dimensions so that the label dimension matched (i.e., the labels were one third the size of the color images).

Training Data and Pre-Processing

To train the network, the data provided by the 2nd International Competition for Structural Health Monitoring (IC-SHM, 2021) was used. The dataset is computer-generated railway viaducts generated in [114]. The viewpoints of the images are that of a UAS. The components were auto-

matically labeled into 7 categories: nonbridge, slab, beam, column, nonstructural, rail, and sleeper. Additionally, damage spots were added to the components with generated damage labels; however, since this paper focuses on component detection, the damage labels were not used. With the dataset, the depth maps of the scene were also given with the focal length of the synthetic camera used. There are 7,575 input images with a resolution of 1920 × 1080-*pixels*. The components labels and depth map images have a resolution of 640 × 360-*pixels* (i.e., one-third the size of the color images). Sample images are shown in Figure 3.43. The dataset has different textures for concrete, metals, etc., and different backgrounds. The data set is unbalanced in terms of the pixel labels. For instance, about 54% of the pixel labels belong to the nonbridge class while less than 1% belongs to the rail and sleeper class. The full distribution is shown in Table 3.11 where the information is used for network optimization. The data was split 20% (1,512 images) for validation and 80% (6,063 images) for training; testing data was provided without labels.

To increase the number of images during training, the data was augmented. About 20% (1,217 images) of the training data was randomly selected for augmentation. Of the images selected for augmentation, the images received both a flip augmentation and color augmentation. For flip augmentation, either a vertical, horizontal, or vertical and horizontal flip was randomly selected. For the color augmentation, either random brightness increase/decrease, random contract increase/decrease, or random principal component analysis (PCA) color alteration [51]. In addition to increasing the number of images in the dataset used for training, augmentation also helps in reducing overfitting of the network by providing additional viewpoints and simulating different lighting conditions.

For the network to be able to handle the input images, the dataset was cropped to 80% of the original size resulting in the input images having a resolution of 1,536 × 864-*pixels* and the labels and depth map images with a resolution of 512 × 288-*pixels*. The input image sizing was chosen to ensure that the pixel dimensions were divisible by 96 (i.e., $2^5 \times 3$) since the labels in the provided dataset were one-third the size of the images. No interpolation or resolution alteration was performed to ensure clean labels and depth map images. With the cropping schema, there was

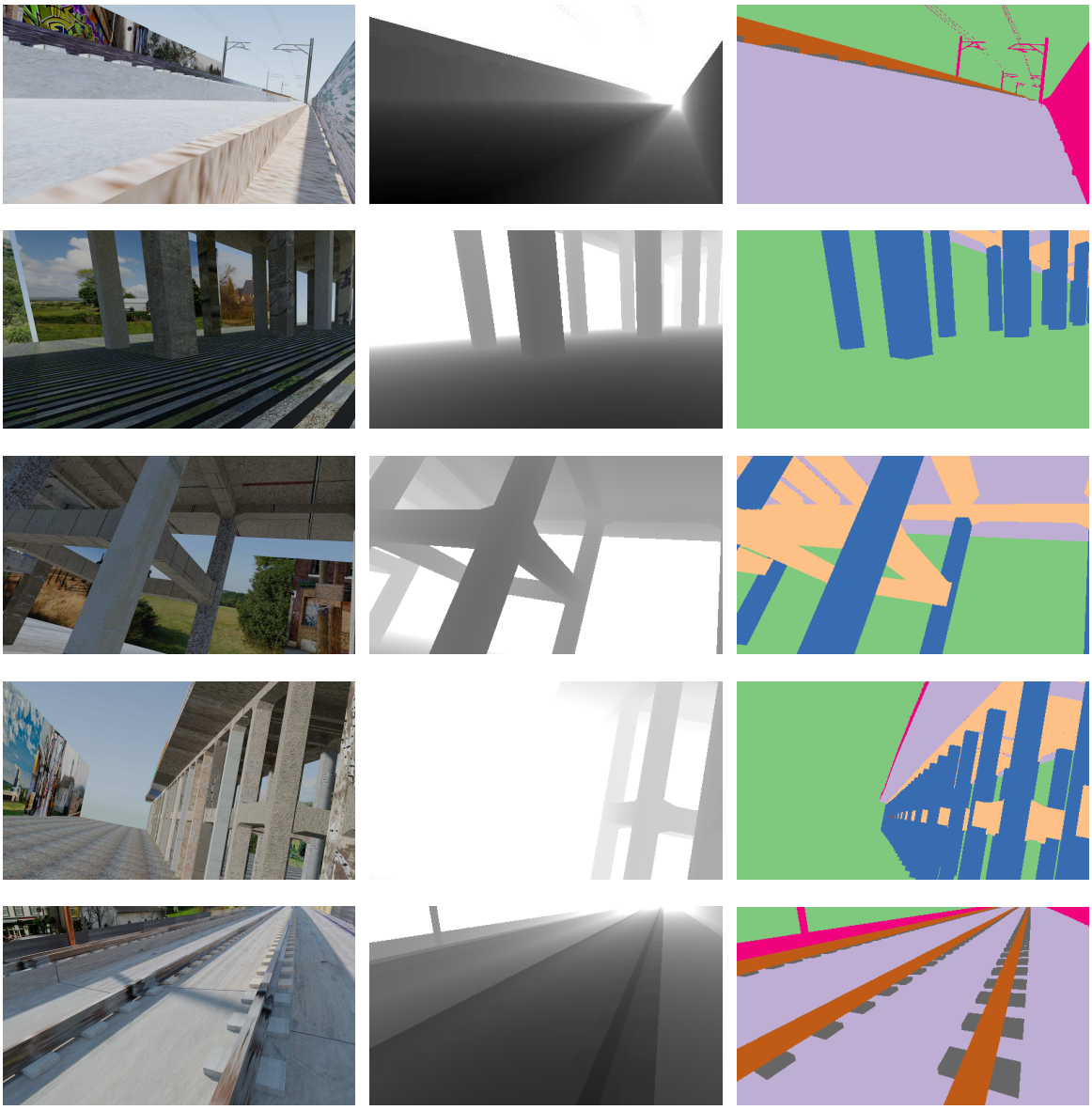


Figure 3.43: Sample testing dataset: Left column is the color images; Center column is the depth maps; and Right column is labeled segmentations

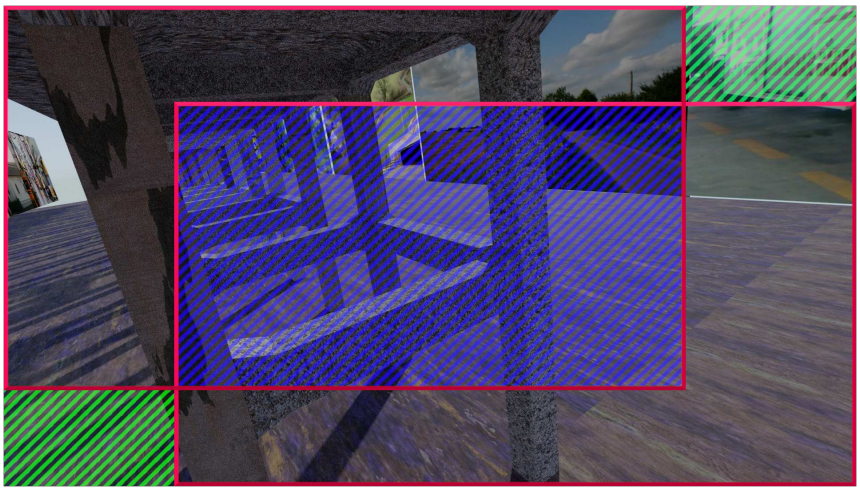


Figure 3.44: 1,536 $\times$ 864-*pixel* cropping schema of full-sized input images (blue center region: region of overlap section; and green corner regions: region of unused section)

a 36% overlap of the cropped section and 4% of the image that was not used during training. The cropping schema is shown in Figure 3.44.

Training and Optimizing the AI Network

The training schema of the network was performed in stages since transfer learning was implemented. Since the color image encoder had pre-trained weights, the weights were initially held constant and not updated after each epoch. During this training, only the weights of the two untrained sections (i.e., the depth map encoder and the decoder) were updated, and the learning rate was relatively larger at 0.001. Therefore, the network will learn how to interpret the feature maps from the pre-trained IRNV2 encoder and make the best predictions possible. After 40 epochs were performed and the weights of the untrained encoder and decoder were learned, the entire network was trained for 20 epochs with a smaller learning rate of 0.0005 to fine-tune the final results. As a note, the parameters of normalization (i.e., mean and variance) were not updated during this fine-tuning stage as it is suggested in the literature that updating these parameters could produce worse results. The training was performed on an Nvidia Tesla A100 GPU with 40-*GB* of GPU RAM. The code of the network was written in Python with Tensorflow [48]. An Adam optimizer was used. Since the pixel labels were significantly unbalanced, a weighted categorical cross entropy

(CCE) loss function was used. The class weights for the loss function, w_i, assigned to the weighted average loss function are shown in Table 3.11. The weights were found using,

$$w_i = \frac{N_{Classes}}{N_{Classes} \times N_{Instances}} \tag{3.34}$$

where, $N_{Classes}$ is the total number of classes and $N_{Instances}$ is the total number of instances the class appears in the training dataset.

The same metrics used in Section 3.2 for the defect identification are also used in this network to measure the performance. In addition, the mean intersection over union ($MIoU$) is also used to measure how well the pixel label predictions match the correct labels. This metric is advantageous to use when training a multi-class segmentation network for it measures the area of overlap of the predicted label and the ground truth label and is calculated as,

$$\text{MIoU} = \frac{1}{k} \sum_{i=1}^{k} \frac{|\tilde{y}_i \cap y_i|}{|\tilde{y}_i \cup y_i|} \tag{3.35}$$

where, k is the number of classes, $\tilde{y}$ is the predicted class, and y is the ground truth label. An $MIoU$ of 1 is a perfect prediction.

Table 3.11: Class distribution and weights

Class	Distribution	Class Weight
Nonbridge	53.93%	0.26570
Slab	12.97%	1.1042
Beam	11.98%	1.1582
Column	17.35%	0.82332
Nonstructural	2.94%	5.1786
Rail	0.657%	21.274
Sleeper	0.184%	79.350

After the training was completed on the synthetic data set provided by IC-SHM, the network was trained on a small amount of real-world data to extend its applicability to real structures and

explore the robustness of the network on real-world data. Images were collected for a concrete double tee bridge in Loveland, Colorado, USA with a UAS. Over four hundred images were collected of this bridge and a 3D point cloud was generated through structure-from-motion [10]. As a corollary to the point cloud creation, depth maps were also generated for each image. Twenty of the 448 images of the bridge were selected and manually labeled with semantic labels to identify the components of the bridge. This bridge had five classes: non-bridge, slab, beam, column, and nonstructural. A sample of the collected bridge data is shown in Figure 3.45. The same cropping schema, augmentation, and training/validation split were performed on the real-world data as with the synthetic data. With this small sample set, the network was refined by training this new data. The last one-third of the layers (i.e., the last 300 layers) were updated with a small learning rate of 0.0005. As a note, the structure-from-motion-generated depth maps are not as pristine and precise as the synthetic data; however, the depth maps still have the component information required for the network. Also, with the additional training, the network is updated to learn how to interpret the less precise, structure-from-motion depth maps.

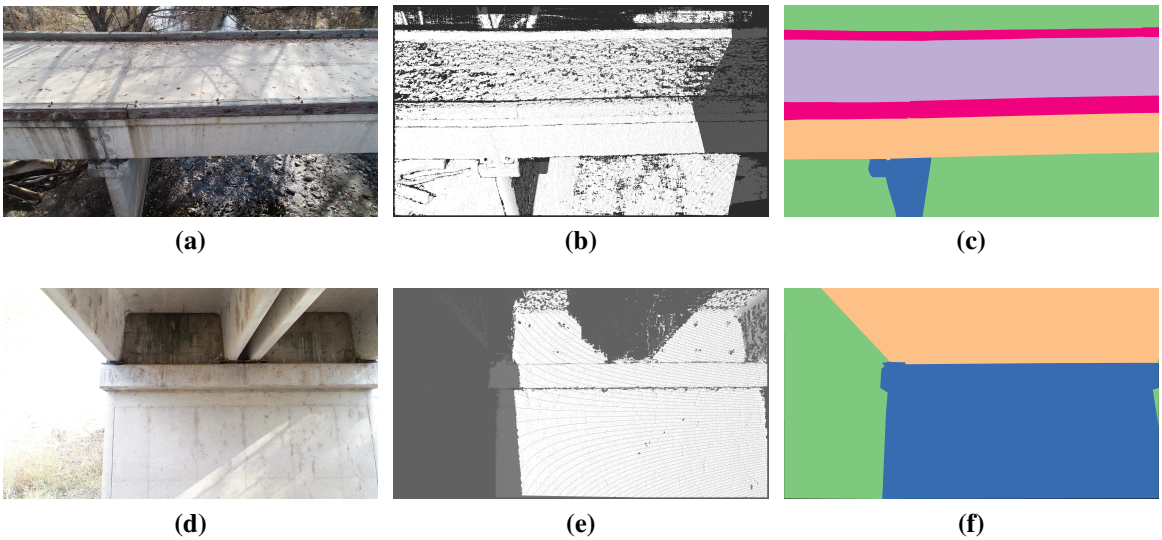
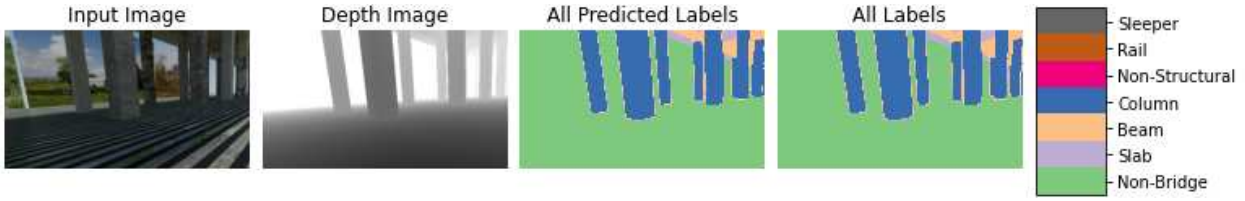


Figure 3.45: Sample real-world data collected of a concrete bridge; data included color images, depth maps, and semantic labels

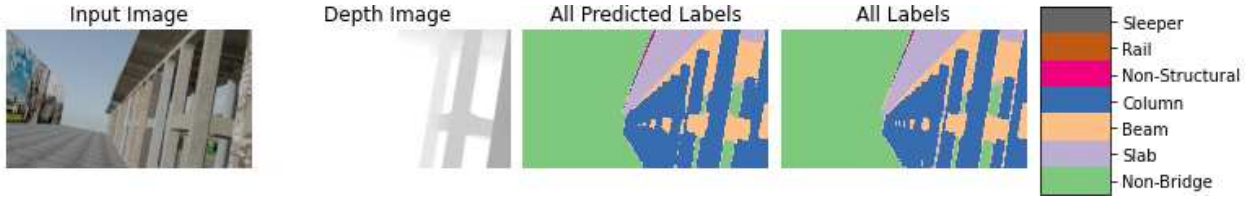
3.7.3 Task 6: Results

Results of Synthetic Training

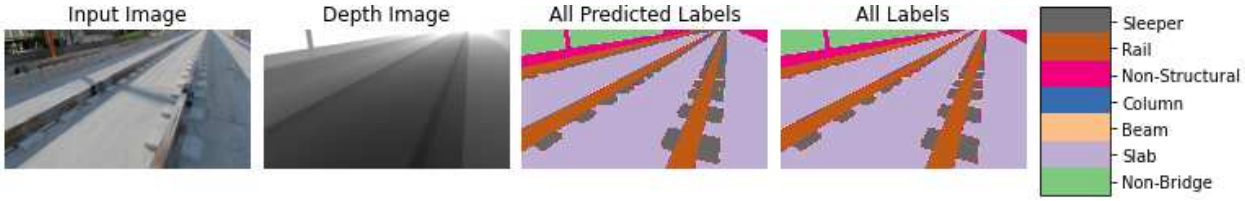
The proposed AI network achieved a high accuracy and $MIoU$ score on the validation data. The results of 3 samples from the validation data are shown in Figure 3.46 and the metrics for the validation data are shown in Table 3.12. The loss and $MIoU$ over the number of epochs are given in Figure 3.47 and Figure 3.48. From Figure 3.47 and Figure 3.48, there is little evidence of overfitting since the validation curves for the $MIoU$ and loss both approach an asymptote. Using the depth map data provided more information to the network. Incorporating this additional information, better network performance was achieved when compared to only using the color image input. In practice, the depth maps can be generated during the process of creating the point cloud with structure-from-motion [10] or using stereo vision sensors such as Intel Realsense [60].



(a)



(b)



(c)

Figure 3.46: Sample validation results for component detection

Table 3.12: Validation metrics

	Validation Data Set	Real-World Images
Accuracy	99.15%	96.80%
Recall	99.15%	96.80%
Precision	99.15%	96.80%
F_1	99.15%	96.80%
$MIoU$	99.04%	96.41%
Loss	0.0628	0.157

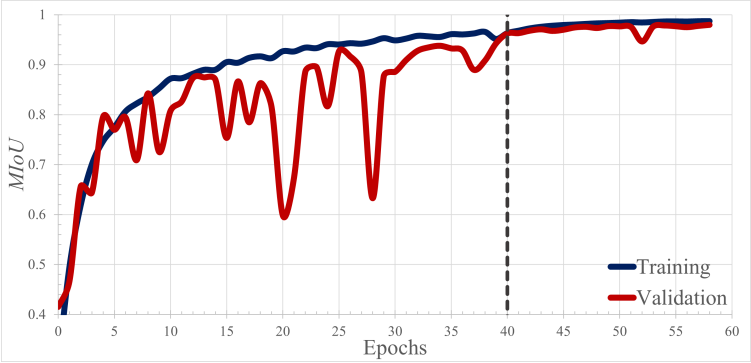


Figure 3.47: $MIoU$ during training (dashed gray line represents where the entire network training began)

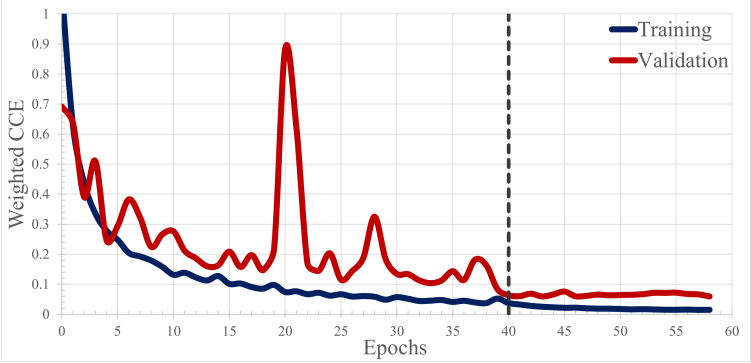


Figure 3.48: Weighted categorical cross-entropy loss during training (dashed gray line represents where the entire network training began)

Results of Model Updating for Real-World Data

After proving the high performance of the network on the synthetic dataset, the efficacy of the network in identifying components from real-world images is also evaluated. A high accuracy was achieved on the validation data (Table 3.12) by training the 20 images of a real concrete bridge for 40 epochs. To further test the network performance, a sample of additional images of the bridge, which the network has not seen, was also passed through the network to generate labels. These images are shown in Figure 3.49. It is seen that the components were successfully identified. The high performance of the proposed network with a small number of additional images shows that an inspector or bridge manager is required to only manually label a subset of the images, update the network, and then the AI can provide the identified components on the entire dataset of the bridge. Ultimately, as this process continues and the network evolves and learns the additional data, manually labeled data will not be required for new applications.

Conclusions

Component identification is an important module within a DT to help facilitate element-wise metrics and automated FE model generation. In this task, an AI network was designed and first trained on synthetically generated images. The proposed AI network features data fusion of both color images and depth map images, which enables the identification of the components of a bridge with a high accuracy (in terms of $MIoU$). After the initial training using the synthetic images, the network was updated with additional training data of images of a real-world concrete bridge. By only updating the network with a small subset of the images, the network was able to accurately segment the components of a bridge on unseen, real-world images.

3.8 Task 7: Connection Identification

3.8.1 Task 7: Motivation

Identifying the connections within a structure can help to streamline the automated FE modeling of a bridge and make the implementation of DTs simpler in practice. The intent of this task

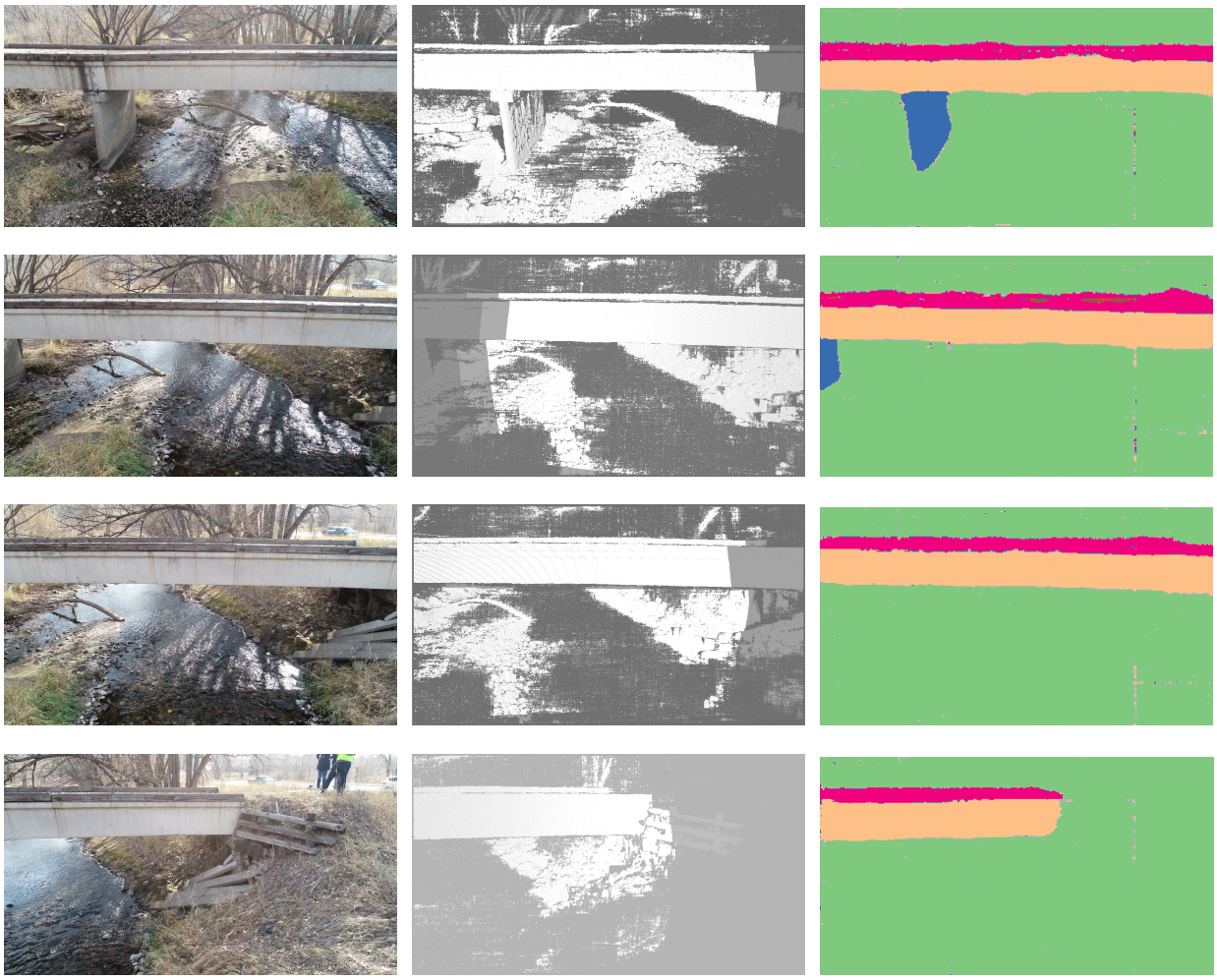


Figure 3.49: Results of real-world testing dataset: Left column input color image; Middle column structure-from-motion-generated depth maps; Right column predicted label

is to automatically localize and classify a connection (i.e., pinned, roller, moment-resisting, etc.) without the need for user input. This task of identifying the connections of the bridge is accomplished through an AI classification network. This classifier network will provide the connection labels needed in the FE and DT modeling. Here, a simpler classification network, compared to the binary semantic segmentation network for defect detection in Section 3.2 and the multi-class semantic segmentation network for component detection in Section 3.7, is selected. Using the proposed classifier network, one can generate a heat map of the locations of different types of connects. This type of classifier network provides a more efficient training and data processing schema. Two networks are built and trained to handle steel and concrete bridges, respectively.

In the following, the two network architectures and training processes are discussed. A specific cropping technique to generate the heat maps of the connection location and type is introduced next. Lastly, the results of the training are presented with the techniques for implementation.

Objectives

1. Build a classifier network to decide if an image contains a connection or not
2. Identify the connection types of a bridge
3. Develop two networks to handle both steel and concrete bridges
4. Train and validate the network with openly available data sources

3.8.2 Task 7: Methodology

In this study, two separate networks are designed and trained to handle connection classifications of steel and concrete bridges. First, the general network architecture for connection classification is discussed. The specific network extensions to handle steel and concrete bridge connections in each of the respective networks are presented here too. Next, the cropping schema of the data is discussed which is necessary to generate the heat map images of the connection locations. The collected data used for training is discussed last in this section.

Network Architectures for Connection Classification

For connection localization and classification, a simpler image classifier network was chosen. The advantages of using a classifier network are that 1) it does not require as much training data as a network for semantic segmentation, and 2) it is much more efficient at passing data through the network when compared to semantic segmentation. To build upon existing research, transfer learning was again implemented to serve as the foundation of the network. The Inception ResNet V2 network was used as the base model for transfer learning [112]. This network was chosen because of the high accuracy in classifying the ImageNet dataset and the good performance for identifying components of a structure in Section 3.7. An input size of 256×256 was chosen for the Inception ResNet V2 because it is a power of 2 (i.e., 2^8) and it was the optimal input size for defect identification of steel members (Section 3.2). The details of the components of the network are discussed in Section 3.2. From the output of the Inception ResNet V2, five additional layers were added to help classify the features. First, a convolutional layer with a size of $(3, 3)$ and a stride of $(1, 1)$, with normalization and a ReLU activation is added. After this, a layer with a size of (m, n, c) is flattened to a vector with a size of $((m \times n \times c), 1)$, and each element of the flattened layer is connected to a dense layer with 512 nodes. In a dense layer, each connection between the previous layer and the dense layer has a weight that is learned during training. A second dense layer and softmax activation are added at the end of the network to provide the probability distribution of the prediction of the class. The network architecture for the steel bridge connection identification is shown in Figure 3.50.

A two-step training schema was used which is typical when implementing transfer learning. The learned weights of the Inception ResNet V2 when originally trained on the ImageNet dataset were uploaded to the network. Since additional layers were added to the network, initially, only these added layers were trained using the bridge data to directly interpret the learned features of the Inception ResNet V2. This training lasted 50 epochs with a larger learning rate of 0.0001. After this initial training, fine-tuning of the network was performed, and the last one-third of the layers (the last 260 layers) were trained. This was performed with a smaller learning rate of 0.00001. All

the training was done using an *Adam* optimizer with the initial parameters of the added layers set to a He normal distribution [47]. A Python script using the TensorFlow library [48] was used. A Sigmoid Focal Cross Entropy (SFCE) loss was used as it is best for unbalanced datasets [115].

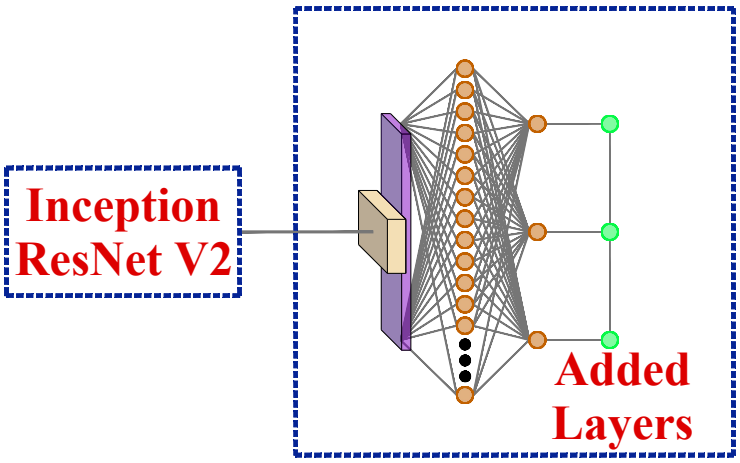


Figure 3.50: Network architecture of steel bridge connection identification with three output classes (tan box: convolution normalization, ReLU activation; purple box: layer flattening; orange: fully-connected dense layer; green: softmax activation)

For concrete structures, two dominant connection types are used in practice: moment connections and slide-bearing (roller) connections. Although pinned connections are used, they are much less common for concrete bridges in Colorado, USA, and, therefore, were not considered. Moreover, a moment connection is not needed to be identified on concrete structures since within an FE model a moment connection is a continuous element. Therefore, the proposed concrete bridge connection identification network only classifies the images as belonging to a "non-slide-bearing-connection" and "slide-bearing connection" class. For steel bridge connections, moment, pinned, and slide-bearing (roller) connections are typical. Again, a moment connection is not needed to be identified for an FE model; therefore, three classes are needed to be identified with the AI network with identification classes of "neither-roller-nor-pin-connection," "roller connection," or "pinned connection." Similar network architectures are used for the two AI networks except for the final layers. For the concrete bridge network, the output (final layer) has two prediction nodes for the

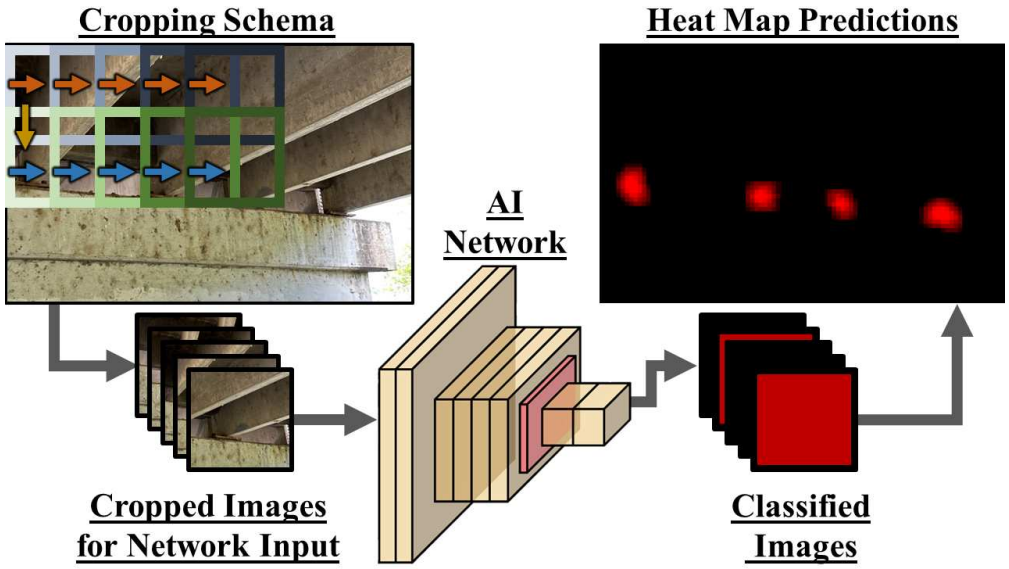


Figure 3.51: Cropping, classification, and restitching schema to generate the connection location heat map

binary classification task; for the steel bridge network, the output (final layer) has three prediction nodes for the three classes.

Generation of Heat Map for Connection Localization

Since the connections are typically smaller relative to an entire image, a sliding window cropping technique is used to crop images in small sections. Then, the classification network is used to determine whether a cropped portion of the image contains a certain type of connection. Once the images have been classified, they are restitched together to create a heat map of the locations of connections in the images with more intense color representing the center location of the connection. The heat map image is the same size as the full-sized input image. An example of this cropping, classification, and restitching schema which generates the connection location heat maps is shown in Figure 3.51. With the general location found and the connection type identified, using the camera's pose matrix, the connection label can be mapped onto a point cloud and ultimately facilitate FE modeling (to be discussed in Section 4.2).

Data Pre-Processing and Training

With the network architecture built, the training data is assembled. There is no published dataset with connections of a bridge labeled for either steel or concrete bridges; therefore, data were manually collected and/or labeled. Two datasets were assembled: steel bridge connections and concrete bridge connections. For the steel bridge connection dataset, the openly-available dataset "Common Objects in Context for Bridge Inspection (COCO-Bridge)" was used [1]. This dataset consisted of 774 images with less than a 1-*MP* resolution. The images are of steel members and bridges taken with a UAS and a handheld camera. The dataset was originally built to train a network to draw bounding boxes around a variety of identifiable features. Since only connection identification through classification was the goal of this project, the provided labels in the COCO-Bridge database were not used. The images were scaled up by a factor of two using linear interpolation of pixel values to increase the size of the images. Next, the images were cropped to a size of 256×256, the input size of the network. There was a 50% overlap of the cropping schema to provide additional training data. Lastly, the images were manually classified into three classes resulting in 9,326 "neither-roller-nor-pin-connection" images, 1,079 "roller connection" images, and 279 "pinned connection" images. A sample of the full-sized images from the COCO-Bridge dataset is shown in Figure 3.52.

Next, the concrete dataset was assembled. Three data sources were combined to create this training dataset. First, 72 images were collected from an open-source online library of street-view images from Mapillary.com shown in column 1 of Figure 3.53 [116]. These images were collected of bridges across Colorado, USA. These samples were of varying size and resolution with between 2-3-*MP*. Next, 575 images were collected of concrete bridges around Fort Collins, Colorado, USA by the author, as shown in Column 2 of Figure 3.53. The images were collected with a cell phone at various zoom levels (i.e., wide-angle, 2X zoom, and 3X zoom) and a resolution of 12-*MP*. The images were also collected at similar angles and positions as a UAS flight. Lastly, 405 images were collected of joints of three concrete parking structures around Fort Collins, Colorado, USA by the author, as shown in column 3 of Figure 3.53. These images were also collected with a cell phone

Figure 3.52: Sample full-sized images for the identification of steel bridge connections [1]

at various zoom levels and had a resolution of 12-*MP*. Although parking garages were not the structure type of focus for this AI network, these images provided more samples of slide-bearing joints that would be found on a concrete bridge. The larger, 12-*MP* images were scaled down by a factor of 2 using linear interpolation so that the scale of all the images from the three data sources would be similar. After scaling, the images were cropped to a size of 256×256 with a 50% overlap between the cropped images. After cropping, the images were manually classified, resulting in 36,247 "non-slide-bearing-connection" class images and 2,388 "slide-bearing" class images.

With all the data assembled, the data is augmented to increase both the number of training samples and the variety of samples. Data augmentation has been shown to improve results in training an AI network in both Section 3.2 and Section 3.7. The same data augmentation technique as in Section 3.7 was used with about 25% of the training data augmented with a color augmentation and an image transformation. The data was randomly split into validation and training datasets (20% for validation and 80% for training).

3.8.3 Task 7: Results

Both networks were trained and evaluated. The validation loss and validation accuracy over each training epoch are shown in Figure 3.54. After fine-tuning, it is seen that the accuracy and loss of the validation dataset improved in both steel and concrete datasets. The final accuracy and loss of the validation dataset for the steel and concrete bridges are shown in Table 3.13.

Table 3.13: Validation metrics for connection identification in steel and concrete bridges

	Validation Dataset	
	Steel Bridge	Concrete Bridge
Accuracy	95.6%	99.2%
Recall	95.6%	99.2%
Precision	96.5%	99.2%
F_1	95.3%	99.2%
Loss	0.0577	0.0275

Figure 3.53: Sample full-sized images for the identification of concrete bridge connections

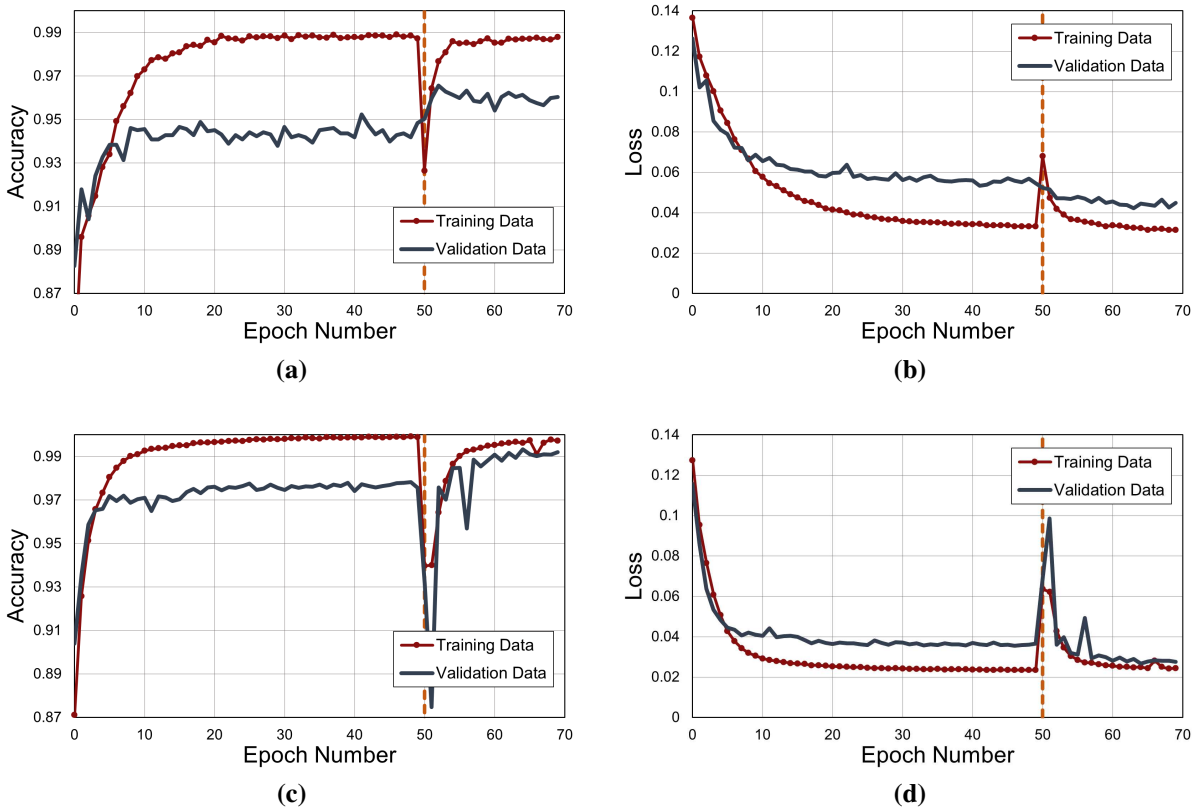


Figure 3.54: Accuracy and loss plots for connection identification on steel bridges (top) and concrete bridges (bottom); Vertical orange dotted line represents the switch from initial training to fine-tuning

The use case for this network is to create a heat map image with the location and type of connection highlighted. A sample heat map of the identified connection classes for steel connection is shown in Figure 3.55. Recall that to generate this heat map, the full-sized images were cropped into 256×256 size images and served as input to the network. To increase the resolution of the heat map, there was a higher overlap cropping rate of 90% (compared to 50% for training). Figure 3.55 shows a high performance of the network to identify both roller and pin connections on a steel bridge. Table 3.13 shows the accuracy of the validation dataset was 96.2%.

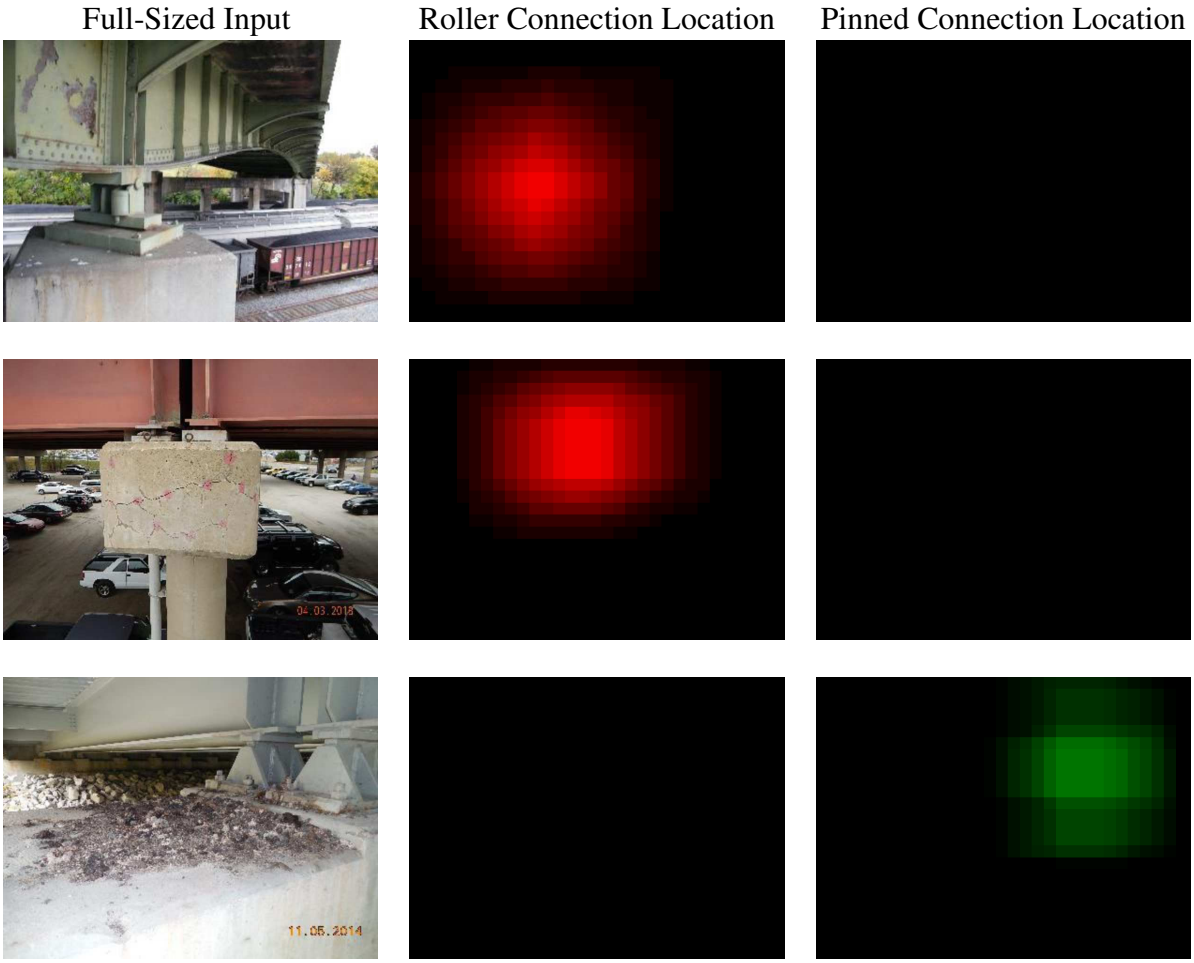


Figure 3.55: Sample heat map output of steel bridge connection identification from the validation dataset

For the concrete bridge network, a sample heat map of the identified slide-bearing connection locations is shown in Figure 3.56. Since the resolution of the concrete bridge images is larger

compared to the steel image dataset, the size of the hot spot on the heat map in the output is smaller. The images of the bridge in Figure 3.56 were collected during a UAS flight. This bridge was not included in the training and validation datasets, thus the images have not been seen by the network. These images served as testing data. From Figure 3.56, the network successfully identified the location of the slide-bearing connection from images. The accuracy of the validation dataset is 99.2%, as shown in Table 3.13.

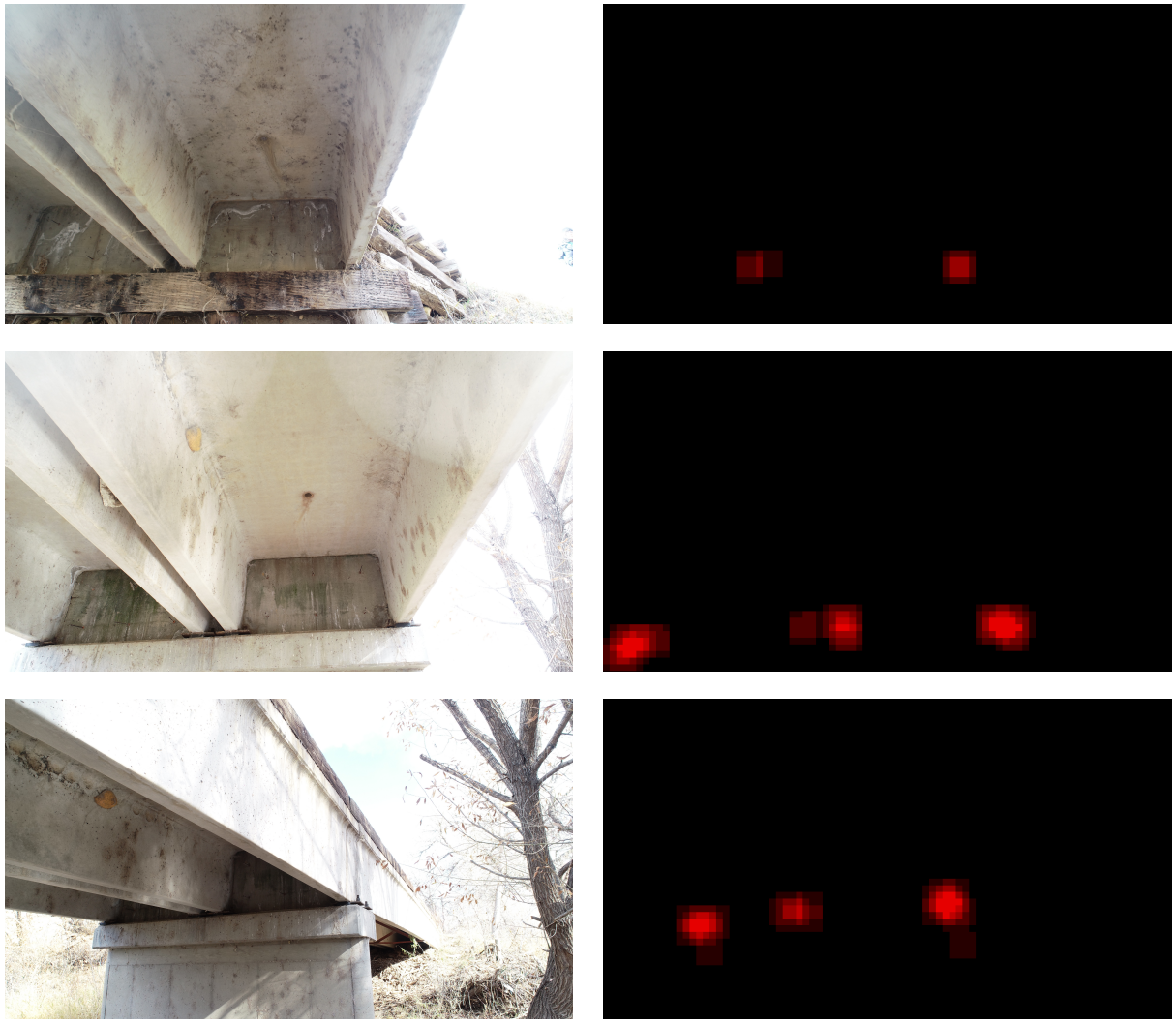


Figure 3.56: Sample heat map of the slide-bearing connections on additional test bridge data

3.9 Summary of Data Analytics

In this chapter, the data analysis was performed. The cracks in steel members were identified with an AI network, and a methodology was presented to track the growth of defects over time. A 3C dynamic displacement measurement technique is proposed to track the dynamic displacements of a structure. Three methodologies were explored with the most advanced technique of using a UAS-enabled dual stereo vision technique capable of measuring full-scale structures with a high accuracy. These 3C measurements can be used to validate an FE model or further manipulated to find anomalies within the structure. Lastly, the components and connections within a structure are identified through separate AI networks to facilitate the automated construction of an FE model from the 3D point cloud. This data analysis is a vital step in linking the raw data collected with a UAS to the model libraries to be discussed in the next section.

Chapter 4

Model Library

4.1 Introduction

The next module within the DT is the *model library*. Within the model library, a variety of models are available to further process and fuse the data within the DT. The model library is the backbone of the DT as it allows for simulations, assessments, and predictions of the structural performance to facilitate decision-making. In this chapter, two models will be discussed. First, a workflow to automatically build an FE model by fusing various types of data (e.g., point cloud, images, etc.) is introduced in Section 4.2. Next, a surrogate model is built by fusing the detected defects and the structural loads obtained with the FE model, detailed in Section 4.3.

4.2 Task 8: Automated FE Modeling and Updating

4.2.1 Task 8: Motivation

Providing a Finite Element (FE) Model on a bridge-by-bridge basis would empower bridge managers and decision-makers to make more informed decisions and perform what-if simulations; however, constructing an FE model from 2D CAD drawing and updating the model from "as-designed" to "as-built" is tedious and time-consuming, and, therefore, costly. Recent advancements in UAS technology have made collecting images trivial. With the collected images, photogrammetry can be implemented to create a 3D point cloud model and 3D photorealistic models [10]. These visualization models can be scaled and geo-referenced to create an "as-built," real-world representation of a structure. However, transforming the 3D point cloud into a solid model that can be meshed and processed with FE analysis software remains a challenge. For a bridge structure, a variety of components and connections need to be handled individually when building an FE model. A point cloud that does not have the component information cannot be directly used in FE modeling.

Previous research has explored using point clouds from both photogrammetry and Light Detection and Ranging (LiDAR) to construct an FE model. Using LiDAR, Tapkin et al. created a 3D point cloud of a masonry bridge to investigate the arch bridge's structural behavior; however, the 3D point cloud was only used to provide measurements, and the FE model was manually built [117]. Using LiDAR data, Hinks et al. implemented voxel sampling to generate an FE model of a facade of a building [118]. Dogan et al. proposed using LiDAR data and voxel sampling to create a solid model and, instead of using an FE analysis, the authors proposed implementing a finite cell method (FCM) to analyze the structural integrity of a cave system [119]. Castellazzi et al. proposed taking the cross-section of a tower building and defining the inside and outside of the cross-section. Then, using voxel sampling, the authors generated a solid model [120]. Voxel sampling has the advantage of being computationally efficient and fast; however, only square-shaped solids can be generated, and it lacks the ability to finely control the fidelity of the model within areas of interest. Kudela et al. used a structure-from-motion-generated point cloud and a space-tree integration sampling technique to directly generate a regular mesh of finite cells [121]. This technique provided more fidelity to the model; however, only cube elements are possible. In addition, these existing techniques may have difficulties in modeling a variety of components in the same structure, especially if the components are not rigidly connected to each other. For instance, a bridge model has many components connected through different support types. There may also be a variety of materials within a bridge model. Modeling connections and assigning different material types to different components may be difficult with current techniques. Ursini et al. created a workflow that integrates LiDAR and structure-from-motion-generated point clouds to generate an FE model using commercial software and manual editing [122]. The proposed workflow fitted the complex geometries of a historical building with non-uniform rational basis spline (NURMS) and planes. Although this process produced an excellent FE model, the computer processing time and user input were significant. Therefore, a more robust technique still needs to be developed to handle a generalized bridge shape. In this context, this study proposes a new technique to automatically build component-wise FE models for bridges from a point cloud.

Objectives

1. Develop a methodology to generate an FE model by fusing different types of data

2. Create an efficient workflow that limits the requirements of significant user inputs

4.2.2 Task 8: Methodology

To facilitate the construction of an FE model, the proposed technique first segments the point cloud into structure components by fusing different types of data, such as images, the 3D point cloud, depth map, and camera poses. The workflow of transforming the 3D point cloud into an FE model follows the workflow in Figure 4.1. First, using the images, the components of the bridge are labeled using the component identification AI network (Section 3.7). The component labeled segmentation maps are projected onto the 3D point cloud using the pose of the cameras. After this projection, the point cloud is segmented into the bridge's components (i.e., girder, pier, etc.). Working with a single component at a time, the cross-section of the component is extracted, outlined, and extruded to the length of the component. This process is repeated for each component and the components are assembled together. This process generates a geometric model that can be uploaded to a variety of FE analysis programs. To assign boundary conditions, the connection identification AI network is used (Section 3.8) to identify connection type and location from images. The localized connections are then projected onto the point cloud to be easily referenced while assigning the boundary conditions to the bridge model. Lastly, the typical FE analysis workflow is followed depending on the software used (i.e., meshing, load placement, output definitions, and processing). In the following, this section will go into detail about each step of the proposed technique.

Generating a Component-Wise Labeled Point Cloud

The point cloud first needs to be segmented into its components. To do this the components of the bridge are segmented out from the raw images of the structure using the technique discussed in Section 3.7. Using the AI network, the images taken with a UAS are passed through the network to generate semantic labels of the components. Next, each semantic label is projected onto the

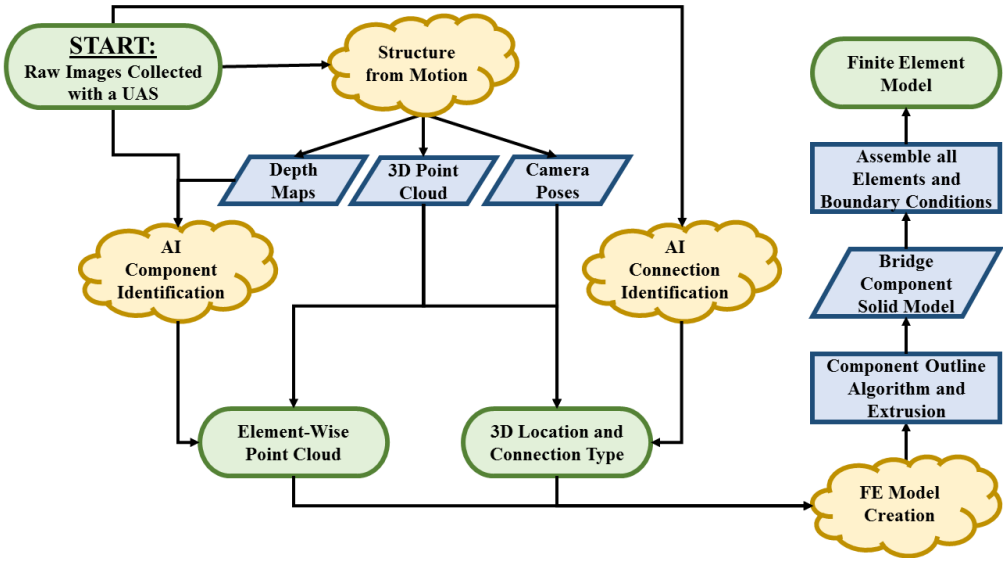


Figure 4.1: Workflow to transform a 3D point cloud to an FE model

point cloud. The technique to project images onto a point cloud was previously developed by the author and published in [10]. The camera poses describing the camera's position and rotation in global coordinates are used for this projection. They are estimated during the creation of the point cloud with structure-from-motion and are composed of a column-vector of the camera position in global coordinates, $\mathbf{C}$, and the rotational matrix of the camera's orientation in global coordinates, $\mathbf{R}_C$ (Equation 4.1). A row of $0, 0, 0, 1$ is added to make the matrix square [123]. This matrix is known and unique for each image.

$$\left[\begin{array}{c|c} \mathbf{R}_C & \mathbf{C} \\ \hline 0 & 1 \end{array}\right] = \underbrace{\left[\begin{array}{c|c} \mathbf{I} & \mathbf{C} \\ \hline 0 & 1 \end{array}\right]}_{\substack{\text{Translation Matrix in} \\ \text{Global Coordinates}}} \times \underbrace{\left[\begin{array}{c|c} \mathbf{R}_C & 0 \\ \hline 0 & 1 \end{array}\right]}_{\substack{\text{Rotation Matrix in} \\ \text{Global Coordinates}}}$$

$$= \left[\begin{array}{ccc|c} R_{c_{1,1}} & R_{c_{1,2}} & R_{c_{1,3}} & x \\ R_{c_{2,1}} & R_{c_{2,2}} & R_{c_{2,3}} & y \\ R_{c_{3,1}} & R_{c_{3,2}} & r_{c_{3,3}} & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4\times4} \tag{4.1}$$

By taking the inverse of the camera pose matrix, the *Extrinsic Matrix*, $\mathbf{E}$, is found. The *Extrinsic Matrix* describes the camera's global position at the time the images were taken. The coordinate system of the *Extrinsic Matrix* is the same coordinate system defined when implementing the structure-from-motion algorithm and is the same as the global coordinate system of the 3D point cloud. The *Extrinsic Matrix* is defined by,

$$\left[\mathbf{E}\right] = \left[\begin{array}{c|c} \mathbf{R}_C & \mathbf{C} \\ \hline 0 & 1 \end{array}\right]^{-1} \tag{4.2}$$

$$= \left[\begin{array}{c|c} \mathbf{R}_C^T & -\mathbf{R}_C^T \cdot \mathbf{C} \\ \hline 0 & 1 \end{array}\right]_{4\times4} \tag{4.3}$$

Lastly, combining the *Intrinsic Matrix* (i.e., the camera parameters found through camera calibration as described in Section 3.6) and *Extrinsic Matrix* (i.e., the global orientation described by the camera pose matrix), the unique *Camera Matrix*, $\mathbf{P}$, for each image can be assembled by

$$\left[\mathbf{P}\right]_{3\times4} = \left[\begin{array}{ccc|c} f_x & s & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]_{3\times4} \times \left[\begin{array}{c|c} \mathbf{R}_C^T & -\mathbf{R}_C^T \cdot \mathbf{C} \\ \hline 0 & 1 \end{array}\right]_{4\times4} \tag{4.4}$$

The *Camera Matrix* describes the transformation from 3D global coordinates to the local 3D camera coordinates described by

$$\underbrace{\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix}}_{\substack{\text{3D Point in} \\ \text{Pixel Units}}} = \left[\mathbf{P}\right]_{3\times4} \times \underbrace{\left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ \hline 1 \end{array}\right]}_{\substack{\text{3D Point in} \\ \text{Global Coordinates}}} \tag{4.5}$$

Next, the points are flattened from 3D coordinates to 2D coordinates by Equation 4.6. This results in a "point-image" in pixel units where the 2D points have the same size as the original image. For instance, if the original image is 3,000×4,000 pixels, the projected "point-image" will be 3,000×4,000 units. Equation 4.6 solves for the $x_1^{'}$ and $x_2^{'}$ of each point and gives the final results of the projection of the 3D point cloud to the 2D points in pixel units.

$$\underbrace{\begin{bmatrix} x_1^{'} \\ x_2^{'} \end{bmatrix}}_{\substack{\text{2D Point in} \\ \text{Pixel Units}}} = \begin{bmatrix} \frac{x_1}{x_3} \\ \frac{x_2}{x_3} \end{bmatrix} \tag{4.6}$$

A relationship between the pixels in an image and the 3D points in the point cloud is created. This relationship allows for an AI-generated label to be readily projected onto the point cloud. The result of this projection is shown in Figure 4.2. Figure 4.2a is the image taken during the UAS inspection. Using the camera pose and camera parameters (Figure 4.2b), the point cloud is projected into the same plane as the image plane (Figure 4.2c). Figure 4.2a and Figure 4.2c are both in pixel units. Next, the AI-generated label image (Figure 4.2d) is overlayed onto the "point-image" (Figure 4.2e). Since both the label image and the projected points are in pixel units, the point cloud points are easily assigned to the labeled classes (Figure 4.2f). Once the points are labeled, they are projected back into the 3D global coordinates (Figure 4.2g). By repeating the process for different segmentation maps to cover the entire bridge, the point cloud is fully segmented, as shown in Figure 4.2h. Note that the time to process the point cloud segmentation using one segmentation map is about 1-*sec* with a typical office computer running a single core. During a typical UAS-enabled inspection, an 80-90% overlap of the images is recommended [10]. Therefore, only about 20% of the image labels must be projected onto the point cloud. With about 500 images taken to generate the point cloud in Figure 4.2, only about 100 segmentation maps are processed to segment the entire point cloud. The entire process is about 2-*min*. This short process time has demonstrated the efficiency of the proposed algorithm.

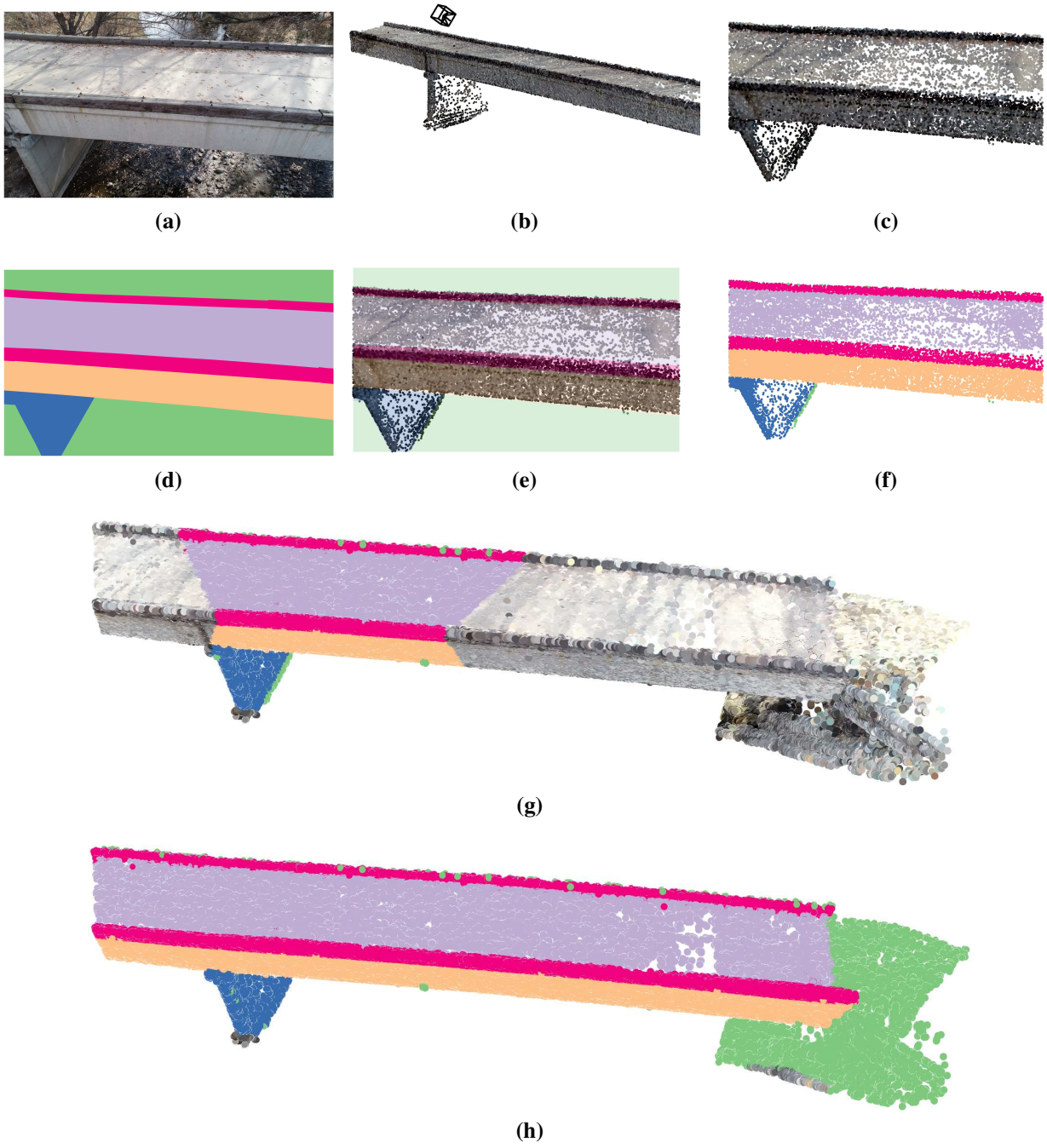


Figure 4.2: Sample process to label a point cloud using an AI-generated segmentation map and camera pose matrix composed of $\mathbf{R}_C$ and $\mathbf{R}$; (a) Input image collected by the UAS; (b) Identified camera pose; (c) 2D "point-image" in pixel units; (d) AI-generated segmentation map; (e) Overlay of AI-generated segmentation map over the 2D "point-image" in pixel units; (f) Labeled "point-image;" (g) Reprojection of the AI-generated segmentation map onto the full point cloud; and (h) Full component-wise segmented point cloud from AI-generated segmentation maps

Defining the Cross-Section of the Components

After the component label projection, the 3D point cloud becomes segmented into its structural components and is labeled based on the AI network. With the segmented components, the next step is to identify the cross-section of each component. It is important to work with each component individually as the dominant direction of the component may be different for each component (i.e., the cross-section of a pier goes up and down while the cross-section of a girder may go left-to-right). The steps to identify the cross-section and build the bridge's component are presented in Figure 4.3. First, the dominant direction of the component is found through *principal component analysis* (PCA) (Figure 4.3a). Next, the point cloud is flattened along its dominant direction, and the general shape of the cross-section is realized (Figure 4.3b). This flattened cross-section is noisy due to the imperfect dominant direction identification and inconsistent point cloud points; therefore, the cross-section is re-sampled by generating a histogram of the point locations. This 3D histogram technique identifies the distribution of the points along the cross-section. By setting a lower bound of the histogram distribution, the cross-section is averaged which filters out the outlier points. The filtered points are shown as black points in Figure 4.3b. With a filtered cross-section, the points need to be enclosed; however, this is a nontrivial problem since the points are not ordered. A novel technique is developed to tightly enclose the points with a set of piece-wise functions. First, an initial set of functions is defined by using an alpha-shape (Figure 4.3c), which is a set of linear piece-wise functions. A maximum curve length is defined which sets how tightly the alpha-curve wraps around the points [124]. This process quickly encloses the cross-section; however, as it is seen in Figure 4.3c, the alpha-shape does not tightly enclose the points. This is especially evident at the corners of the structure (Figure 4.3c). Preliminary tests using only the alpha-shape resulted in significant errors of the estimated stresses to a ground truth model (on the order of a 50% difference). To more tightly enclose the points, the nearest neighbor of the quarter-points of the alpha-shape line segments is identified. Specifically, each alpha-curve is linearly sampled at the quarter points of each line segment. Then, the nearest neighbor of the filtered points at the quarter points is found. This creates a tightly enclosed shape around the points; however,

only the points on the outer edge are used. Therefore, the final step is to erode the shape using a square structuring element with a size of 10 (Figure 4.3d). The erosion operation is defined as,

$$\textit{erosion}(x, y) = \min_{(x',y'):\textit{element}(x',y')\neq 0} f\left(x + x', y + y'\right) \tag{4.7}$$

where (x', y') is the size of the structuring element, f is the domain of the (x, y) points. The erosion step shrinks the outlining shape inward so that the piece-wise function is within range of all the filtered points. With a piece-wise function defining the shape of the cross-section, it is extruded to the length of the component (Figure 4.3e). This process is repeated for all the components of the bridge to define the geometry of the structure.

Compiling the FE Model

With the extruded components, the geometric model is built. The last step before the FE model generation is defining the boundary conditions and assigning them to the model. To streamline this process, the boundary conditions are automatically localized and categorized through the AI network discussed in Section 3.8. The AI network generates a heat map of the connection type and location. The heat maps are projected onto the point cloud by following the same procedure in "Generating a Component-Wise Labeled Point Cloud" for projecting the component segmentation maps onto the point cloud. The result of the projection of an AI-generated localized connection map projected on the points cloud is shown in Figure 4.4. The geometric model with the localized boundary conditions is uploaded into an FE analysis software. In this implementation, Ansys's *SpaceClaim* and *Mechanical* Software were used [125]. Most FE analysis software can be used as long as they can accept a geometric model built outside the analysis program. The geometric model was exported as an STL file which can easily handle raw, unstructured triangulated surfaces. Once uploaded into Ansys, the geometric model was converted into a solid, meshed, and processed following a typical Ansys workflow.

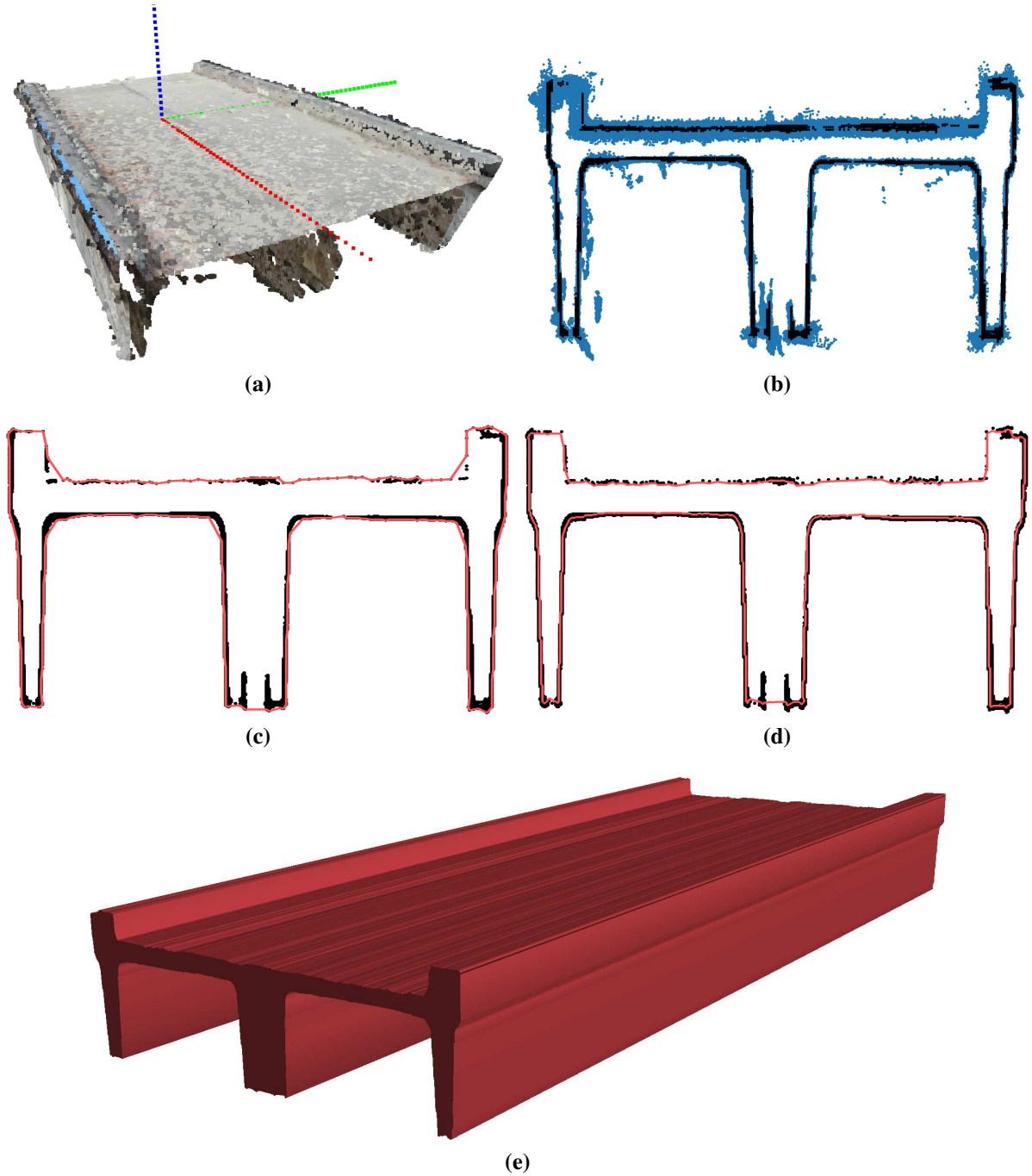


Figure 4.3: Steps to go from the point cloud of a bridge component to an extruded cross-section shape: (a) Point cloud with dominant direction labeled; (b) Flattened cross-section with raw points (blue) and filtered points (black); (c) Alpha-shape (red line) of filtered points (black points); (d) Final cross-section outline with nearest neighbor quarter points added and erosion performed; and (e) Extruded cross-section

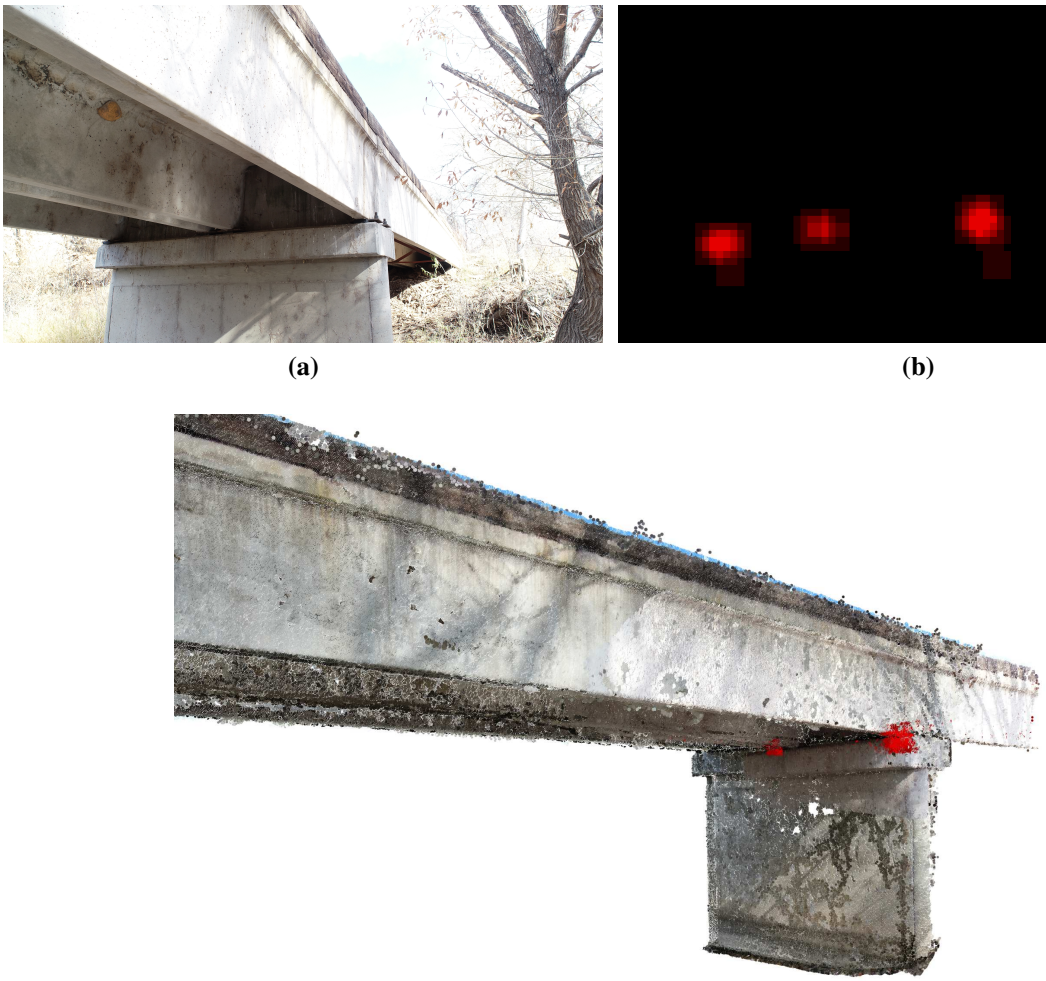
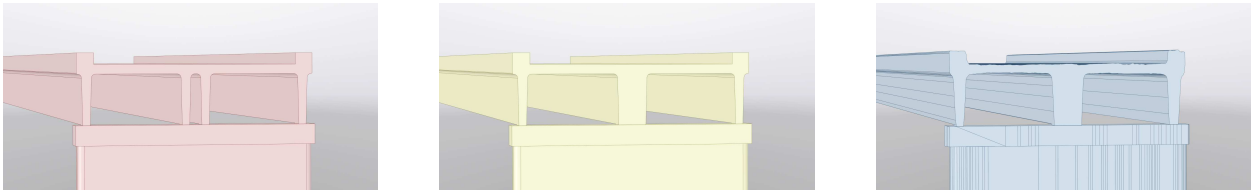


Figure 4.4: Projection of the AI-generated connection location heatmaps onto the 3D point cloud: (a) Image obtained through UAS-enabled inspection; (b) AI-generated label of connection location; and (c) Connection location projected onto full 3D point cloud

4.2.3 Task 8: Results

To assess the performance of the proposed technique, a concrete bridge is used as an example. The bridge is a simply-supported bridge constructed of two precast double-T concrete girders. A model was developed from the point cloud data using the proposed technique and was meshed and analyzed with Ansys. The model was then scaled in the x-, y-, and z-directions to ensure proper sizing of the model. Since the purpose of this study is to prove the concept of building an accurate solid FE model using the point cloud, the modeling of rebars and pre-tension might not be necessary and thus is omitted. The developed geometric model is shown in Figure 4.5. A ground-truth model was manually built using the typical procedure to build an as-built model to compare the proposed technique to traditional FE modeling techniques. To build the ground truth model, manual measurements of the bridge were taken from the field, and a solid geometric model was created in *SpaceClaim*, as shown in Figure 4.5a. Since the space between the double-T girders is not entirely visible by the UAS-enabled camera (see the limits of the black points at the bottom of the cross-section in Figure 4.3c), the proposed technique was not able to outline the space between the double-T girders (Figure 4.5c). Although this error could have easily been corrected in *SpaceClaim*, it was not to test the error of the proposed method with as little user input as possible. To provide a comparison basis that more closely resembles the geometry resulting from the proposed technique (Figure 4.5c), a second ground truth geometric model was also built, where the two center webs of the bridge girder were combined into one larger web (Figure 4.5b).



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

Figure 4.5: Geometric model comparison for proposed workflow: (a) Ground truth geometric model; (b) Ground truth geometric model with center infill; and (c) Geometric model built with the proposed technique from the point cloud

With the geometric models built, the FE analysis was performed. The same material properties, boundary conditions, loads, and meshing technique were used on all three models. The normal stress, Von Mises stress, and total displacements were measured and are shown in Figure 4.6, Figure 4.7, and Figure 4.8, respectfully. A comparison of the results is listed in Table 4.1. The time to build the ground truth model from the field measurements took about twice as long as implementing the proposed technique from the geometric component models in this study. In general applications, this time difference could vary depending on the complexity of the structure, the user's skill, and the familiarity with the software. Note that when comparing the processing time, the steps to assign boundary conditions and loads, define the mesh, and perform the analysis were not included in the timing since these steps were the same for both techniques. To build the ground truth model, significant field measurements were needed and significant model manipulation was required. However, to build the FE model with the proposed technique, it was only required to convert the geometric models to a solid, which is straightforward in Ansys, and assemble the components. From Table 4.1, the model built with the proposed technique consistently has a slightly stiffer performance compared to the other two models. The model built with the proposed technique was still able to get maximum normal stress with about a 30% difference from the ground truth model and about an 8% difference from the ground truth model with center infill. This difference is expected as the point-cloud-generated FE model has additional mass compared to the ground truth models, as seen in Figure 4.9. Table 4.1 shows that the error in modeling the space between the double T girders contributes more to the total modeling error, suggesting that manually correcting this geometric error in *SpaceClaim* might be needed. The proposed technique's model differs from the second ground truth model because the structure's lines and edges are not as neat. This difference is due to the noise of the points in the point cloud, which results in additional material. The mass difference is seen in Figure 4.9b with the model generated with the proposed technique being slightly larger. This modeling error can easily be controlled by supplementing the point cloud data with additional control measurement points. However, depending on the level of

fidelity of a model that is needed, the bridge manager can choose if this additional fine-tuning is required for the bridge.

Table 4.1: Comparison of the results of the three tested models

	Max Normal Stress (MPa)		Von Mises Stress at Center Point (MPa)		Max Deform. (mm)	
Ground Truth Model	5.60	-	4.79	-	1.08	-
Ground Truth Model with Center Infill	4.31	(26.0%)	4.29	(11.0%)	0.861	(22.6%)
Model Built with Proposed Technique	4.00	(33.3%)	3.88	(21.0%)	0.824	(26.9%)

*Numbers in parenthesis represent percent difference from ground truth model

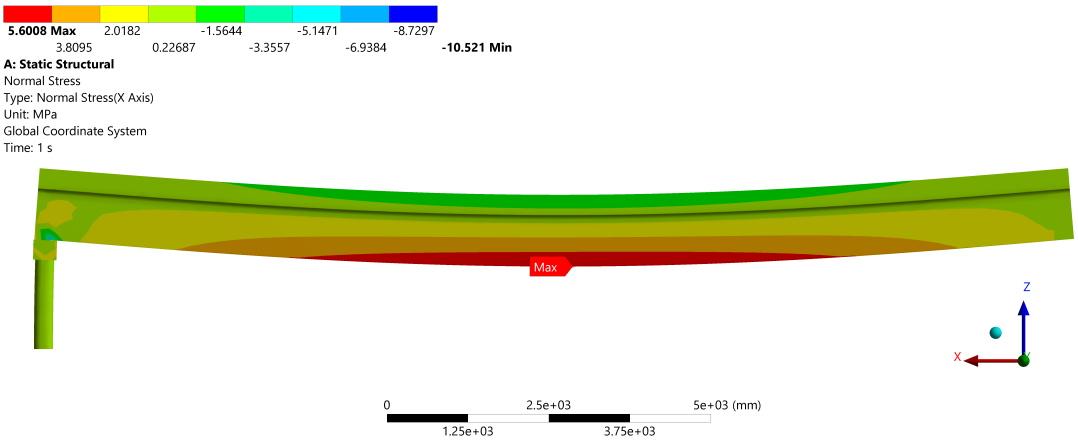
Conclusions

A novel workflow to transform point cloud data into a solid model for FE analysis is proposed. By fusing together AI-generated component segmentation maps and connection locations with a structure-from-motion-generated point cloud, an FE model was automatically built with Ansys. There is a significant user-input reduction from the proposed technique compared with traditional methods to develop an "as-built" FE model. The percent difference from the ground truth model was on the order of 30%, and an 8% difference was achieved when compared to the ground truth model with the center infill. The sources of errors were analyzed, providing the user with information to fine-tune the model based on the desired level of fidelity for different use cases.

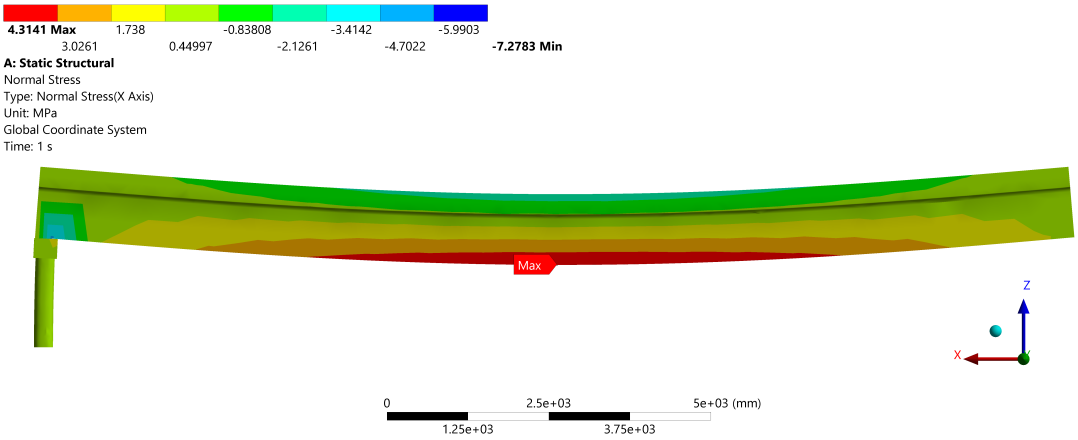
4.3 Task 9: Surrogate Modeling

4.3.1 Task 9: Motivation

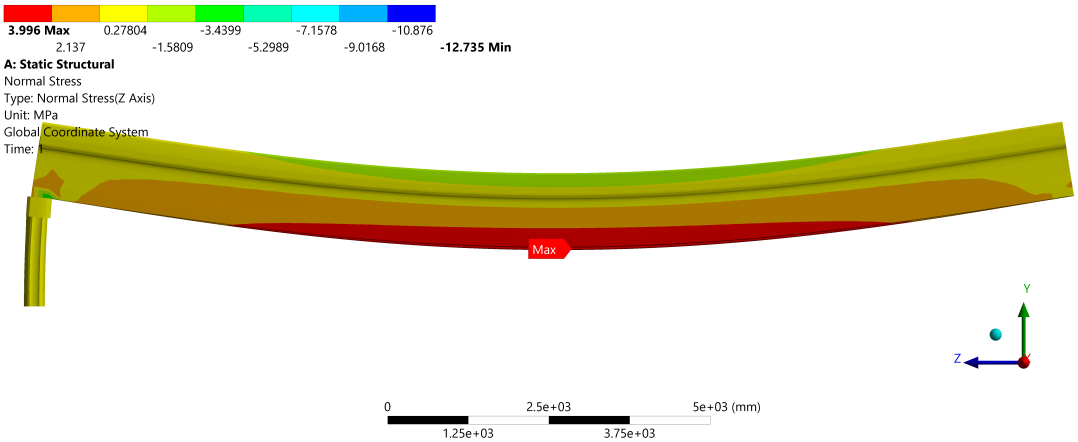
Surrogate modeling is incorporated into the model library of the DT to improve computational efficiency and facilitate rapid decision-making. A proposed surrogate model to be used identifies the stress intensity factor of a steel crack. The process of constructing the surrogate model is discussed in this section. The proposed surrogate model has been published in [10]. After first



(a)

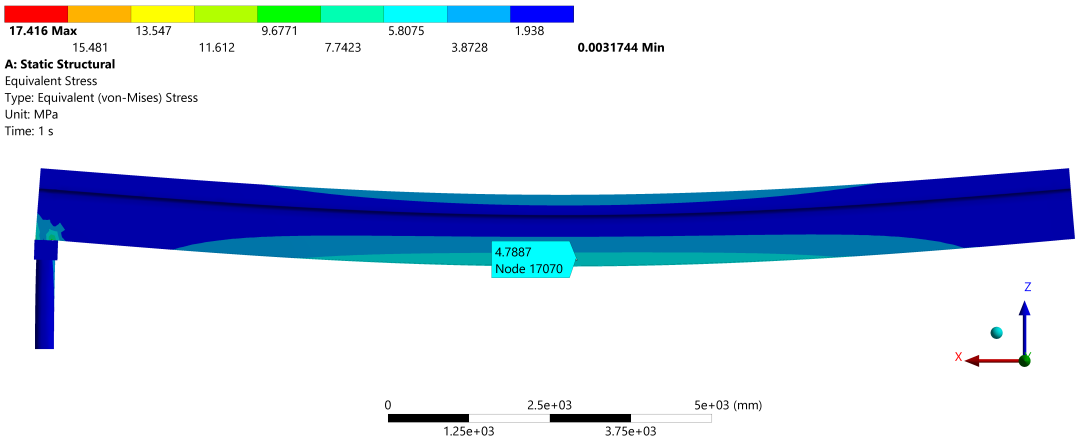


(b)

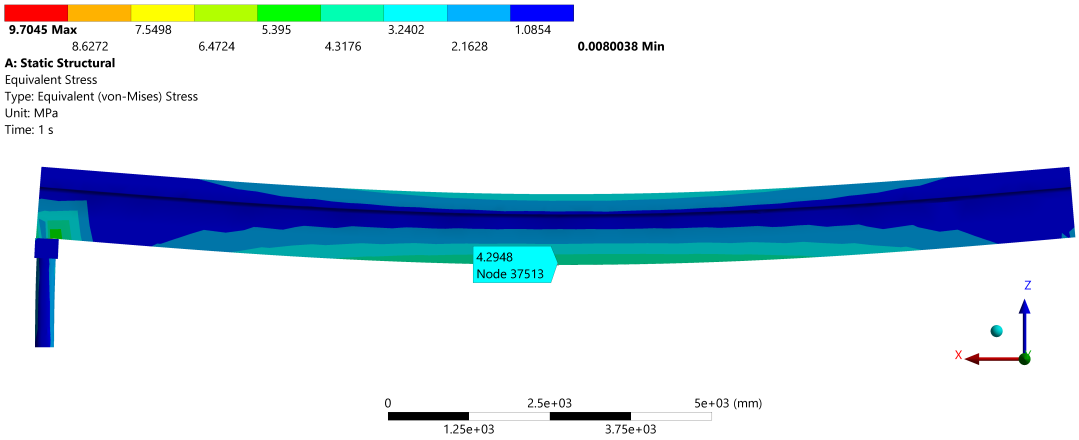


(c)

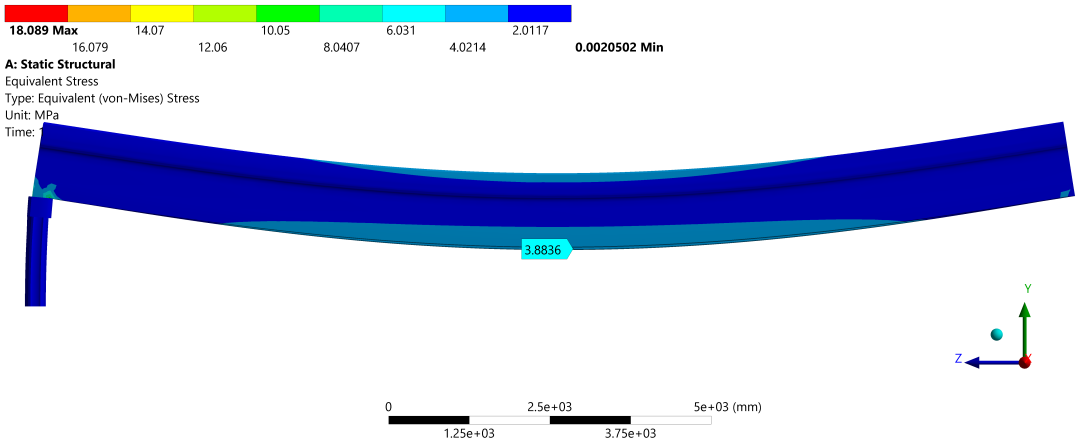
Figure 4.6: Normal Stresses from the FE Analysis: (a) Ground truth geometric model; (b) Ground truth geometric model with center infill; and (c) Geometric model built with the proposed technique



(a)

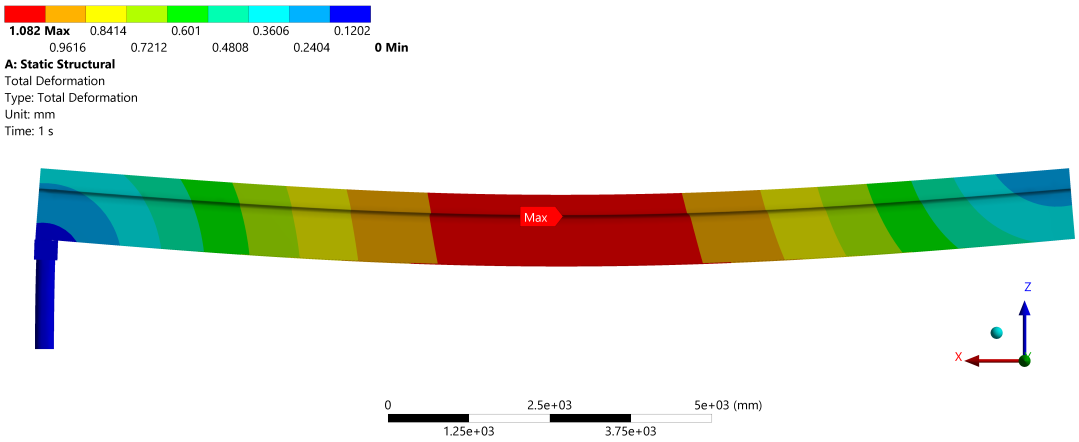


(b)

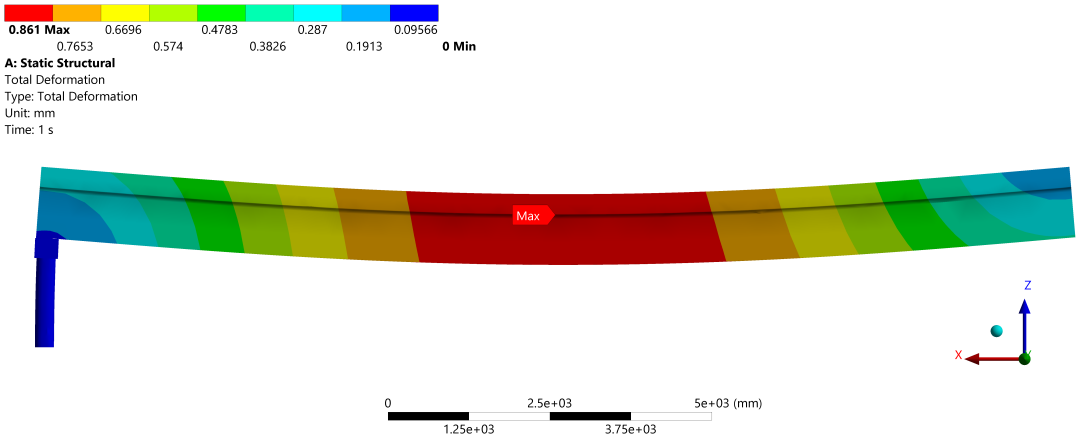


(c)

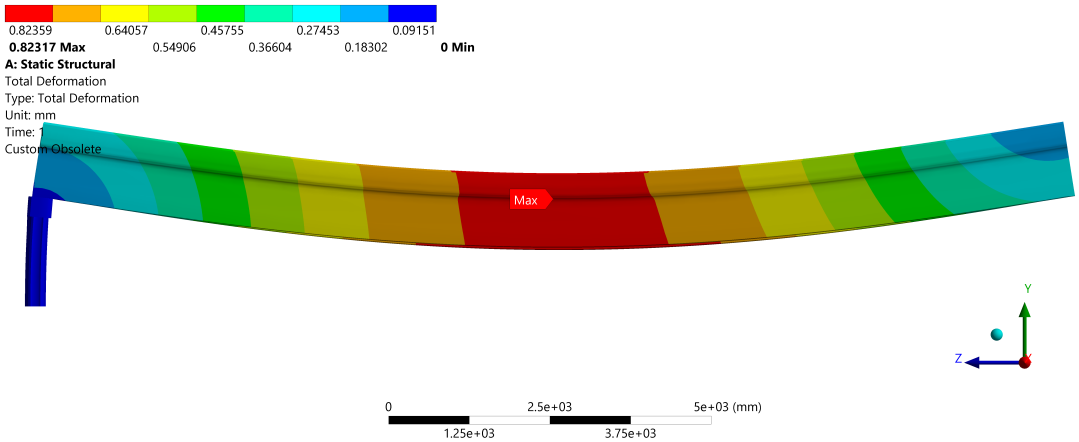
Figure 4.7: Von Mises stress from the FE analysis: (a) Ground truth geometric model; (b) Ground truth geometric model with center infill; and (c) Geometric model built with the proposed technique



(a)

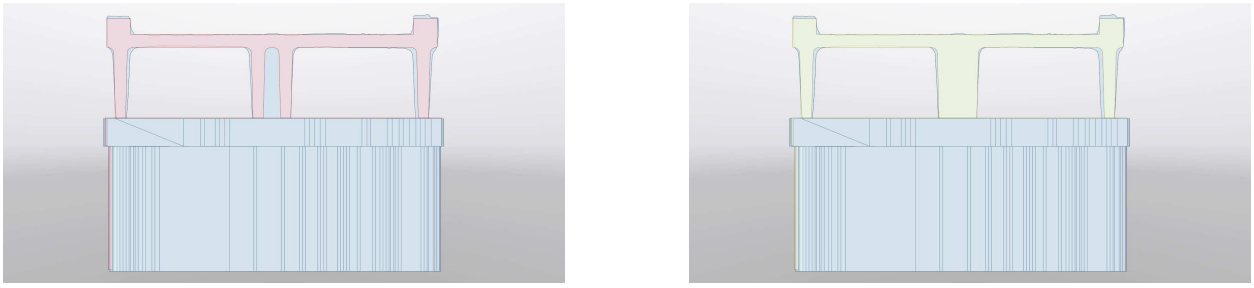


(b)



(c)

Figure 4.8: Total deformation from the FE analysis: (a) Ground truth geometric model; (b) Ground truth geometric model with center infill; and (c) Geometric model built with the proposed technique



(a) **(b)**

Figure 4.9: Comparison of geometric models; Blue model in background is the geometric model developed with proposed technique: (a) Comparison to ground truth model; and (b) Comparison to ground truth model with center infill

introducing the topic of stress intensity factors, the methodology and results of the surrogate model are presented. A full case study implementing this surrogate model is examined in Chapter 5.

With only the data analytics of pixel-level damage, the full potential of the AI platform still cannot be realized in practice due to the lack of sufficient information needed to take immediate action. The executable actions regarding repairs and maintenance require critical information of crack mechanism and proper assessment of the potential for fatigue crack propagation and brittle fracture, which cannot be directly informed from a detected crack in an image. One of the key engineering terms used for fatigue and fracture assessment is the stress intensity factor, K, which describes the stress field ahead of a crack tip. Extensive studies have been conducted over the past few decades on developing closed-form solutions for K for different crack shapes in different geometrical specimens (i.e., surface cracks, penny cracks, etc.) [126]. For complex crack shapes and specimen geometries, researchers have relied on high-fidelity FE models of the structure and cracks [127], which limits its application in routine inspection and structural assessment despite its usefulness.

To enable rapid assessment of steel structures and fully utilize the advantages of AI, a highly efficient and automated steel structure assessment scheme to estimate the stress intensity factor by integrating the defect detection analysis (discussed in Section 3.2) with a surrogate model based on fracture mechanics. The dimensional information (i.e., crack length and angle of a kink) of a crack is automatically extracted from the identified cracks using computer vision techniques, and

the nominal stresses (i.e., normal stress and shear stress) of a structure are obtained through either field monitoring, the FE model discussed in Section 4.2, or design loads. Next, a Kriging surrogate model for estimating the stress intensity factors, $K1$, and $K2$, for crack Mode I (opening) and Mode II (sliding), respectively, is established using K values obtained through FE simulations using linear elastic fracture mechanics (LEFM). In the end, the stress intensity factors can be readily estimated using the surrogate model. The proposed workflow bypasses the need for a high-fidelity FE model for calculating the stress intensity factors, thus facilitating rapid and easy applications in practice. The main advantage of this workflow is that it allows an inspector to assess the steel crack mode and estimate the propagation of a crack from inspection images using a data-driven, ML approach instead of the traditional technique of a human expert in steel fatigue and fracture. With the proposed workflow, the preliminary decisions on effective actions regarding structure conditions, repairs, and maintenance can be achieved on-site at a low cost. The proposed surrogate model has been published in [26].

Objectives

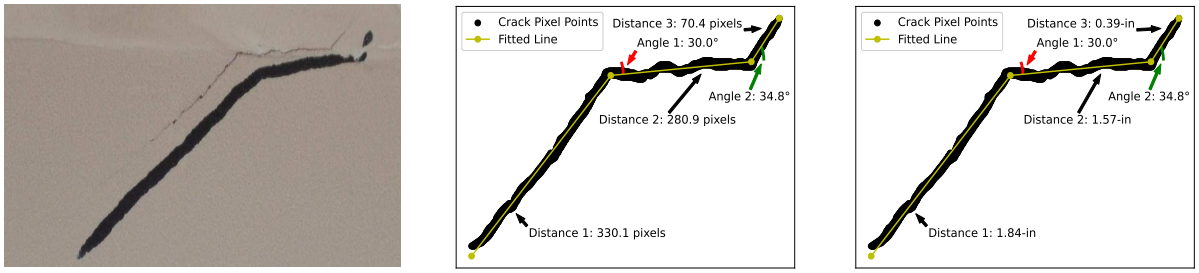
1. Develop a surrogate model to estimate the stress intensity factor of a steel crack

2. Fuse data within the DT together to assess the state of the structure

4.3.2 Task 9: Methodology

This section will first examine the inputs needed for the model and the compilation of the stress intensity factor database built with 4,000 FE model simulations. The proposed surrogate model fuses together a variety of data available within the DT. In particular, the surrogate model will take four parameters as inputs (i.e., 1) crack length, a, 2) kink length, s, 3) the angle of the kink, α, and 4) nominal stress ratio, σ/τ, based on the analysis of Melin et al. [128]) and predict the Mode I and Mode II stress intensity factors. Inputs 1, 2, and 3 (listed above) come directly from the binary segmentation images generated in Section 3.2, and input 4 comes from the FE model discussed in Section 4.2. To extract inputs 1, 2, and 3 from the binary label, an image processing technique, direct linear transformation (DLT), is performed by knowing some initial distance in

the images (i.e., from the scaled point cloud or distances obtained using the methodology for 3C dynamic measurements discussed in Section 3.6). An example output from the extraction of the input parameters from an image is shown in Figure 4.10. An efficient surrogate model is trained and then used for the prediction of the stress intensity factors to alleviate the computational challenge and the need for running separate FE analyses. To gather the data needed to train the surrogate model, a database of FE model simulations is assembled by running the FE model under different combinations of parameter values.



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

Figure 4.10: Crack mapping results from labeled crack image with the dimensions drawn on image: (a) Original crack; (b) Mapped crack in pixel-lengths; and (c) Mapped crack in engineering units

With the inputs identified to compile the stress intensity factor database and build the surrogate model, an FE model was developed to measure the Mode I and Mode II stress intensity factors based on the four input parameters. First, since the calculation of the stress intensity factors is a direct factor of the nominal normal stress, σ, and crack length, a, a normalized stress intensity factor was used,

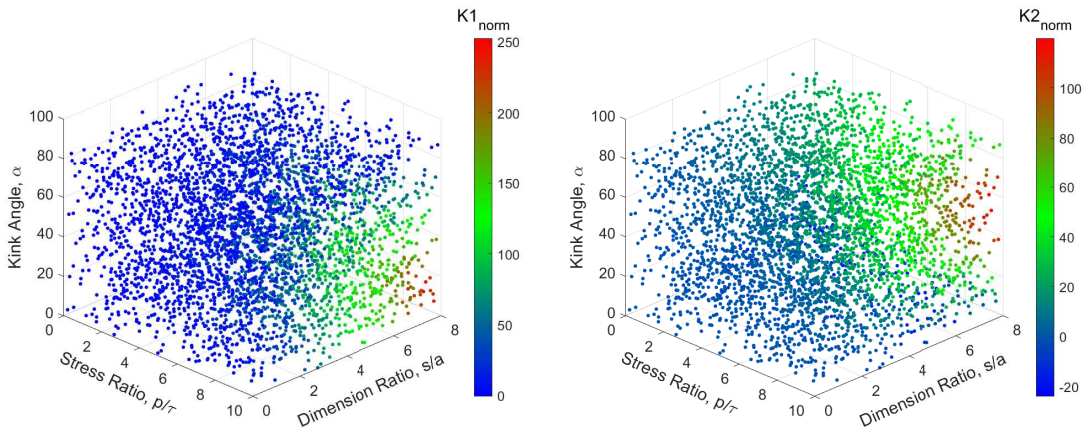
$$K_{norm} = \frac{K\left(\frac{s}{a}\right)}{\tau\sqrt{\pi a}} \tag{4.8}$$

Abaqus FE analysis (FEA), which is a general-purpose FE software [129] was implemented for the elastic analysis to directly calculate the Mode I and Mode II stress intensity factors and later converted to the normalized stress intensity factor. A 2D Steel plate model with dimensions of

100-*in*×80-*in*, modulus of elasticity of 29,000-*ksi*, and Poisson's ratio of 0.30 was used for the analysis. Two simpler crack orientations were tested to validate the Abaqus FEA model: horizontal and slanted. A normal stress, σ, of 10, 20, 30, 40, or 50-*ksi*, a crack length, a, of 2, 4, 6, 8, or 10-*in*, and a slant angle, θ, of 0, 15, or 40° were applied. The stress intensity factors obtained from the 75 Abaqus FEA models were validated against the closed-form solutions [126]. The mean percent error of the Abaqus FEA model's stress intensity factor to the closed form stress intensity factor solutions is 1.79% and 1.58% for Mode I and Mode II, respectively; therefore, the established Abaqus FEA model can be used with confidence.

With the general FE model validated, samples are generated to encompass a large range of crack dimensions and nominal loading. The four parameters (s, a, α, and σ/τ) were sampled using the Latin-Hypercube sampling technique. The ranges for s/a, and α were found using the images from the dataset and set to 0.01-8.02 and 0-90°, with increments of 0.008, and 0.09, respectively. The range for the stress ratio, σ/τ was set with typical values of stresses in steel. For rolled beams, the shear stress is typically lower than the normal stress due to the thicker webs; however, for plate girders, the shear stress can be higher. Therefore, the range of the stress ratio, σ/τ, was set to 0.2 - 10 with increments of 0.01. The normal stress, σ, remained constant at 50-*ksi* and the shear stress, τ, was found from the sampled stress ratio. Since the tests were performed with the linear elastic assumption, a change in the nominal normal stress does not affect the normalized stress intensity factor. This was verified in a preliminary validation test of the normalized stress intensity factors where the nominal normal stress was varied, but the input parameters remained constant. Moreover, since the normalized stress intensity factor, K_{norm}, is a function of the crack length, a, a preliminary study suggested there was about 10% mean error in K_{norm}, when a was changed; therefore, a was also sampled as either 2, 1, 0.5, or 0.25-*in*. 1,000 samples were collected at each crack length where the crack length remained constant and the dimension ratio, s/a, and the resulting s varied for a total of 4,000 samples. A Python script was written to run the 4,000 models of the 2D plate and measure the stress intensity factors. The outputs for Mode I and Mode II stress intensity factors are shown in a 3D scatter plot in Figure 4.11a and Figure 4.11b, respectively, and

were saved as the stress intensity factor database and used for the training of the surrogate model. A sample output of the FE model simulations is shown with Von Mises Stresses in Figure 4.12.



(a) **(b)**

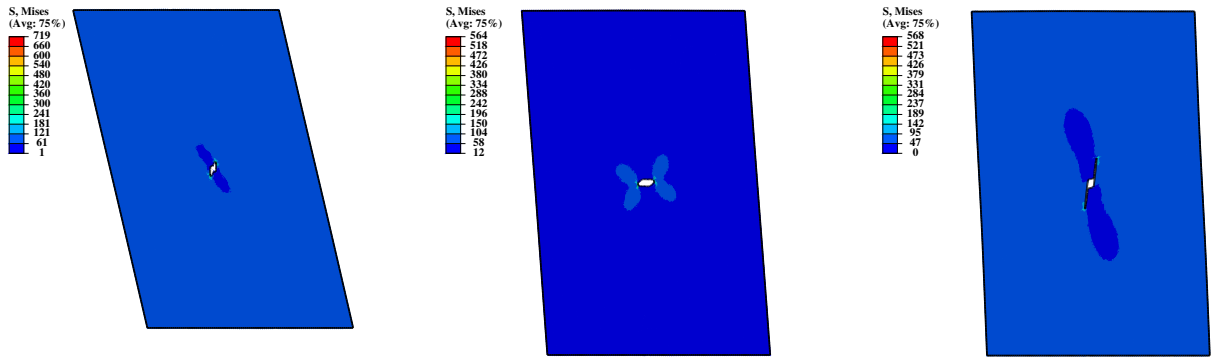
Figure 4.11: Stress intensity factors as a function of s/a, σ/τ, and α: (a) Normalized Mode I stress intensity factors; and (b) Normalized Mode II stress intensity factors

With the input parameters defined and the resulting stress intensity factor database built, a surrogate model was trained using the database. A Universal Kriging model with a second-order regression model and Gaussian correlation function was adopted. The Kriging model was chosen due to its robustness for interpolation and its ability to estimate the uncertainty associated with each prediction. The DACE toolbox [130] in MATLAB was used to build the Kriging model. A separate validation set with 500 samples was generated to check the prediction accuracy of the established Kriging model. To measure the performance of the Kriging Model, two metrics were measured on the validation set, the coefficient of determination, RD^2, and the mean percent error, ME, were used based on [131].

$$RD^2 = 1 - \frac{SSE}{SST} \tag{4.9}$$

where,

$$SSE = \sum_{p=1}^{N} \left(\boldsymbol{y}_p - \hat{\boldsymbol{y}}_p\right)^2 \quad ; \quad SST = \sum_{p=1}^{N} \left(\boldsymbol{y}_p - \sum_{p=1}^{N} \frac{\boldsymbol{y}_p}{N}\right)^2 \tag{4.10}$$



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

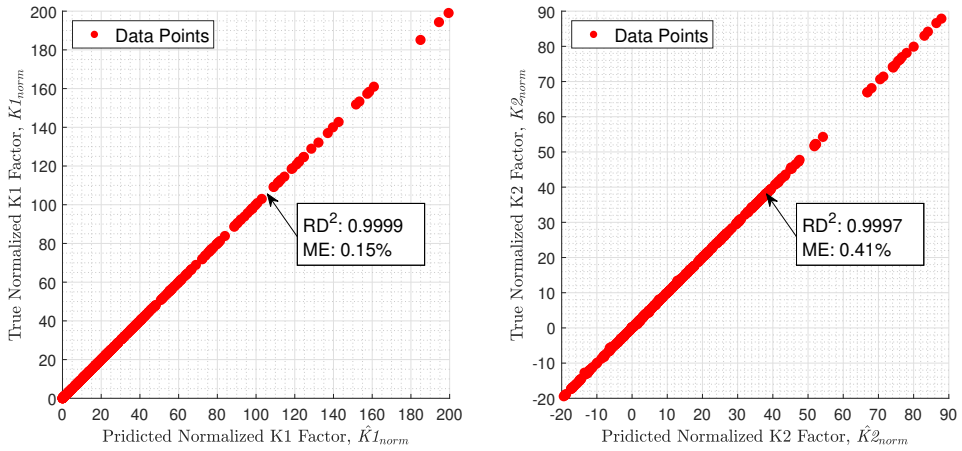
Figure 4.12: Sample FE model simulations of the stress intensity factors with Von Mises stress shown: (a) Sample 1: a = 1.0-*in*, σ/τ = 1.64, s/a = 5.47, α = 55.08°; (b) Sample 2: a = 2.0-*in*, σ/τ = 5.20, s/a = 2.48, α = 11.97°; and (c) Sample 3: a = 2.0-*in*, σ/τ = 9.69, s/a = 7.54, α = 75.06°

$$ME = \frac{\sum_{p=1}^{N} \left| \boldsymbol{y}_p - \hat{\boldsymbol{y}}_p \right|}{\sum_{p=1}^{N} \left| \boldsymbol{y}_p \right|} \tag{4.11}$$

where, $\hat{\boldsymbol{y}}_p$ is the estimation from the model and $\boldsymbol{y}_p$ is the true value from the p^{th} FE model simulation. Each metric can be taken in an average sense across the two stress intensity factors, $K1$ and $K2$. The corresponding metrics are denoted ARD^2 and AME. Note that for the coefficient of determination, ARD^2, values closer to 1 indicate better performance of the Kriging model, while for the mean percent error, AME, values closer to 0 indicate better performance.

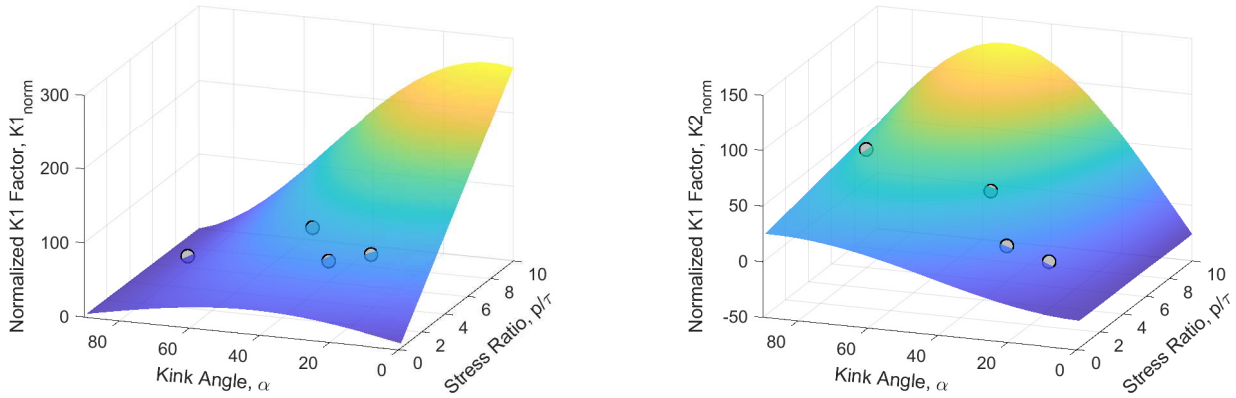
4.3.3 Task 9: Results

High performance of the Kriging Model was achieved with an ARD^2 of 0.99998 and AME of 0.23%. The true value versus the predicted stress intensity factors for Mode I and Mode II are shown Figure 4.13; the coefficient of determination, RD^2, and mean percent error, ME, for each mode are also shown. Figure 4.14 shows two 3D surface plots of the Mode I and Mode II stress intensity factors from the Kriging models; to plot the Kriging Model, s/a was set to 7.66, and the four testing data points that had the same s/a are shown as gray points on the curves.



(a) **(b)**

Figure 4.13: True versus the predicted stress intensity factors for Mode I and Mode II: (a) Normalized Mode I stress intensity factor; and (b) Normalized Mode II stress intensity factor



(a) **(b)**

Figure 4.14: Kriging model surface plot at s/a = 7.66 with testing data points shown as gray points: (a) Normalized Mode I stress intensity factor; and (b) Normalized Mode II stress intensity factor

4.4 Summary of Model Library

In this chapter, two types of models within the model library are proposed. First, an FE model was automatically built using UAS remote-sensing data and compared against a ground truth model. A surrogate model to estimate the formation mode of a crack in steel was also proposed to assess the structure's current state. Although the two proposed models do not represent all the available models that can be used within a DT, they do demonstrate how different data can be used and fused to demonstrate the current state of the structure (stress intensity factor surrogate model in Section 4.3) and allow for simulation of the structure (FE model generation in Section 4.2). As a note, data-driven models (i.e., AI models) are also incorporated into the DT; however, they are used in the data analysis module (Chapter 3) to identify the defects, components, and connections.

Chapter 5

Discussions on the Application of a Digital Twin Framework

5.1 The Use of a Digital Twin

This section discusses how each module within the DT can be used and implemented for performance assessment and decision-making. Information from the data analytics module (Chapter 3) and the model library (Chapter 4) can be incorporated together to provide decision-making support. One of the benefits of using a DT over traditional bridge management practices is its ability to analyze and fuse data together to allow bridge managers to make more informed decisions. To illustrate the benefits of DT via vivid discussions, two virtual illustrative examples that incorporate all the aspects of a DT are used to show the potential use of the DT. Specifically, this chapter is divided into two sections to discuss the application of a DT using a virtual steel and concrete bridge. This section will step through the processes of collecting, analyzing, and interpreting the collected data of the bridges using the two virtual bridge examples. Lastly, a summary is presented.

Objectives

1. Provide two full examples to show how to fuse the different data types within the DT for steel and concrete bridges

2. Explore the use of the DT to facilitate decision-making

5.2 Workflow for a Steel Bridge

5.2.1 Performance Assessment for a Steel Bridge

First stage assessment: Assume that the virtual steel structure has a concerning crack that needs to be repaired. The workflow of DT is illustrated assuming the data is obtained from a recurring

inspection. To begin the workflow of the DT analysis, a UAS-enabled inspection is performed on a steel structure. 3,000 16-*MP* images are collected from various viewpoints around a steel bridge and 20-*minutes* of video is collected as traffic drives across the structure with the dual stereo vision camera rig. Then the preliminary data analysis starts by generating visualization models using the structure-from-motion algorithm. The resulting 3D point cloud and photorealistic models provide the basis for the user to visualize the DT. During this step, the camera's poses and depth maps are calculated. Then, more in-depth data analysis can be conducted. Specifically, the defects are identified through machine learning (Section 3.2). Since this is a recurring inspection, the growth of the defects is measured (Section 3.3). The dimensions of the defects and the growth of the defects are measured using stereo vision or by relating the scale from the structure-from-motion camera pair (discussed in Section 4.3). Using the proposed AI network, the components and connections within the bridge are identified by fusing the original color images from cameras and depth images generated from the 3D point cloud. These segmented images are then projected onto the 3D points resulting in a component-wise 3D point cloud. With the identified structural components and camera poses, the damage information and other data (i.e., design loads, ADT, NDE results, etc.) are mapped onto the identified components and stored within the DT to facilitate data visualization. For example, selecting a component or damaged location on the visualization model displays all the information about the component or damage. Additionally, the video collected from the four optical cameras is processed, and the 3C dynamic displacements are measured. Next, by fusing the 3D point cloud, connection identification, and component segmentation information together, the FE model is automatically generated (Section 4.2). Using the measured dynamic displacement and identified damage information, the automatically built FE model is updated and validated.

Second stage assessment: With the aforementioned first stage processing completed, the bridge manager now has quantitative damage information which is advantageous over existing bridge inspection techniques. By utilizing the model library, however, more information about the structure's health can become available to inform the decision-making, which is described below. During the defect detection analysis of this virtual bridge, there was a crack detected on a steel member

from the first stage analysis. Fusing the data of the defect's dimension and the stress of the member found with the automatically built FE model, the information is passed through the surrogate model of the stress intensity factor to quickly find the K values and dominant Modes of crack formation without the need for a high-fidelity FE model. The Mode I stress intensity factor, K_I, is compared to the critical stress intensity factor for Mode I (K_{IC}). At this point, the bridge manager knows how critical the steel crack is in the steel member. The K factor, mapped damage information, and as-built FE model empower the bridge manager to better understand how the structure is performing at its current state. As discussed in the following section, the bridge manager is able to use this information to make more informed decisions. The time and user-input requirements for the steel bridge assessment are shown in Table 5.1.

Table 5.1: Time and user-input requirements for steel structure

	Time	User-Input
Data Collection	1-*day*	Significant
Data Analytics		
Defect Detection	3.2-*hours*	Minimal
Defect Growth	*minutes*	Minimal
Measurement of Defects	*minutes*	Median
Component Identification	3.5-*hours*	Minimal
Connection Identification	1-*hour*	Minimal
3C Dynamic Displacement Measurements	1-*hour*	Minimal
Model Library		
Visualization Models Creation	5-*hours*	Minimal
FE Model Generation	0.5-*hours*	Median
K Estimation (Surrogate Model)	*minutes*	Minimal
Decision Making		
Mapping Defect to FE Model	*minutes*	Minimal
Validating FE Model	*minutes*	Minimal
Simulating Repair Options	*hours*	Median

5.2.2 Decision Making for a Steel Bridge

With a more holistic representation of the structure's current condition, the DT is used to help the bridge manager make decisions on whether a repair is needed of the steel crack, how soon repair is needed, and what repair option is the most cost-effective over time. First, comparing the measured Mode I stress intensity factor to the critical Mode I stress intensity factor, K_{IC}, the crack is nearing critical. Using the Paris' Equation (Equation 5.1), the number of cycles remaining until failure is estimated,

$$\frac{\mathrm{d}a}{\mathrm{d}N} = C\left(\Delta K\right)^m \tag{5.1}$$

where $\frac{\mathrm{d}a}{\mathrm{d}N}$ is the change of crack length over cycles, C and m are deterministic material property, ΔK is the change of stress intensity factor from minimum and maximum stresses applied [132]. With Equation 5.1, the number of cycles until failure is estimated. Also, the ΔK variable represents the stress range of the member; therefore, a load rating can be placed on the structure if needed to alleviate the need for a structure closure. In this structure, it was determined that a load rating would bypass the need for closure; however, after 3-*months*, the structure needs to be closed. The as-built FE model in DT is then used to simulate repair decisions and predict which type of repair would be the most effective for the structure. Three options are automatically identified to repair the defect based on previous and known repair options: drilling a hole at the tip of the crack, welding the crack together, and welding a plate over the crack. Each option is tested within the FE model to predict how the structure will perform in the future and the additional effects the repair will have on the structure. In addition, a cost-benefit analysis is performed to identify the most efficient repair option over time within the DT. The best repair option can be identified, and the bridge manager reviews the options and decides on the best practice. Because such a significant crack was identified, the next inspection is scheduled to take place soon after the repair. The data and models are stored within the DT. If the structure performs how the DT predicted after the repair, then the DT is used more confidently. Relying on the simulated performance of the structure from the DT, the inspection interval can be determined.

5.3 Workflow for a Concrete Bridge

5.3.1 Performance Assessment for a Concrete Bridge

Assume that the virtual concrete bridge is in satisfactory condition and is undergoing a routine inspection. The performance assessment of the concrete bridge is similar to the "First stage assessment" of the steel bridge as detailed in the previous section. In the entire data analysis procedure, only minimal user inputs are required for model initiation and output verification, and it is mostly fully automated. The time and user input requirements for this concrete bridge are presented in Table 5.2. The data processing burden for the inspector and crew is reduced. Compared to traditional inspection techniques, the proposed framework requires more computing time; however, the algorithms can run autonomously or be left overnight, thus drastically reducing report generation time for the inspector.

Table 5.2: Time and user-input requirements for concrete structure

	Time	User-Input
Data Collection	1-*day*	Significant
Data Analytics		
Defect Detection	3.2-*hours*	Minimal
Defect Growth	*minutes*	Minimal
Measurement of Defects	*minutes*	Median
Component Identification	3.5-*hours*	Minimal
3C Dynamic Displacement Measurements	1-*hour*	Minimal
Model Library		
Visualization Model Creation	5-*hours*	Minimal
FE Model Generation	0.5-*hours*	Median
Decision Making		
Reviewing Data Analytics Results	*minutes*	Minimal

5.3.2 Decision Making for a Concrete Bridge

After data analysis is performed on the concrete structure, the structural elements, defects, and defect growth are identified and mapped onto the structural components of the DT. An in-

spector performs a quality control assessment afterward to ensure the proper performance of the algorithms. Additionally, system identification is performed on the bridge using the 3C dynamic displacement measurements. In the case of the second inspection instance of the bridge using the proposed framework, the collected data can be compared to the previous data more cohesively and accurately. The previous data, including the images, quantified defect information, and measured system identification parameters is also accessible to the inspector to make more informed decisions. The steps to maintain the structure are determined with the additional information available to a bridge inspector or manager. There was minimal defect growth from inspection to inspection; therefore, a typical inspection cycle is decided on the structure with high confidence. The bridge was determined not to have any severe issues. If determined to be needed by an inspector, the FE model module is readily available in the DT to facilitate decision-making in the daily operation of the bridge, such as load ratings.

5.4 Summary of the Application of the Digital Twin Framework

Two virtual bridges were used to discuss the potential usage of the DT. First, for the steel bridge, a crack can be identified in the steel structure through the defect detection module of the DT. Using the DT, an "as-built" FE model was generated and validated. Three repair options for the defect were simulated on the FE model of the DT. Ultimately, the bridge manager quantitatively reviewed how critical the defect was and was empowered to take immediate action on the structure. With the results of the FE simulations, the cost/benefit analysis of the repairs, and a prediction estimation on when the crack reached a critical level, the bridge manager was able to decide on the best course of action on the defect and plan accordingly. In the next inspection cycle of the bridge, the performance predictions of the FE simulations are validated to ensure the structure is performing as expected and the DT can be updated to ensure it evolves with its physical twin. In the example of the concrete bridge, it was determined to not have significant issues. However, quantitative information was collected and stored so that more informed decisions can be made during the

present and future inspections. Additionally, an FE model can readily be built automatically and used for load rating.

Chapter 6

Conclusions

6.1 Summary

A comprehensive DT framework was proposed to incorporate various metrics to assess the current state of a bridge, model future performance, and provide decision-making support. The proposed DT framework mainly includes a data analytics module and a model library. First, the data analytics module was proposed to process remote sensing data from a UAS. Within this module, the cracks on structural members were able to be segmented from the images, and the growth of the cracks was tracked over time. Three techniques were presented to measure the dynamic displacements of objects with cameras attached to a UAS. The most advanced technique can accurately capture the 3C dynamic displacements of full-scale structures. Lastly, two AI networks were developed to semantically segment the components of the bridge from images and localize the connections for both steel and concrete bridges. Using the data obtained from the analytics module, two models were proposed within the model library: (1) an automatically built FE model and (2) a surrogate model to predict the stress intensity factor. For (1), by fusing images, depth maps, a 3D point cloud, semantic component labels from images, and connection identification from images, an FE model was automatically generated with little user input. For (2), the semantic crack identification labels and measurements as well as data from FE analysis were fused to develop a surrogate model to efficiently estimate the Mode I and Mode II stress intensity factors. Lastly, a discussion was presented on implementing and utilizing DTs for steel and concrete bridges.

6.2 Contributions

The research performed and proposed in this paper will provide the following contributions to the research community:

1. **Identifying cracks in steel structures and measuring defect growth:** Although there is literature on identifying cracks in concrete (partly due to the availability of datasets and the wider and longer dimensions of the concrete defect), much less research has been identified to semantically segment the location of cracks in steel structures. This research proposed an AI network to semantically segment cracks in steel structures. The geometric features, such as length, kink angles, etc. have also been automatically extracted from images. Moreover, with the proposed automated and quantitative techniques to identify defects, the ability to track the growth over time becomes possible which few researchers have explored (mainly due to available data over time).

2. **3C dynamic displacement measurements with a UAS:** Much of the current research of measuring displacement with a UAS mainly focuses on either the 1C measurements (i.e., depth) or the 2C measurements (i.e., planar to the camera). Although there are a number of sensors available (mainly for human-computer interactions with a computer), there are many limitations to using these sensors for dynamic structural displacements. A portable UAS platform that is able to measure the 3C displacements of many bridges with one platform in a cost-effective manner is currently unavailable. This study fills the research gap.

3. **Constructing an FE model automatically with information collected with a UAS:** Currently, there is no automated framework that requires little user input to build a FE model from point cloud data for bridges. By fusing two AI models (i.e., component identification and connection identification) with computer vision techniques, the automated construction of an FE model was realized in this study.

4. **Stress intensity factor estimation of steel cracks in a computationally efficient manner using images of steel cracks:** While estimating the stress intensity factor and crack mode type would enable more informed decision-making, it typically requires a high fidelity FE model for complicated crack geometries which is time- and resource-consuming in practice. By identifying the steel cracks from images with AI and building a database of steel crack

geometries and nominal stresses through FE simulations, a surrogate model is established to estimate the stress intensity factor directly from the images in a more efficient manner. Keep in mind that the stress intensity factor indicates the crack formation modes, and thus can inform the appropriate type of repair. The proposed surrogate model allows for the availability of critical information on the stress intensity factor at a low cost to better support decision-making.

5. **Streamlined modeling:** There are a variety of models that bridge managers can use to understand a structure; however, gathering the required data and building the model may be too cumbersome for many DOTs. Within a DT, data for these models are readily available (i.e., either the raw data directly from remote sensing or the analyzed remote sensing data found in the data analysis section). Models and data processing tools (such as the three models proposed in the model library) have been incorporated into the DT and allow for a more streamlined processing of the data for data-driven decision-making which is currently a barrier for DOTs to utilize the models.

6. **Automatically building a Digital Twin with UAS-enabled optical sensors:** Currently, there is no centralized portal for structural analysis, modeling, assessment, and simulation. This study has filled this gap by proposing a first-of-its-kind DT. The proposed workflow to establish DT requires, at a minimum, UAS-enable images (or videos) of bridges, which can be very cost-effective for stakeholders, such as state DOTs.

7. **Facilitate decision-making and structural assessment:** While on average, current inspection techniques, reporting, and decision-making for structures have been established, there is still room for a more efficient and streamlined decision-making process. Currently, decisions are made in a more ambiguous environment. Providing a DT for assessments, predictions, and simulations allows for a more data-driven approach to structural asset management. Bridge managers can perform "what-if" simulations on a specific structure in a more efficient manner which is currently time-consuming and costly.

6.3 Future Directions

1. **Increase pixel accuracy of 3C displacement measurement:** Although a high accuracy of 3C displacement measurements was possible with the proposed UAS-enabled dual stereo vision system, to increase the practicality in the field, an even longer working distance between the cameras and the structure might be needed for very large-scale structures. Theoretically, a longer working distance is possible with the proposed technique if better cameras are used. By using higher-performing cameras, which in turn have a similar GSD but at a longer distance, a similar accuracy as in this study can be achieved. This proposal needs to be explored and validated. Moreover, there are other post-processing techniques to help improve the performance of cameras such as super-resolution and super-sensitivity which may be able to artificially increase the resolution of cameras. These post-processing techniques also need to be explored.

2. **Incorporate additional models within model library:** An advantage of the proposed DT is the ability to collect and run different models depending on the situation and structural need. Additional models can be incorporated into a digital twin. Building and/or training these models would make the digital twin framework even more versatile and robust. Additional model types can still be explored and studied. Some proposed model types are listed below.

   (a) **Deterioration models:** A model to predict the growth of a defect over time (i.e., crack, section loss, corrosion, shrinkage, etc.). Inspection data (from the "Data Analytics" section) will be incorporated to update these deterioration models and improve prediction accuracy. The engineering judgment by experts could also be incorporated into these model predictions to determine the likelihood and consequences of different damage modes.

   (b) **Inspection interval model:** A decision framework could be developed to select an inspection interval by integrating information from deterioration models, inspection data, and expert judgment using a Bayesian framework.

(c) **Cost/Benefit Analysis:** Incorporating historical data, a cost-benefit model could be developed within the DT to help determine repair options for a structure automatically or determine if replacement is the better long-term solution.

3. **Fusing different data types:** In this study, only images and videos collected with a UAS and load data were used to build and develop the DT. However, a variety of data can be collected, stored, and processed within a DT. These additional data types (i.e., average daily traffic, NDE techniques, weather, etc.) can be incorporated within a DT to provide an even more holistic overview of a structure.

4. **Evolve the DT over time:** This study illustrated two digital twins for concrete and steel bridges and how an inspector or bridge manager could use the data to make more informed decisions in the discussion chapter. However, real-world implementation and evolution of DT have not been tested with real data. A longitudinal study on how to continuously incorporate and implement new data into the DT can be explored more.

5. **Increase labeled data sets of all AI networks:** The proposed AI networks have shown strong performance metrics. However, with additional labeled data, the proposed AI network can be updated to perform more robustly on a variety of structures and structural types. Once more labeled data is available, the models can be easily updated. Eventually, human input might not be needed anymore.

Bibliography

[1] Eric Bianchi, Amos Lynn Abbott, Pratap Tokekar, and Matthew Hebdon. COCO-Bridge: Structural Detail Data Set for Bridge Inspections. *Journal of Computing in Civil Engineering*, 35(3):04021003, 2021.

[2] Khatereh Vaghefi, Renee C. Oats, Devin K. Harris, Theresa (Tess) M. Ahlborn, Colin N. Brooks, K. Arthur Endsley, Christopher Roussi, Robert Shuchman, Joseph W. Burns, and Richard Dobson. Evaluation of Commercially Available Remote Sensors for Highway Bridge Condition Assessment. *Journal of Bridge Engineering*, 17(6):886–895, 2011.

[3] American Society of Civil Engineering. The Vision for Civil Engineering in 2025. In *The Vision for Civil Engineering in 2025*, pages 1–103, Reston, VA, 6 2007. American Society of Civil Engineers.

[4] Federal Highway Administration. National Bridge Inventory.

[5] AASHTO. *The Manual for Bridge Evaluation, Third Edition, 2017*. American Association of State Highway and Transportation Officials, Washington, D.C., 3rd edition, 2018.

[6] Tarek Omar and Moncef L. Nehdi. Remote sensing of concrete bridge decks using unmanned aerial vehicle infrared thermography. *Automation in Construction*, 83(4):360–371, 2017.

[7] Mark Moore, Brent Phares, Benjamin Graybeal, Dennis Rolander, and Glenn Washer. Reliability of Visual Inspection for Highway Bridges, Volum I: Final Report. Technical Report FHWA-RD-01-020, FHWA, McLean, VA, 2001.

[8] Eric VanDerHorn and Sankaran Mahadevan. Digital Twin: Generalization, characterization and implementation. *Decision Support Systems*, 145(6):113524, 2021.

[9] Feng Jiang, Ling Ma, Tim Broyd, and Ke Chen. Digital twin and its implementations in the civil engineering sector. *Automation in Construction*, 130(7):103838, 2021.

[10] B. J. Perry, Y. Guo, R. Atadero, and J. W. van de Lindt. Streamlined bridge inspection system utilizing unmanned aerial vehicles (UAVs) and machine learning. *Measurement*, 164:108048, 2020.

[11] Vivi Qiuchen Lu, Ajith Kumar Parlikad, Philip Woodall, Gishan Don Ranasinghe, and James Heaton. Developing a Dynamic Digital Twin at a Building Level: using Cambridge Campus as Case Study. In *International Conference on Smart Infrastructure and Construction 2019 (ICSIC)*, pages 67–75, Cambridge, England, 1 2019. ICE Publishing.

[12] Brendan McGuire, Rebecca Atadero, Caroline Clevenger, and Mehmet Ozbek. Bridge Information Modeling for Inspection and Evaluation. *Journal of Bridge Engineering*, 21(4):04015076, 2016.

[13] Saleh Abu Dabous, Salam Yaghi, Sabah Alkass, and Osama Moselhi. Concrete bridge deck condition assessment using IR Thermography and Ground Penetrating Radar technologies. *Automation in Construction*, 81(April):340–354, 2017.

[14] Abe Danaher. Digital twins to improve bridge assessment in rural South Carolina, 2021.

[15] Dimitrios Giagopoulos, Alexandros Arailopoulos, Vasilis Dertimanis, Costas Papadimitriou, Eleni Chatzi, and Konstantinos Grompanopoulos. Structural health monitoring and fatigue damage estimation using vibration measurements and finite element model updating. *Structural Health Monitoring*, 18(4):1189–1206, 2019.

[16] Sin-Chi Kuok and Ka-Veng Yuen. Structural health monitoring of Canton Tower using Bayesian framework. *Smart Structures and Systems*, 10(4):375–391, 2012.

[17] H. F. Lam, L. S. Katafygiotis, and N. C. Mickleborough. Application of a Statistical Model Updating Approach on Phase I of the IASC-ASCE Structural Health Monitoring Benchmark Study. *Journal of Engineering Mechanics*, 130(1):34–48, 2004.

[18] Stefan Boschert and Roland Rosen. *Digital Twin—The Simulation Aspect*. Springer International Publishing, Cham, 2016.

[19] Pamela A. Kobryn. The Digital Twin Concept. In *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2019 Symposium*, pages 17–22, Washington, DC, 2020. The National Academies Press.

[20] Eric J. Tuegel, Anthony R. Ingraffea, Thomas G. Eason, and S. Michael Spottswood. Reengineering Aircraft Structural Life Prediction Using a Digital Twin. *International Journal of Aerospace Engineering*, 2011(2011):1–14, 2011.

[21] Alireza Adibfar and Aaron M. Costin. Creation of a Mock-up Bridge Digital Twin by FusingIntelligent Transportation Systems (ITS) Data intoBridge Information Model (BrIM). *Journal of Construction Engineering and Management*, 148(9):04022094, 2022.

[22] Manuel Chiachío, María Megía, Juan Chiachío, Juan Fernandez, and María L. Jalón. Structural digital twin framework: Formulation and technology integration. *Automation in Construction*, 140(5):104333, 2022.

[23] Cong Ye, Liam Butler, Bartek Calka, Marat Iangurazov, Qiuchen Lu, Alastair Gregory, Mark Girolami, and Campbell Middleton. A Digital Twin of Bridges for Structural Health Monitoring. In *Structural Health Monitoring 2019*, pages 1619–1626, Lancaster, PA, 11 2019. DEStech Publications, Inc.

[24] Hussam N. Mahmoud, Guillermo A. Riveros, Mehrdad Memari, Anuj Valsangkar, and Bashir Ahmadi. Underwater Large-Scale Experimental Fatigue Assessment of CFRP-Retrofitted Steel Panels. *Journal of Structural Engineering*, 144(10):04018183, 2018.

[25] Arkansas Trucking Association. I-40 Bridge Closure Estimated to Cost Trucking $2.4M Per Day, 2021.

[26] Brandon J Perry, Yanlin Guo, and Hussam N Mahmoud. Automated site-specific assessment of steel structures through integrating machine learning and fracture mechanics. *Automation in Construction*, 133(1):104022, 2022.

[27] Arsalan Mahmoodzadeh, Mokhtar Mohammadi, Hawkar Hashim Ibrahim, Krikar M. Gharrib Noori, Sazan Nariman Abdulhamid, and Hunar Farid Hama Ali. Forecasting sidewall displacement of underground caverns using machine learning techniques. *Automation in Construction*, 123(12):103530, 2021.

[28] S. Mustapha, A. Kassir, K. Hassoun, Z. Dawy, and H. Abi-Rached. Estimation of crowd flow and load on pedestrian bridges using machine learning with sensor fusion. *Automation in Construction*, 112(8):103092, 2020.

[29] Bo Xiao, Qiang Lin, and Yuan Chen. A vision-based method for automatic tracking of construction machines at nighttime based on deep learning illumination enhancement. *Automation in Construction*, 127(3):103721, 2021.

[30] Wilson Ricardo Leal da Silva and Diogo Schwerz de Lucena. Concrete Cracks Detection Based on Deep Learning Image Classification. In *Proceedings of Eighteenth International Conference of Experimental Mechanics*, volume 2, page 489, Brussels, Belgium, 6 2018. MDPI.

[31] Cao Vu Dung and Le Duc Anh. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in Construction*, 99(November 2018):52–58, 2019.

[32] Hoang Nhat-Duc, Quoc-Lam Nguyen, and Van-Duc Tran. Automatic recognition of asphalt pavement cracks using metaheuristic optimized edge detection algorithms and convolution neural network. *Automation in Construction*, 94(6):203–213, 2018.

[33] Lei Zhang, Fan Yang, Yimin Daniel Zhang, and Ying Julie Zhu. Road Crack Detection Using Deep Convolutional Neural Network. In *IEEE International Conference on Image Processing*, pages 3708–3712, Pheoniz, AZ, 9 2016. IEEE.

[34] Wooram Choi and Young-Jin Cha. SDDNet: Real-Time Crack Segmentation. *IEEE Transactions on Industrial Electronics*, 67(9):8016–8025, 2020.

[35] Yahui Liu, Jian Yao, Xiaohu Lu, Renping Xie, and Li Li. DeepCrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, 338(4):139–153, 2019.

[36] Xincong Yang, Heng Li, Yantao Yu, Xiaochun Luo, Ting Huang, and Xu Yang. Automatic Pixel-Level Crack Detection and Measurement Using Fully Convolutional Network. *Computer-Aided Civil and Infrastructure Engineering*, 33(12):1090–1109, 2018.

[37] Allen Zhang, Kelvin C. P. Wang, Baoxian Li, Enhui Yang, Xianxing Dai, Yi Peng, Yue Fei, Yang Liu, Joshua Q. Li, and Cheng Chen. Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep-Learning Network. *Computer-Aided Civil and Infrastructure Engineering*, 32(10):805–819, 2017.

[38] Ahmed Mahgoub Ahmed Talab, Zhangcan Huang, Fan Xi, and Liu HaiMing. Detection crack in image using Otsu method and multiple filtering in image processing techniques. *Optik*, 127(3):1030–1033, 2016.

[39] S. Sankarasrinivasan, E. Balasubramanian, K. Karthik, U. Chandrasekar, and Rishi Gupta. Health Monitoring of Civil Structures with Integrated UAV and Image Processing System. *Procedia Computer Science*, 54:508–515, 2015.

[40] Hung Manh La, Tran Hiep Dinh, Nhan Huu Pham, Quang Phuc Ha, and Anh Quyen Pham. Automated robotic monitoring and inspection of steel structures and bridges. *Robotica*, 37(5):947–967, 2019.

[41] Yang Xu, Yuequan Bao, Jiahui Chen, Wangmeng Zuo, and Hui Li. Surface fatigue crack identification in steel box girder of bridges by a deep fusion convolutional neural network based on consumer-grade camera images. *Structural Health Monitoring*, 18(3):653–674, 2019.

[42] Chuanzhi Dong, Liangding Li, Jin Yan, Zhiming Zhang, Hong Pan, and Fikret Necati Catbas. Pixel-Level Fatigue Crack Segmentation in Large-Scale Images of Steel Structures Using an Encoder–Decoder Network. *Sensors*, 21(12):4135, 2021.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of The 18th International Conference of Medical Image Computing and Computer-Assisted Intervention*, volume 3, pages 234–241, Munich, Germany, 10 2015. Springer.

[44] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–14, San Diego, CA, USA, 9 2014.

[45] Xuan Liu, Mingmin Chi, Yunfeng Zhang, and Yiqing Qin. Classifying High Resolution Remote Sensing Images by Fine-Tuned VGG Deep Networks. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 7137–7140, Valencia, Spain, 7 2018. IEEE.

[46] Muhammad Mateen, Junhao Wen, Nasrullah, Sun Song, and Zhouping Huang. Fundus Image Classification Using VGG-19 Architecture with PCA and SVD. *Symmetry*, 11(1):1, 2018.

[47] Di He, Ke Xu, and Peng Zhou. Defect detection of hot rolled steels with a new object detection framework called classification priority network. *Computers & Industrial Engineering*, 128(August 2018):290–297, 2019.

[48] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya

Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

[49] Yuequan Bao, Jian Li, Tomonori Nagayama, Yang Xu, Billie F Spencer, and Hui Li. The 1st International Project Competition for Structural Health Monitoring (IPC-SHM, 2020): A summary and benchmark problem. *Structural Health Monitoring*, 20(4):2229–2239, 2021.

[50] Zhiyi Tang, Fangqiao Hu, Yuhu Quan, and Kun Fang. APESS2018 Steel Girder Crack ID Data Set, 2018.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[52] Daniel T. Gillins, Christopher Parrish, Matthew N. Gillins, and Chase Simpson. Eyes in the Sky: Bridge Inspections With Unmanned Aerial Vehicles. Technical report, Oregon Department of Transportation, Salem, OR, 2018.

[53] Cao Vu Dung, Hidehiko Sekiya, Suichi Hirano, Takayuki Okatani, and Chitoshi Miki. A vision-based method for crack detection in gusset plate welded joints of steel bridges using deep convolutional neural networks. *Automation in Construction*, 102(March):217–229, 2019.

[54] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157, Kerkyra, Greece, 9 1999. IEEE.

[55] ASTM International. ASTM C496 Standard Test Method for Splitting Tensile Strength of Cylindrical Concrete Specimens. Technical report, ASTM, West Conshohocken, PA, 2017.

[56] Y. Xu, J. Brownjohn, and D. Kong. A non-contact vision-based system for multipoint displacement monitoring in a cable-stayed footbridge. *Structural Control and Health Monitoring*, 25(5):2155, 218.

[57] G. Jeon, S. Kim, S. Ahn, H. Kim, and H. Yoon. Vision-based automatic cable displacement measurement using Cable-ROI Net and Uni-KLT. *Structural Control and Health Monitoring*, 29(8):2977, 2022.

[58] D. Feng, T. Scarangello, M. Q. Feng, and Q. Ye. Cable tension force estimate using novel noncontact vision-based sensor. *Measurment*, 99(3):44–52, 2017.

[59] Z. Ma, J. Choi, and H. Sohn. Noncontact cable tension force estimation using an integrated vision and inertial measurement system. *Measurement*, 199(8):111532, 2022.

[60] B. J. Perry and Y. Guo. A portable three-component displacement measurement technique using an unmanned aerial vehicle (UAV) and computer vision: A proof of concept. *Measurement*, 176:109222, 2021.

[61] Y. Tian, C. Zhang, S. Jiang, J. Zhang, and W. Duan. Noncontact cable force estimation with unmanned aerial vehicle and computer vision. *Computer-Aided Civil and Infrastructure Engineering*, 36(1):73–88, 2020.

[62] M. K. Alkharisi and P. R. Heyliger. Modal dynamics of twisted cables. *Journal of Sound and Vibration*, 514(8):116431, 2022.

[63] B. J. Perry, P. R. Heyliger, Y. Guo, and M. K. Alkharisi. Unmanned aerial system (UAS)-based portable sensing for blast-loaded cables. *Journal of Structural Engineering*, Under Review, 2023.

[64] C. Z. Dong and F. N. Catbas. A review of computer vision–based structural health monitoring at local and global levels. *Structural Health Monitoring*, 20(2):692–743, 2020.

[65] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proceedings of the Alvey Vision Conference*, pages 23.1–23.6, Manchester, UK, 1988. Alvety Vision Club.

[66] Jean-Yves Bouguet et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel corporation*, 5(1–10):4, 2001.

[67] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(8):185–203, 1981.

[68] H. M Irvine and T. K. Caughey. The linear theory of free vibrations of a suspended cable. In *Proceedings of the Royal Society of London*, 341, pages 299–315, London, England, 1974. The Royal Society Publishing.

[69] M. K. Alkharisi and P. R. Heyliger. Frequency order and modal shifts in sagged cables. *Finite Elements in Analysis and Design*, 215(1):103889, 2023.

[70] Z. Li and C. C. Chang. Tracking of Structural Dynamic Characteristics Using Recursive Stochastic Subspace Identification and Instrumental Variable Technique. *Journal of Engineering Mechanics*, 138(6):591–600, 2012.

[71] C. Rainieri and G. Fabbrocino. Automated output-only dynamic identification of civil engineering structures. *Mechanical Systems and Signal Processing*, 24(3):678–695, 2010.

[72] Rune Brincker, L. Zhang, and P. Andersen. Modal Identification from Ambient Responses using Frequency Domain Decomposition. In *IMAC 18: Proceedings of the International Modal Analysis Conference*, pages 625–630, San Antonio, TX, 2000. Aalborg Universitet.

[73] Y.L. Guo, A. Kareem, Y.Q. Ni, and W.Y. Liao. Performance evaluation of Canton Tower under winds based on full-scale data. *Journal of Wind Engineering and Industrial Aerodynamics*, 104-106(5):116–128, 2012.

[74] Yanlin Guo, Dae Kun Kwon, and Ahsan Kareem. Near-Real-Time Hybrid System Identification Framework for Civil Structures with Application to Burj Khalifa. *Journal of Structural Engineering*, 142(2):04015132, 2016.

[75] Yanlin Guo and Ahsan Kareem. System identification through nonstationary data using Time–Frequency Blind Source Separation. *Journal of Sound and Vibration*, 371(6):110–131, 2016.

[76] Yanlin Guo and Ahsan Kareem. System Identification through Nonstationary Response: Wavelet and Transformed Singular Value Decomposition—Based Approach. *Journal of Engineering Mechanics*, 141(7):04015013, 2015.

[77] Xirui Ma, Yizhou Lin, Zhenhua Nie, and Hongwei Ma. Structural damage identification based on unsupervised feature-extraction via Variational Auto-encoder. *Measurement: Journal of the International Measurement Confederation*, 160(8):107811, 2020.

[78] Y. Q. Ni, X. G. Hua, K. Y. Wong, and J. M. Ko. Assessment of Bridge Expansion Joints Using Long-Term Displacement and Temperature Measurement. *Journal of Performance of Constructed Facilities*, 21(2):143–151, 2007.

[79] X. W. Ye, T. Jin, and C. B. Yun. A review on deep learning - based structural health monitoring of civil infrastructures. *Smart Structures and Systems*, 24(5):567–586, 2019.

[80] X. W. Ye, C. Z. Dong, and T. Liu. A Review of Machine Vision-Based Structural Health Monitoring: Methodologies and Applications. *Journal of Sensors*, 2016:7103039, 2016.

[81] Taha Teimoori and Mussa Mahmoudi. Damage detection in connections of steel moment resisting frames using proper orthogonal decomposition and wavelet transform. *Measurement*, 166:108188, 12 2020.

[82] Vedhus Hoskere, Jong Woong Park, Hyungchul Yoon, and Billie F. Spencer. Vision-Based Modal Survey of Civil Infrastructure Using Unmanned Aerial Vehicles. *Journal of Structural Engineering (United States)*, 145(7):1–14, 2019.

[83] Michail Kalaitzakis, Sreehari Rajan Kattil, Nikolaos Vitzilaios, Dimitris Rizos, and Michael Sutton. Dynamic Structural Health Monitoring using a DIC-enabled drone. In *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 321–327, Atlanta, GA, 6 2019. IEEE.

[84] Piyush Garg, Fernando Moreu, Ali Ozdagli, Mahmoud Reda Taha, and David Mascareñas. Noncontact Dynamic Displacement Measurement of Structures Using a Moving Laser Doppler Vibrometer. *Journal of Bridge Engineering*, 24(9):1–13, 2019.

[85] Piyush Garg, Roya Nasimi, Ali Ozdagli, Su Zhang, David Dennis Lee Mascarenas, Mahmoud Reda Taha, and Fernando Moreu. Measuring Transverse Displacements Using Unmanned Aerial Systems Laser Doppler Vibrometer (UAS-LDV): Development and Field Validation. *Sensors*, 20(21):6051, 2020.

[86] Sean Catt, Benjamin Fick, Matthew Hoskins, Joseph Praski, and Javad Baquersad. Development of a Semi-Autonomous Drone for Structural Health Monitoring of Structures using Digital Image Correlation (DIC). In *Structural Health Monitoring, Photogrammetry & DIC Volume 6. Proceedings of the 36th IMAC, A Conference and Exposition on Structural Dynamics 2018*, volume 6, Orlando, FL, USA, 2019. Springer.

[87] Hyungchul Yoon, Jaeho Shin, and Billie F. Spencer. Structural Displacement Measurement Using an Unmanned Aerial System. *Computer-Aided Civil and Infrastructure Engineering*, 33(3):183–192, 2018.

[88] Hyungchul Yoon, Vedhus Hoskere, Jong-Woong Park, and Billie F. Spencer Jr. Cross-Correlation-Based Structural System Identification Using Unmanned Aerial Vehicles. *Sensors*, 17(9):2075, 2017.

[89] R. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[90] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[91] Juyang Weng, Paul Cohen, and Marc Herniou. Camera Calibration with Distortion Models and Accuracy Evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, 1992.

[92] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 666–673, Kerkyro, Greece, 8 1999. IEEE.

[93] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, San Juan, PR, 6 1997. IEEE.

[94] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab, 2015.

[95] Marcin Adamczyk, Paweł Liberadzki, and Robert Sitnik. Temperature Compensation Method for Digital Cameras in 2D and 3D Measurement Applications. *Sensors*, 18(11):3685, 2018.

[96] Satoshi Suzuki and KeiichiA Be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

[97] Dav Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 11, pages 2544–2550, San Francisco, CA, 6 2010. IEEE.

[98] Alan Lukežič, Tomáš Vojíř, Luka Čehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative Correlation Filter Tracker with Channel and Spatial Reliability. *International Journal of Computer Vision*, 126(7):671–688, 2018.

[99] Yong Wang, Wei Shi, and Shandong Wu. Robust UAV-based tracking using hybrid classifiers. *Machine Vision and Applications*, 30(1):125–137, 2019.

[100] Guido Morgenthal and Hagen Höpfner. The application of smartphones to measuring transient structural displacements. *Journal of Civil Structural Health Monitoring*, 2(3-4):149–161, 2012.

[101] T. Kijewski and A. Kareem. Wavelet Transforms for System Identification in Civil Engineering. *Computer-Aided Civil and Infrastructure Engineering*, 18(5):339–355, 2003.

[102] Amin Havaran and Mussa Mahmoudi. Extracting structural dynamic properties utilizing close photogrammetry method. *Measurement*, 150(1):107092, 2020.

[103] Li-Jun Wu, Fabio Casciati, and Sara Casciati. Dynamic testing of a laboratory model via vision-based sensing. *Engineering Structures*, 60(2):113–125, 2014.

[104] C.Z. Dong, X.W. Ye, and T. Jin. Identification of structural dynamic characteristics based on machine vision technology. *Measurement*, 126(5):405–416, 2018.

[105] Reza Aghlara and Mahmood Md Tahir. Measurement of strain on concrete using an ordinary digital camera. *Measurement: Journal of the International Measurement Confederation*, 126(May):398–404, 2018.

[106] Xianglei Liu, Xiaohua Tong, Wensheng Lu, Shijie Liu, Baofeng Huang, Pingbo Tang, and Tongxin Guo. High-speed videogrammetric measurement of the deformation of shaking table multi-layer structures. *Measurement*, 154(3):107486, 2020.

[107] B. J. Perry, Y. Guo, and R. Atadero. Non-Contact Dynamic 3-Component (3C) Displacement Measurements with a Dual-Stereo Vision Enabled Uncrewed Aerial System (UAS). *Journal of Structural Engineering*, Under Review, 2023.

[108] Reoxiang Li, Bing Zeng, and M.L. Liou. A new three-step search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(4):438–442, 1994.

[109] Sabharwal and Guo. Tracking the 6-DOF Flight Trajectory of Windborne Debris Using Stereophotogrammetry. *Infrastructures*, 4(4):66, 2019.

[110] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical Multi-Scale Attention for Semantic Segmentation. Technical report, Nvidia, 5 2020.

[111] Shervin Minaee, Yuri Y. Boykov, Fatih Porikli, Antonio J. Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2021.

[112] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, NV, USA, 6 2016. IEEE.

[113] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Santiago, Chile, 12 2015. IEEE.

[114] Yasutaka Narazaki, Vedhus Hoskere, Koji Yoshida, Billie F. Spencer, and Yozo Fujino. Synthetic environments for vision-based structural condition assessment of Japanese high-speed railway viaducts. *Mechanical Systems and Signal Processing*, 160:107850, 11 2021.

[115] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

[116] Meta Platforms, Inc. Mapillary. Available at https://www.mapillary.com/ (2023/06/01).

[117] Serkan Tapkin, Emre Tercan, Siphesijle Mpho Motsa, and Georgios Drosopiulos. Structural Investigation of Masonry Arch Bridges Using Various Nonlinear Finite-Element Models. *Journal of Bridge Engineering*, 37(7):04022053, 2022.

[118] Tommy Hinks, Hamish Carr, Linh Truong-Hong, and Debra F. Laefer. Point Cloud Data Conversion into Solid Models via Point-Based Voxelization. *Journal of Surveying Engineering*, 139(2):72–83, 2013.

[119] Selahattin Doğan and Hamza Güllü. Multiple methods for voxel modeling and finite element analysis for man-made caves in soft rock of Gaziantep. *Bulletin of Engineering Geology and the Environment*, 81(23):1–20, 2022.

[120] Giovanni Castellazzi, Antonio D'Altri, Gabriele Bitelli, Ilenia Selvaggi, and Alessandro Lambertini. From Laser Scanning to Finite Element Analysis of Complex Buildings by Using a Semi-Automatic Procedure. *Sensors*, 15(8):18360–18380, 2015.

[121] László Kudela, Stefan Kollmannsberger, Umut Almac, and Ernst Rank. Direct structural analysis of domains defined by point clouds. *Computer Methods in Applied Mechanics and Engineering*, 358:112581, 2020.

[122] Andrea Ursini, Alessandro Grazzini, Francesca Matrone, and Marco Zerbinatti. From scan-to-BIM to a structural finite elements model of built heritage for dynamic simulation. *Automation in Construction*, 2022(142):104518, 2011.

[123] Kyle Simek and Mark Reid. Dissection the Camera Matrix, 2013.

[124] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983.

[125] Ansys. Ansys FEA, 2021.

[126] Hiroshi Tada, Paul C. Paris, and George R. Irwin. *Stress Analysis of Cracks Handbook*. ASME Press, New York, NY, 3rd edition, 2000.

[127] Mazin Irfaee and Hussam Mahmoud. Mixed-Mode Fatigue and Fracture Assessment of a Steel Twin Box-Girder Bridge. *Journal of Bridge Engineering*, 24(7):04019056, 2019.

[128] Solveig Melin. Which is the most unfavourable crack orientation? *International Journal of Fracture*, 51:255–263, 1991.

[129] Dassault Systemes. Abaqus FEA, 2020.

[130] Soren N Lophaven, Hans Bruun Nielsen, and Jacob Sondergaard. DACE - A Matlab Kriging Toolbox, 2002.

[131] Gaofeng Jia and Alexandros A. Taflanidis. Kriging metamodeling for approximation of high-dimensional wave and surge responses in real-time storm/hurricane risk assessment. *Computer Methods in Applied Mechanics and Engineering*, 150(3):834–839, 2013.

[132] P. Paris and F. Erdogan. A Critical Analysis of Crack Propagation Laws. *Journal of Basic Engineering*, 85(4):528–533, 12 1963.