MOUNTAIN-PLAINS CONSORTIUM

MPC 21-443 | B.J. Perry, Y. Guo, R. Atadero and J.W. van de Lindt

A STREAMLINED BRIDGE INSPECTION FRAMEWORK UTILIZING UNMANNED AERIAL VEHICLES (UAVS)

Technical Report Documentation Page

1. Report No. MPC-535, MPC-592 (combined)	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle A Streamlined Bridge Inspection Framework Utilizing Unmanned Aerial Vehicles (UAVs)		5. Report Date December 2021 6. Performing Organization Code
7. Author(s) Yanlin Guo Brandon J. Perry Rebecca Atadero John W. van deLLindt		8. Performing Organization Report No. MPC 21-443
9. Performing Organization Name and Address Department of Civil and Environmental Engineering Colorado State University 1372 Campus Delivery Fort Collins, Colorado 80523-1372		10. Work Unit No. (TRAIS) 11. Contract or Grant No.
12. Sponsoring Agency Name and Address Mountain-Plains Consortium North Dakota State University PO Box 6050, Fargo, ND 58108		13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code
15. Supplementary Notes Supported by a grant from the US DOT, University Transportation Centers Program		

Recently, the rapid development of commercial unmanned aerial vehicles (UAVs) has made collecting images of bridge conditions trivial. Measuring a defect's extent, growth, and location from the collected big image set, however, can be cumbersome. This paper proposes a streamlined bridge inspection system that offers advanced data analytics tools to automatically: (1) identify type, extent, growth, and 3-D location of defects using computer vision techniques; (2) generate a 3-D point cloud model and segment structural elements using human-in-the-loop machine learning; and (3) establish a georeferenced elementwise as-built bridge information model to document and visualize damage information. This system allows bridge managers to better leverage UAV technologies in bridge inspection and conveniently monitor the health of a bridge through quantifying and visualizing the progression of damage for each structural element. The efficacy of the system is demonstrated using two bridges.

17. Key Word bridges, data collection, drone aircraft, feasibility analysis, guidelines, inspection, mathematical models, remote sensing		18. Distribution Statement Public distribution	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 58	22. Price n/a

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

A Streamlined Bridge Inspection Framework Utilizing Unmanned Aerial Vehicles (UAVs)

Brandon J. Perry
Yanlin Guo
Rebecca Atadero
John W. van de Lindt

Department of Civil and Environmental Engineering
Colorado State University
Fort Collins, CO 80523

December 2021

Acknowledgements

The funds for this study were provided by the United States Department of Transportation to the Mountain-Plains Consortium (MPC).

Disclaimer

ABSTRACT[1]

The recent rapid development of commercial unmanned aerial vehicles (UAVs) has made collecting images of bridge conditions trivial. Measuring a defect's extent, growth, and location from the collected large image set, however, can be cumbersome. This paper proposes a streamlined bridge inspection system that offers advanced data analytics tools to automatically (1) identify type, extent, growth, and 3-D location of defects using computer vision techniques; (2) generate a 3-D point cloud model and segment structural elements using human-in-the-loop machine learning; and (3) establish a georeferenced elementwise as-built bridge information model to document and visualize damage information. This system allows bridge managers to better leverage UAV technologies in bridge inspection and conveniently monitor the health of a bridge through quantifying and visualizing the progression of damage for each structural element. The efficacy of the system is demonstrated using two bridges.

[1] This study has been published as a journal paper: Perry, B.J., Guo, Y., Atadero, R., and van de Lindt, J.W. (2020). "Streamlined bridge inspection system utilizing unmanned aerial vehicles (UAVs) and machine learning," *Journal of Measurement*, 164 (Nov 2020): 108048.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Inspections of medium to large scale bridges are often cumbersome, expensive, and time-consuming. With the large number of bridges in the United States that require at least a bi-yearly inspection, there is a need to improve bridge inspection techniques to save time and reduce costs. Unmanned aerial vehicles (UAVs) have made tremendous advancements in recent years to allow for better data collection with enhanced sensors, more controllability with precise global-positioning systems and inertial measurement units, and increased safety with omnidirectional sensors to avoid collisions while becoming more affordable. In current bridge inspection practice, UAVs have been used as "eyes-in-the-sky," simply assisting inspectors to view bridges or other structures from different vantage points with the inspectors still taking measurements and making decisions with the traditional, more subjective techniques. However, to take full advantage of the UAV's capabilities, perform quantitative inspections automatically, and create a more streamlined workflow, there is a need for more robust data processing of the information attained by the UAV. A streamlined decision-making support framework is proposed that uniquely integrates UAV-based field inspection, automates data processing, and establishes an element-wise as-built bridge information model (AB-BrIM) for the damage documentation. In this framework, a UAV platform with optical sensors first collects the data through flight missions around a structure. Next, the collected images are fed through an automated damage detection algorithm, which is developed in this study, that identifies cracks using morphological transforms. With this algorithm, quantitative information of cracks (i.e., type, size, amount, and location) is readily highlighted and documented. Next, a 3-D point cloud is created using photogrammetry based on a structure-from-motion algorithm. This generated 3-D point cloud is scaled and georeferenced to be incorporated into current GIS databases. Moreover, to provide component-wise information, the 3-D point cloud is then segmented into its structural elements (e.g., beam, girders, deck, etc.) using human-in-the-loop machine learning clustering techniques. The point's coordinates and surface normals are used to semi-autonomously segment the point cloud into the pertinent components. This segmented 3-D point cloud serves as the base for the AB-BrIM. The identified damage information is automatically linked to each element by mapping the images from the UAV to the 3-D point cloud using the camera poses generated through structure-from-motion. This mapping allows for documentation of component-wise damage information, which is consistent with current bridge inspection practices in the United States. The resulting AB-BrIM with 3-D visualization of elementwise, quantitative damage information offers a transparent condition evaluation tool and has the advantage of also supporting localized storage of any relevant inspection data, including past reports, images collected by the UAV, and repair logs. Moreover, the AB-BrIM incorporates component-wise damage information to facilitate adaptation from research to practice and can greatly ease the planning of repair and maintenance decisions.

1. INTRODUCTION AND LITERATURE REVIEW

1.1. Background

Conventional bridge inspection 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 [1, 2], can be relatively subjective, unsafe, slow, and costly. Inconsistency in ratings due to the subjectivity of human inspectors is very common [3]. Variations of ratings in routine, bi-yearly inspections of ±2 points or greater have been reported for 32% of bridges [4]. For difficult-to-access bridge elements, roping rigs (Figure 1.1a), scaffolding, or "snooper trucks" (Figure 1.1b) are often employed to bring inspectors "within-arm's-reach" of bridge elements according to the FHWA and AASHTO requirements, which may endanger the inspector. For large-scale and in-depth inspections, two weeks or more may be needed for setup, inspection, and tear-down of the equipment with additional time required in the office for reporting. The direct and indirect costs of the current inspection procedure—including renting, purchasing, and operating the equipment, the inspectors' wages, and economic impact due to prolonged traffic interruptions—can be significant. These shortcomings of current bridge inspection practices highlight the need to develop a more consistent, efficient, and cost-effective data collection technique while maintaining a high level of safety for inspectors.

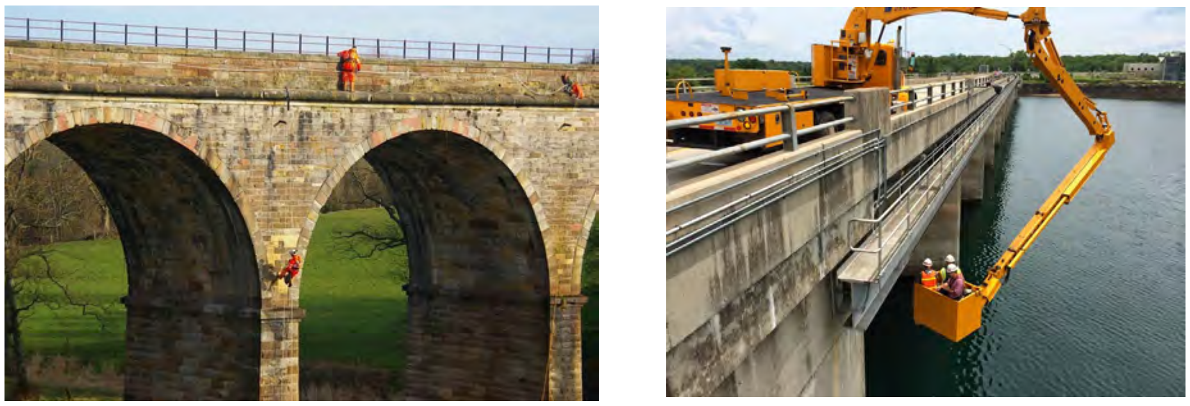


Figure 1.1 Current techniques for accessing the underside of a bridge during inspections using: (a) roping rig [5], and (b) "Snooper" truck [6]

1.2. Literature Review

Recently, unmanned aerial vehicle (UAV) based remote sensing technology has emerged as a promising tool to address these shortcomings and improve bridge inspection practices. With rapid technology developments, UAVs' performance has greatly improved in terms of in-flight stability, maneuverability, and size. The price of a UAV system has been reduced considerably, making the technology more accessible for government and research institutions. Wells and Lovelace have demonstrated the potential for cost savings of about 60% for a bridge inspection contract using a UAV system as opposed to traditional techniques [7]. Some researchers have developed customized UAV systems to take photographs of bridges during inspections [8,9]; however, these studies only used the obtained photographs to aid the inspectors by providing "eyes-in-the-sky" without locating, quantifying, or measuring damage to assess bridge condition. The Minnesota Department of Transportation has completed a three-phase project on utilizing UAVs to assist inspectors on a wide variety of bridges in Minnesota [7,10]. They created photogrammetric models with the use of Pix4D® to log and organize the images collected from the UAVs. The obtained photogrammetric models supplied a wealth of data that would otherwise not be possible; however, the bridge condition assessment still largely relies on manual inspection of the obtained photogrammetric models by inspectors without the aid of any in-depth data analytics tools. Moreover, Duque et al. used a UAV system to inspect two wooden bridges and manually identify the defects in the structure from the collected images [11]. This study greatly improved the inspection process by providing quantified damage assessment; however, this method cannot automatically map the damage location to structural elements, nor can it systematically document the damage information, which may present a gap between the new technology and its application to element-based bridge rating practice. Xu and Turkan developed a framework to create a bridge information model (BrIM) with documented damage [12]. The full framework shows the potential and enthusiasm from the local DOTs to create a UAV-enabled inspection system; however, their 3-D visualization techniques rely on a conversion of the 2-D plan drawings to create mass elements of the bridge to create the BrIM. This conversion from 2-D plans to 3-D BrIM may prove difficult depending on the level of detail of the plans, the changes within the structure during/after construction, or infeasibility due to the unavailability of 2-D drawings of older bridges. Moreover, their framework was not automated. Manual data processing is required to build the BrIM and project the damage location to the structural elements, which might be cumbersome for recurring inspections. Overall, despite the benefits of safety, cost-effectiveness, and valuable inspection data demonstrated in these studies, the function of existing data processing tools is rather limited, and challenges still exist for efficiently post-processing the obtained big data and extracting useful information regarding bridge conditions. The full potential of the UAV-based inspection has not been reached.

1.3. Proposed UAV-Based Bridge Inspection Framework

In response to the challenges posed by traditional bridge inspection practices and the limitations of data processing of recent UAV-based bridge inspection research, this study proposes a holistic inspection system that integrates UAV-based field inspection with advanced data analytics tools to provide streamlined decision-making support for bridge managers. To develop these advanced data analytics tools, the recent advances in image-based computer vision [13–19] and machine learning [20–25] were leveraged. The developed new inspection system has two unique features that are expected to promote fast future technology transfer into current practice: 1) a data analysis module is developed to provide element-wise damage assessment that is consistent with current inspection practices outlined by AASHTO and FHWA, and 2) a geo-referenced as-built bridge information model (AB-BrIM) is established to enable the integration of the new inspection system with a bridge manager's current geographic information system (GIS)-based data management system. An overview of the proposed system is presented in Figure 1.2. A UAV platform carrying visual sensors hovers around the structure collecting data through flight missions. Post-flight, the data pass through a series of data processing tools. With the collected photos, a 3-D point cloud and a photorealistic model are generated using structure-from-motion (SfM) to serve as the basis for an AB-BrIM. From the point cloud, each bridge element (e.g., beam, girders, deck, etc.) is segmented to create an element-wise model using human-in-the-loop machine learning techniques. Next, the images pass through an automated damage/defect detection algorithm to identify cracks, spalling, and scaling of concrete. This provides quantitative information on damage location, type, and severity as well as offers a basis to measure the defect's growth over time. The extracted damage information is then automatically mapped to each segmented element of the bridge in the AB-BrIM using recovered camera poses. The resulting AB-BrIM with element-wise damage information and 3-D visualization capability will make problem areas more obvious and provide quantitative measures to allow bridge inspectors to efficiently rate the structural conditions of existing bridges according to AASHTO's and FHWA's recommendations (e.g., AASHTO's "Guide for Commonly Recognized (CoRe) Structural Elements" or FHWA's "Recording and Coding Guide for The Structure Inventory and Appraisal of the Nation's Bridges"), where the ratings are based mainly on surface conditions, such as section loss, cracking, spalling, scour, and delamination.

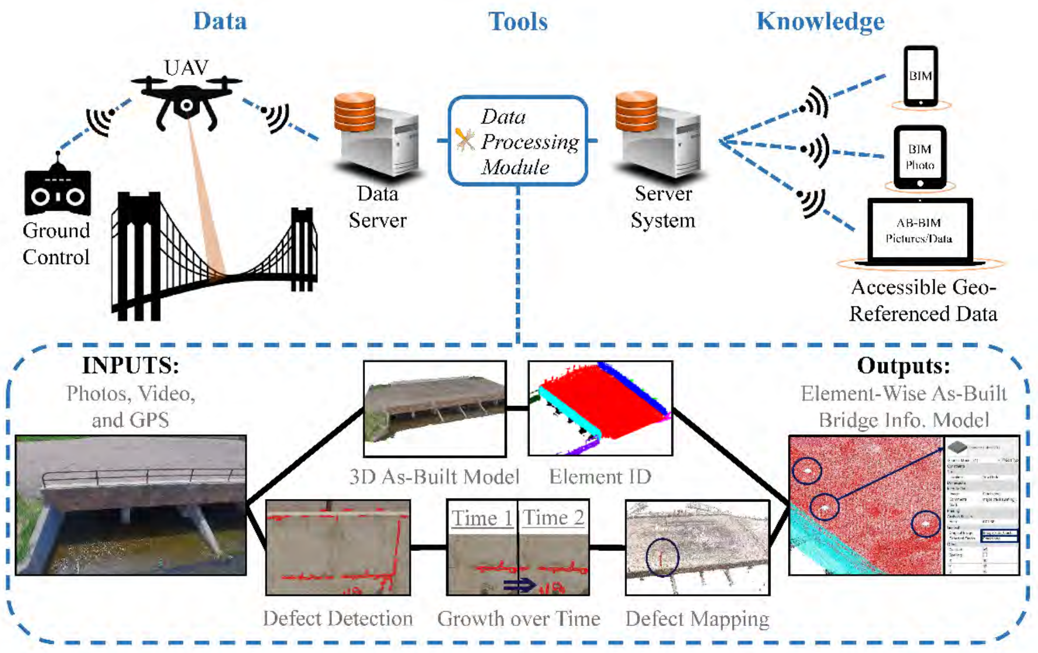


Figure 1.2 Proposed automated bridge inspection system

1.4. Organization of this Report

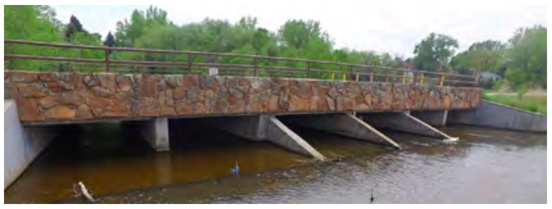
In the subsequent sections, each module of the proposed framework is illustrated through case studies of two bridges. First, the field test procedure and logistical issues are discussed in Section 2. Next, the methodology of the data processing tools and corresponding results of each module are elaborated on. Specifically, the modules discussed are 1) 3-D point cloud and photorealistic model generation using SfM and element identification from the 3-D models (Section 3), 2) damage detection, damage growth tracking, and damage mapping onto the point cloud (Section 4), and 3) AB-BrIM establishment (Section 5). A discussion and comparison between the implementation of the proposed system and existing techniques are conducted (Section 6). Finally, there are some concluding remarks and future directions (Section 7).

2. FIELD TESTS

An overview of the field tests implemented to collect the data will be presented in this section. Two test sites, the Palmer Bridge and the Oxbow Bridge, will be introduced first. Then, the logistics of flight planning and preparation will be discussed with a focus on the considerations specific to bridge inspection.

2.1. Test Bridges

Both the Palmer Bridge and the Oxbow Bridge (Figure 2.1) provided real-world implementation of the proposed system and demonstrated the potential of future UAV-enabled bridge inspections. Both flights were conducted safely and were considered successful. The images and data collected from the Palmer Bridge will be used and analyzed in subsequent sections to demonstrate the potential of UAV-enabled bridge inspections. The flights at the Oxbow Bridge were used to demonstrate the systems used and the feasibility of flying under a structure as discussed in Section 2.4.


(a)

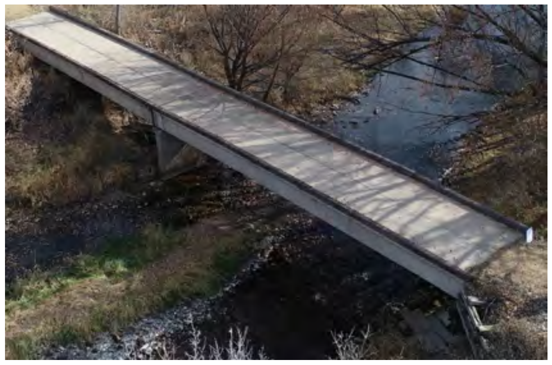

(b)

Figure 2.1 Test bridge locations overview: (a) Palmer Bridge, and (b) Oxbow Bridge

2.1.1. Palmer Bridge

The Palmer Bridge (Figure 2.1a) is composed of six concrete box girders approximately 7 × 12 meters with a 13-cm asphalt overlay and about a 1-meter clearance under the bridge. The last inspection was performed on June 5, 2013. It was reported to be "not deficient," but there were "a few hairline transverse cracks in each span." In 2011, due to required and costly upgrades, low average daily traffic flow, and the difficulty of inspection under the structure, the bridge was closed to vehicular traffic. This bridge provided an excellent case study for the proposed new inspection system as there was minimal, but typical, damage and little to no traffic control needed during the UAV flight. The DJI Matrice 600 Pro (Table 2.1 in Section 2.2) was the only UAV tested at this site; given its wide wingspan, it was deemed unsafe to fly underneath the structure. Therefore, images were only taken for the sides and top, which still provided sufficient information to demonstrate the proposed system.

2.1.2. Oxbow Bridge

The second structure was the Oxbow Bridge (Figure 2.1b). During inspection, the bridge was a private structure over a creek to a landowner's property. Recently, the bridge and some of the surrounding lands were donated to the city; therefore, many upgrades were scheduled shortly afterward to meet safety requirements and ensure the longevity of the bridge. The UAV-based inspection was completed prior to the scheduled upgrades. Due to the high clearance, this bridge provided a good test site for a UAV to fly underneath and collect upward-angled imagery of the underside of the structure. However, around the structure there were overgrown trees and shrubbery, which had to be identified and planned for accordingly. Lastly, since the structure was privately owned during the time of the flight, there was no traffic control required to ensure safe operation. The DJI Mavic 2 Zoom and DJI Phantom 4 Pro (Table 2.1 in Section 2.2) were used. There was some minor damage to the structure mainly due to aging and weather.

2.2. Tested UAV Platforms

Current UAV technology has advanced significantly, making applications in aerial photography, surveying, and infrastructure inspection possible, manageable, and more cost-effective. Leveraging the recent technological advancements, UAVs can provide a significant spatial resolution of objects from above, providing additional perspectives and more data about areas of interest otherwise impossible or impractical to attain. Additionally, from concerns of airspace safety from governmental, private, and public entities, many consumer-level UAVs are equipped with additional optical, infrared, and ultrasonic sensors to detect and avoid obstacles and collisions. These additional sensors also assist in the navigation of the UAV in GPS-deprived environments, which makes it feasible to fly under a bridge or in a more confined area.

Three UAV platforms were tested in this study and are compared in Table 2.1. These UAV platforms were purchased for general applications in multiple research areas at CSU by the CSU Drone Center. Although not specifically acquired for bridge inspection, all three platforms showed effectiveness in the current research applications for structure inspection and were the systems used in multiple research studies [9, 26–31]. The systems were compared on several metrics, including camera, size, sensor payload, and price, and provided an idea of the trade-off and benefit of each system.

One should note that when comparing optical sensors, it is important to consider not only the number of pixels (megapixels) but also the camera's sensor size and focal length. The resolution of real-world objects captured by a camera is determined by the camera's ground sampling distance (GSD), which is a function of the number of pixels, the focal length of the camera, and sensor size. The GSD is the relationship between the pixel in the image and the spatial size of the object photographed in real-world, for example, how many centimeters of the object/ground is represented in each pixel width in the image. It is defined as,

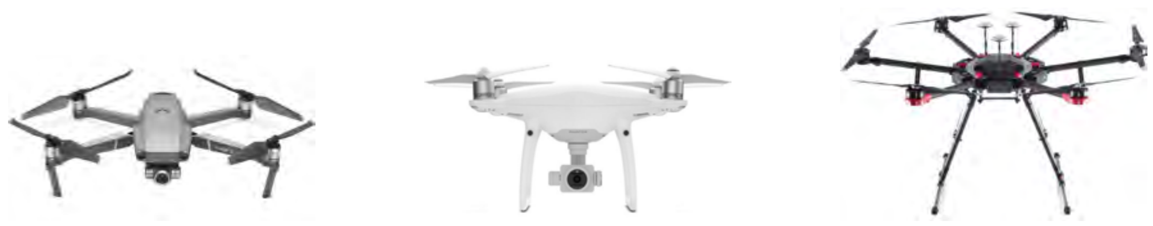
$$GSD = \frac{S_W \cdot H}{F_R \cdot W} \qquad \textbf{(1)}$$

where S_W is the sensor width, H is the height of the camera above-ground-level (AGL), F_R is the focal length, and W is the pixel width of the image. Table 2.1 compares the GSD of the platforms used in this study assuming a 5-meter camera height, H. A smaller GSD means a higher real-world resolution, therefore, leading to better image performance on photogrammetric model generation, damage detection, and damage tracking, discussed in Section 3, Section 4.1, Section 4.2, respectively.

A DJI Mavic 2 Zoom, DJI Phantom 4 Pro, and DJI Matrice 600 Pro were used in this study. These three UAVs were tested in different situations and locations. One platform did not necessarily perform better overall when compared with the others but offered specific advantages and drawbacks, which are discussed in the next subsections. For future implementation, the chosen UAV system(s) should be selected given the requirements of the job, bridge type, location, and other considerations. Each UAV is pictured in Figure 2.2 and the specifications are itemized in Table 2.1.

Table 2.1 Comparison of the tested UAV platforms

	DJI Matrice 600 Pro with Zenmuse X3 Camera	DJI Phantom 4 Pro V2.0	DJI Mavic 2 Zoom
Built-in Obstacle Detection	None	5-Directions	Omni-Directional
Confined Space Flight Practicality	Larger System	No Upward Obstacle Sensors	Compact
Resolution (*MP*)	12.4	20	12
Sensor (*inch*)	1/2.3	1	1/2.3
Optical Zoom	None	None	2X
Flight Time (*min*)	32	30	27
Additional Payload	13.2-lbs	None	None
Wingspan (*feet*)	5.47	1.15	1.05
Weight (*lbs*)	20.9	3.06	0.675
Ground Sampling Distance (*cm/pixel*)	0.214	0.137	0.175
Flight Control and Stability	Sensitive Controls	Average	Corrected by Software
Price (*USD*)	$5,898	$1,499	$1,249



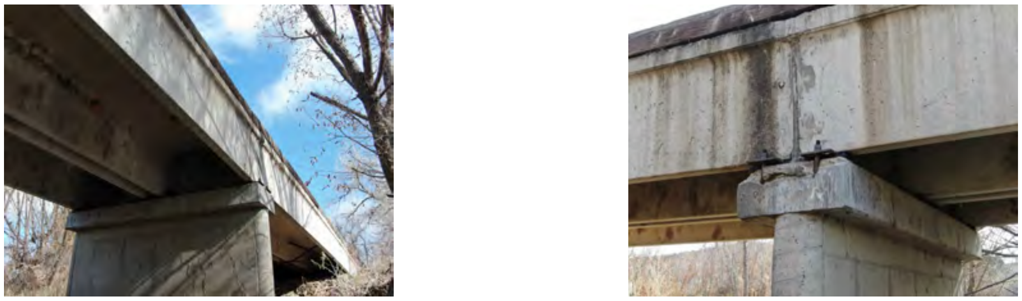
(a) (b) (c)

Figure 2.2 UAV platforms tested: (a) DJI Mavic 2 Zoom - wingspan: 1.05 feet [32], (b) DJI Phantom 4 Pro - wingspan: 1.15 feet [33], and (c) DJI Matrice 600 Pro - wingspan: 5.47 feet [34]

2.2.3. DJI Mavic 2 Zoom

The DJI Mavic 2 Zoom is a smaller platform with easy controllability and zoom capabilities. It is pictured in Figure 2.2 a with its specifications in Table 2.1. The biggest advantage of this system is the 2X optical zoom, which allows more detailed images of specific areas or defects. The camera has a resolution of 12-MP, resulting in the GSD of 0.175-cm/pixel in the horizontal direction hovering at 5-meters AGL. The entire system is small and compact (1.05-foot wingspan), allowing for maneuverability and control under a structure and in tighter spaces; however, due to the small size and weight, it is more susceptible to winds. DJI accounted for this susceptibility of drift automatically in its flight software and sensor array; therefore, the controls are not as sensitive as the larger, more stable Matrice 600 Pro. The Mavic 2 Zoom has an omnidirectional obstacle sensing array that avoids collisions with objects and assists in the stability and navigation of the UAV in GPS-deprived environments. It is the only tested platform with this omnidirectional sensing capability. The Mavic 2 Zoom has a dual visual sensor array on both the front and back, a dual vision sensor and an infrared sensor on the underside, a single vision sensor on the left and right, and an infrared sensor on the top side. With this sensory array built in, if the Mavic 2 Zoom is flying within 20 meters AGL, the onboard processor recognizes key points of the surroundings and can either maintain its course or hover in position. The images were clear due to the stabilization of the camera from the three-axle gimbal attached. The UAV was rated with a battery life of 27 minutes; however, the landing of the UAV was initiated at 20% battery remaining due to safety reasons and the full 27-minute range was not experienced. The batteries are small and simple to exchange.

The UAV proved to be a good candidate for general bridge inspections due to its size and enhanced ability to fly in GPS-deprived environments. A sample of the collected images is shown in Figure 2.3 with Figure 2.3a being a typical image and Figure 2.3b an image with 2X optical zoom. The UAV, however, does not allow for additional payload or sensors. As technology progresses, different or additional payloads (i.e., LiDAR or infrared) could be required. This system does not allow for this customizability.



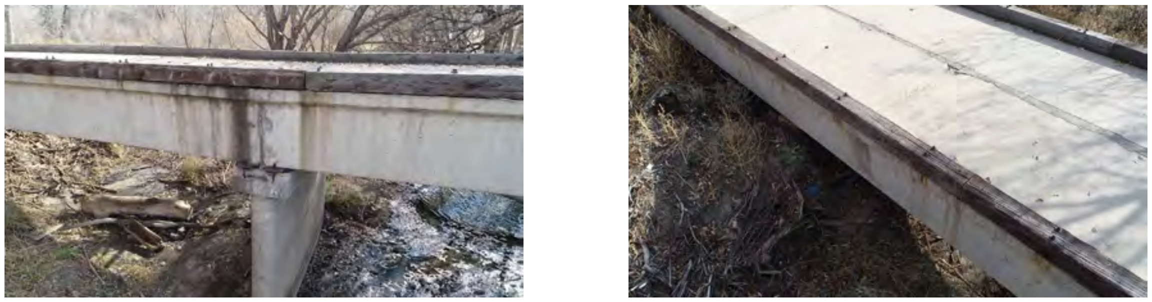
(a) (b)

Figure 2.3 Sample images captured by DJI Mavic 2 Zoom: (a) sample image, and (b) 2X zoom sample image

2.2.4. DJI Phantom 4 Pro

Next, a DJI Phantom 4 Pro (Figure 2.2b) was tested. This is a medium-sized UAV offering obstacle avoidance and a larger megapixel camera with its complete specifications in Table 2.1. The 20-MP camera attached does not have any optical zoom features but delivers a 0.137-cm/pixel resolution at 5-meters AGL. It also has a similar obstacle sensor array as the Mavic 2 Zoom to avoid collisions in the air and actively avoids obstacles (dual vision sensors in the forward, backward, and downward directions, infrared sensors on the left, right, and underside, and an additional ultrasonic sensor on the underside); however, it does not have any upward-facing sensors. Therefore, the obstacle sensing ability is not as robust as the Mavic 2 Zoom. Although it can remain stable in GPS-deprived environments, it cannot sense obstacles above the aircraft, which is important when flying under a structure or in a confined space. The rated flight time is similar to the Mavic 2 Zoom at around 30 minutes; however, the actual flight time was also reduced due to safety concerns and began to land when the UAV reached 20% remaining battery.

Given its ability to fly in GPS-deprived conditions, this system proved applicable for flying under a structure. Additionally, the camera has a smaller GSD than the Mavic 2 Zoom. However, the larger size makes maneuverability in tighter environments more difficult. Also, the aircraft is not customizable and additional sensors and payloads cannot be added. A sample of the images collected during a survey is shown in image Figure 2.4.



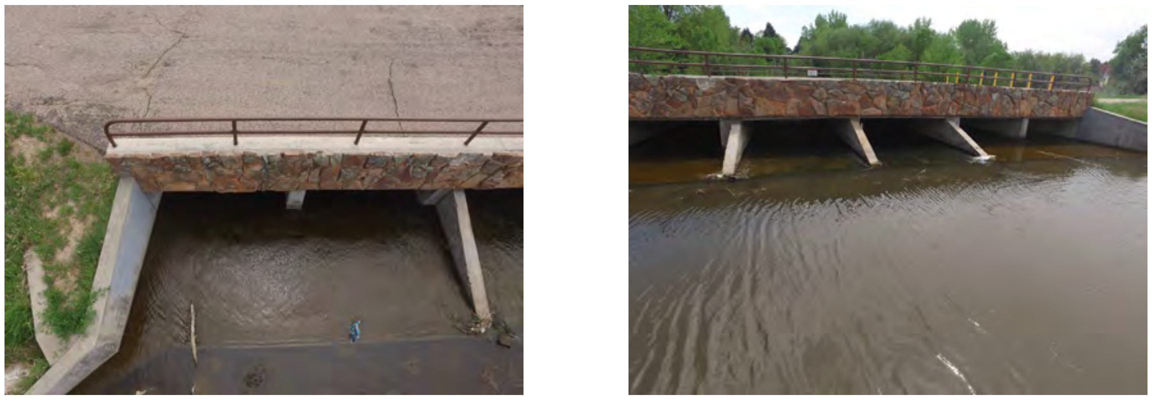
(a) (b)

Figure 2.4 Sample images captured by DJI Phantom 4 Pro: (a) sample image 1, and (b) sample image 2

2.2.5. DJI Matrice 600 Pro

Lastly, a DJI Matrice 600 Pro was tested (Figure 2.2c). This is a larger system and is extremely stable in the air and under winds. Since it is a more professional system, the reaction of the aircraft from the controls is more sensitive, which creates a higher learning curve to pilot when compared with the Mavic 2 Zoom or Phantom 4 Pro. The Matrice 600 Pro has the capability of attaching up to 13.2 lbs of additional sensors and payload either above or below the UAV. The tested camera was a gimbaled DJI Zenmuse X3 with a 12-MP camera. The Zenmuse X3 has a 0.214-cm/pixel resolution hovering at 5 meters AGL, the largest GSD compared with the tested platforms. This camera can be easily exchanged between other DJI cameras, gimbals, or customized systems, allowing flexibility of the sensor payload. This platform included enhanced GPS reliability with real-time kinematic (RTK) positioning technology. This allows for more precise location information of the aircraft during flight. Mulakala et al. showed survey level accuracy with UAVs using RTK positioning technology with a relative horizontal accuracy of 1.2 cm and, when using ground control points, 0.9-cm [26]. This UAV does not have standard obstacle avoidance sensors; however, there are options to add these sensors to the system as an additional payload module. For future work, this system would allow for easy implementation of other equipment such as LiDAR, infrared sensors, or higher-quality cameras. The rated battery life is 32 minutes, but the landing was initiated sooner for safety. The system requires six batteries, which are easy to exchange but are more cumbersome.

Because of its larger size, it would be more difficult to fly closer to a structure or in a tighter area. This system was not tested under a structure to analyze the flight performance in a GPS-deprived environment. Sample images taken by this system are shown in Figure 2.5.



(a) (b)

Figure 2.5 Sample images captured by DJI Matrice 600 Pro: (a) sample image 1, and (b) sample image 2

2.3. Federal Aviation Administration Regulations

With the recent advancements for UAV technology prompting a significant increase in the number of hobbyist and professional UAV pilots in the past couple of years, regulatory agencies ensure airway safety. In the United States, the Federal Aviation Administration (FAA) regulates all airspace and creates the rules and standards professional pilots must follow. In August 2016, FAA's Part 107 rules went into effect. The regulations most pertinent to bridge and structure inspections are listed in the following:[2]

- UAV must be registered with the FAA if more than 0.55 lbs.
- UAV's take-off weight cannot exceed 55 lbs.
- UAV cannot fly above 400 feet AGL unless the UAV is inspecting a structure, in which case the UAV is permitted to fly 400 feet above the structure given that the UAV is still within a "Class E" or "Class G" airspace.
- UAV must be within "visual-line-of-sight" (VLOS) from the "person-in-control" (PIC) at all times and must maintain VLOS without the need for visual aids (i.e., binoculars or camera).
- PIC must be up to date with Part 107's required rules, licensure, and training.
- UAV is not permitted to fly above people unless the people are directly involved with the flight of the UAV.
- UAV is not permitted to fly above moving vehicles.
- Exceptions to most rules in Part 107 can be permitted with written consent from the FAA; however, operating over people and moving vehicles is rarely granted.

The above list is not comprehensive but gives the most pertinent rules for inspection flights. Additionally, the FAA recommends, but does not currently require, having at least two people flying the UAV at one time. The first PIC would be in direct control of the aircraft and the second a "visual observer" (VO). The VO is responsible for constantly maintaining eyesight with the aircraft and constantly verifying the airspace is clear and safe for the PIC. The PIC and VO should be in proximity without communication aids (i.e., walkie-talkie or cell phone). Even though a VO is present, the PIC should still routinely scan the airspace and maintain VLOS with the UAV. The PIC also takes full responsibility for the actions during a flight. CSU requires a PIC and at least one VO to be present during all flights; therefore, the CSU survey team met those requirements.

These regulations are not standard across the world and can vary by country, region, state, or city. Hobbyists are not required to follow Part 107; however, the rules are recommended to ensure safe airspace. Nonetheless, in the United States, any non-hobbyist or compensated pilot is required to follow FAA's Part 107 and register the UAV in FAA's national database. Just over a year after Part 107 adoption, one million UAVs were registered with the FAA, showing the sheer number and common presence of UAVs in the United Stated [35].

2.4. Flying Logistics

Four different flights were planned for the system. Two flights were manually controlled while the other two were autonomously flown. The autonomous flights are discussed in Section 2.5. In the manually controlled flights, two flying techniques were tested: flying parallel to the traffic flow and flying perpendicular to the traffic flow. Separately, both techniques produced similar photogrammetric models with equivalent quality. In the end, the image sets were combined and produced a better model due to the multiple angles captured. Although previous research recommended 50% to 60% overlap in image collection, the image set obtained from the manual flight missions had an overlap of about 95% due to the less precise control on image taking rate [36, 37]. In future studies, the image overlap rate could be reduced

[2] This list is not exhaustive. Additionally, the rules and regulations are updated frequently and should be checked before a flight to ensure compliance.

based on Chiu et al. and Shahnaz et al. recommendations to save on flight time, data storage, and computational expense. A subset of the images collected at the Palmer Bridge site is shown in Figure 2.6.

One major concern for bridge inspections is the loss of GPS signal under the structure. However, due to the advancements in UAV sensory mentioned in Section 2.2, the UAVs tested under the structure performed well in the GPS-deprived environment. On the Oxbow Bridge, in the open, clear space, 12 satellites were locked and positioned. Under the bridge, the number of locked satellites reduced to around eight. The minimum number of satellite connections for a safe and controlled flight according to DJI is 10 to 12. However, because of the sensor array on the underside of the Mavic 2 Zoom and Phantom 4 Pro, the UAV was able to maintain its course and remain stable in the air under the bridge. One note is that because there was flowing water under the bridge, the UAV locked onto the key points of the water flow and slightly drifted in the direction of the water. This drift was maintainable once realized, and control of the UAV was never lost or impeded and caused the flight of the underside of the bridge to take slightly longer than the upper side. The Matrice 600 Pro was not tested under a structure, so the effects of the GPS signal were not tested on a system without the visual guidance and obstacle avoidance sensory array.

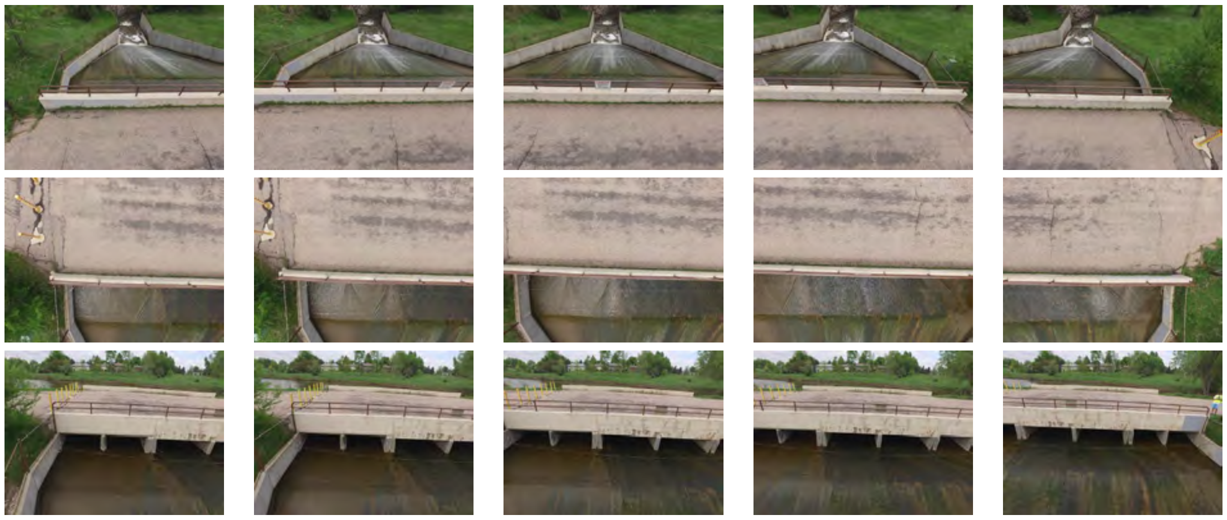


Figure 2.6 Subset of images collected by the UAV at the Palmer Bridge site

The second concern is the safety of piloting a UAV within a confined space. The test flights have shown that the obstacle avoidance sensory array installed on the UAV systems assisted to avoid collisions under the bridge effectively. For instance, the sensors on the Mavic 2 Zoom and Phantom 4 Pro, under factory settings, prevented the UAV from colliding and approaching too closely to the bridge during flight. As a test to the sensors, the Mavic 2 Zoom was piloted at typical flying speed toward an object. The UAV was able to sense and avoid the object and hover at a safe distance even when the PIC manipulated the controls at full throttle toward the object. This obstacle avoidance technology ensured a safe and successful survey of the entire structure.

The third concern in particular for bridge inspection is the performance of upward-facing photography under a bridge. The Mavic 2 Zoom and Phantom 4 Pro are not equipped with upward-facing cameras on the UAV; however, they can point upwards at a 30° and 25° angle, respectively. A sample image from the underside of each system is shown in Figure 2.7. Although this was not the best for the upward imagery, it did provide enough upward angle to capture the underside and create a 3-D point cloud model. The Phantom, when pointed at the 25° angle, did capture the propellers in the imagery. This did not seem to affect the 3-D model or texture creation once stitched. This is shown in Figure 2.7c.

Another consideration for bridge inspection is flying near traffic. Although the UAV would fly within the right-of-way and/or beside traffic (i.e., not directly over moving traffic which is currently prohibited under Part 107) given its unusual and atypical nature, it may pose a risk for a distracted driver. This could prove especially dangerous with UAV piloting crews on the ground adjacent to traffic. It is recommended to implement traffic control during flights. This traffic control will not only slow the traffic but will make the driver cognizant of the crews on the ground. When flying a UAV for any type of inspection project, it is important to not only follow the regulations (i.e., Part 107) but also consider other safety concerns (i.e., traffic control). Additionally, UAVs are not completely accepted by the general public. Although it is currently legal to fly in any authorized airspace, many residents or civilians will wonder what the UAV is photographing and how the data will be used and shared. It is also important to be clear and transparent about the process to ease these worries and have an additional person present during flights to communicate to the community members so that the PIC and VO are not disturbed.

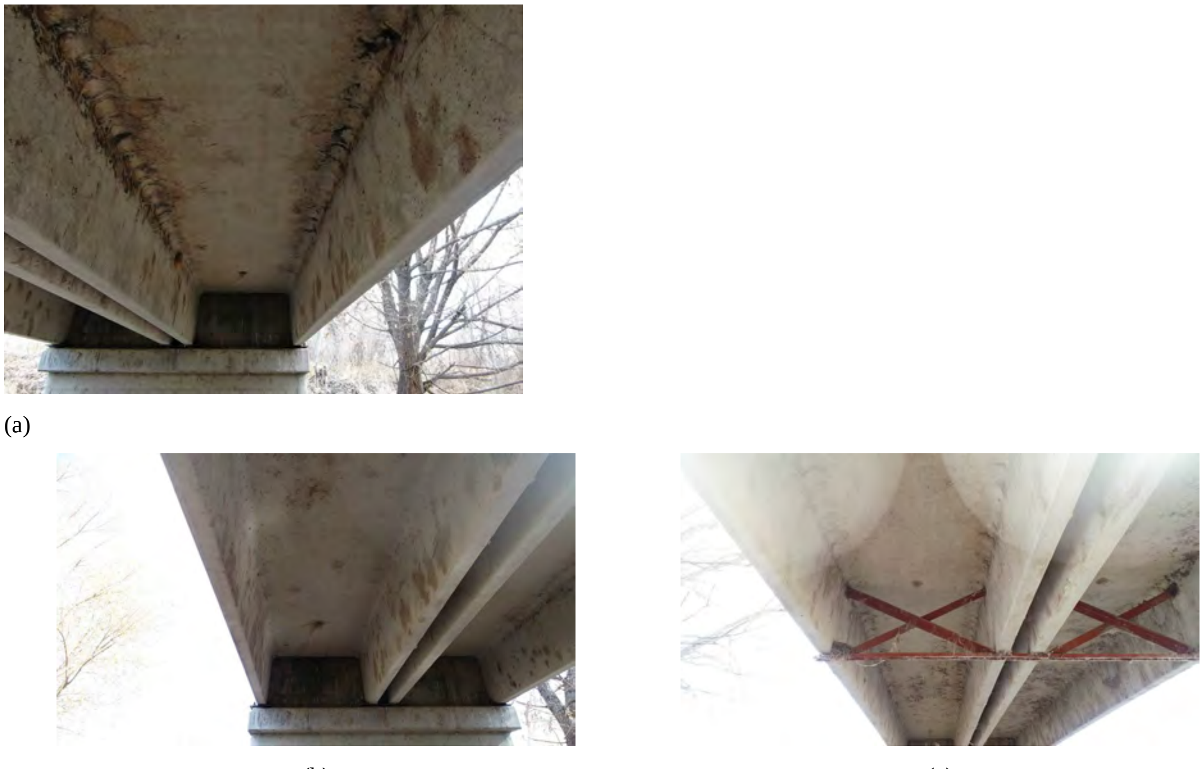


Figure 2.7 Sample images from the underside of the Oxbow Bridge: (a) Mavic 2 Zoom sample image, (b) Phantom 4 Pro sample image, and (c) Phantom 4 Pro sample image with the propellers in the frame

2.5. Future Automation of Flight

The feasibility of autonomous flights was tested for future applications. During the flight of the Palmer Bridge and Oxbow Bridge, two consumer-level applications for autonomous flight control (PrecisionFlight by PrecisionHawk$^{®}$ and Flight1 by SkyCatch$^{®}$) were used to fly the UAV autonomously [38, 39]. Using a smartphone or tablet, points around the bridge were uploaded on-site and a flight plan was created within the applications. The UAV was able to automatically take off, fly around the bridge, capture images, and land without any human interaction during the flight. These two flights lasted three to five minutes but did not provide the best images and angles of the bridge for later photogrammetric modeling.

The reason for the limited image collection for photogrammetric modeling is due to the fixed UAV elevation during the two autonomous flights. When creating a flight plan in the applications, a fixed elevation for the entire flight needs to be set; consequently, the UAV is not able to change vertical elevation during flight. When surveying land or creating digital elevation models, a fixed elevation is extremely useful to evenly capture the entire area; however, for inspecting bridges or other structures, a change of elevation is needed. There are some applications (i.e., Pix4D Capture by Pix4D) that allow for a 3-D path, but it only can fly in a circular motion [40]. This is useful for the facade of a building structure but would not provide the best results for a bridge due to the geometry difference. To improve the image quality and attain the best angles to create a robust photogrammetric model during autonomous flight, a detailed flight plan with elevation markers needs to be created and uploaded to the UAV. This would allow more precise control points with elevations, therefore, permitting better flight control and data quality. Additionally, the manually flown flight logs can be downloaded and saved providing future flights the ability to replicate the same flight path. Future work discussed in Section 7.1 could use these flight plans to assist in defect propagation and more robust health monitoring.

3. 3-D POINT CLOUD, PHOTOREALISTIC MODEL, AND ELEMENT IDENTIFICATION DEVELOPMENT

An important feature of the proposed framework is the 3-D visualization of damage information. The visualization serves as a way for inspectors to easily identify the damage location and retrieve quantified damage information to make more informed decisions about repair and maintenance. As discussed further in Section 5, an inspector can click on a damaged section of the bridge and get detailed, quantified damage information about that particular defect. This aids not only in localized storage of all the bridge‘s information but provides a one-stop decision-making support system. To establish a base model onto which the damage information will be referenced, a 3-D point cloud and 3-D photorealistic model are created using structure-from-motion (SfM). The rising trend is to use 3-D computer-aided drawing (CAD) as a building information model (BIM) during construction. These CAD models could serve as the BIM for the 3-D visualization; however, with 20% of the bridges in the United States over 50 years old [41], these CAD models do not exist for all of the presently constructed bridges. Therefore, a point cloud created with SfM is used as the general base for the AB-BrIM and helps to visualize the geometric information and surface condition of the structure.

3.1. Comparison of LiDAR and Photogrammetry

To create a 3-D point cloud, “light detection and ranging,” (LiDAR) or “photogrammetry” can be implemented to generate 3-D points with associated colors and normals. Both technologies have been well researched in structural surveying and are at a mature state producing 3-D point clouds with similar accuracy; however, LiDAR often contains more 3-D points within the point cloud with slightly more precision [36, 42–47]. A comparison of the pros and cons is shown in Table 3.1. LiDAR uses pulses of light or laser beams to measure the distance from the light source to the object, and, in conjunction with optical sensors, RGB values can be associated with the measured points to create a real-world colorized 3-D point cloud. In contrast, photogrammetry uses a collection of 2-D images taken from various angles and locations around an object to create 3-D points. Since LiDAR readings travel at the speed of light and take micro-seconds to relay (2.0-μs at 300-meters AGL), the systems can take a large number of points in a short amount of time; whereas, with photogrammetry, the number of points is dependent on the number of images and GSD of the camera. Because photogrammetry relies on matching image features to create the 3-D points, there is a significant computational expense compared with LiDAR, which can produce 3-D points in near real-time. However, the only equipment required for photogrammetry is an optical sensor, while UAV-based LiDAR systems require expensive LiDAR sensors and enhanced GPS systems that decrease battery life by adding additional weight to the system. LiDAR offers an advantage of penetration through coverage such as foliage or trees to the ground surface and can better detect thin, linear features such as wires and fences. These two advantages of LiDAR, however, are not necessarily preferable with inspections of bridges as these structures are large and foliage penetration is not required. To create the 3-D base for the AB-BrIM, photogrammetry using SfM was chosen, as it provides a cost-effective system with longer battery life and no additional equipment required.

Table 3.1 Comparison of LiDAR and photogrammetry

	LiDAR	Photogrammetry
Data Acquisition	Pulses of light to measure distances	Measurements from a large library of images
Real-Time Data Processing	Yes	No
Additional Payload	Yes: Enhanced GPS and LiDAR sensor	No: Optical sensor (GPS optional)
Foliage Penetration	Yes	No
Thin, Linear Feature Detection	Yes	Not reliably
Cost	\$50,000+	\$1,000+

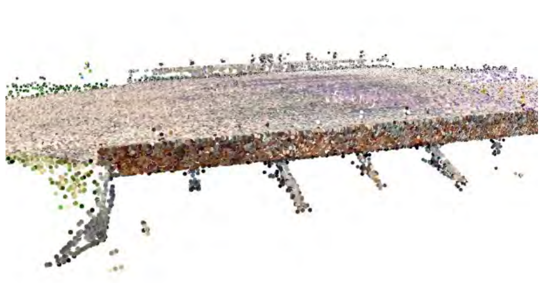
3.2. Point Cloud Generation

Once the UAV flight to survey the bridge was completed, the images collected by the UAV are processed using photogrammetry to create the 3-D point cloud. To accomplish the 3-D point creation using SfM, Meshroom with AliceVision, an open-source photogrammetry code was used [48, 49]. SfM finds key points from every image, in this case through scale-invariant feature transform (SIFT), and stores the identified key points and their associated descriptors [50]. The SIFT algorithm relies on the "difference of Gaussians" (DoGs) algorithm to extract features in the image. The DoGs perform a Gaussian blur convolution over an image and then the previous image is subtracted from the blurred image. This process is repeated 20 times with each time performing a Gaussian blur on the previously blurred image and subtracting it. The local extrema in the DoGs of the different layers were the identified key points. Using the approximate nearest neighbor algorithm, the descriptors of the key points are matched across the images [51]. These descriptors are compared through the image set to match the key points of different images. An acontrario random sampling and consensus (AC-RANSAC) algorithm is implemented to identify inliers and outliers in the dataset [49]. Given the found matches, as well as the focal length and optical center of the camera, the pose of the camera from each image is calculated, which describes the position and orientation of the camera when the image is taken Eq. (2). This camera pose matrix is uniquely found for each image in the dataset.

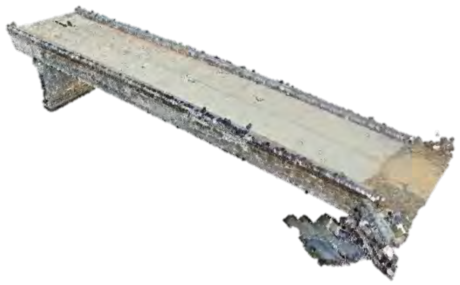
$$\begin{bmatrix} R_C & C \\ 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} I & C \\ 0 & 1 \end{bmatrix}}_{\text{Translation Matrix in World Coordinates}} \times \underbrace{\begin{bmatrix} R_C & 0 \\ 0 & 1 \end{bmatrix}}_{\text{Rotation Matrix in World Coordinates}}$$

$$= \left[\begin{array}{ccc|c} R_{C1,1} & R_{C1,2} & R_{C1,3} & x \\ R_{C2,1} & R_{C2,2} & R_{C2,3} & y \\ R_{C3,1} & R_{C3,2} & R_{C3,3} & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4\times4} \quad \textbf{(2)}$$

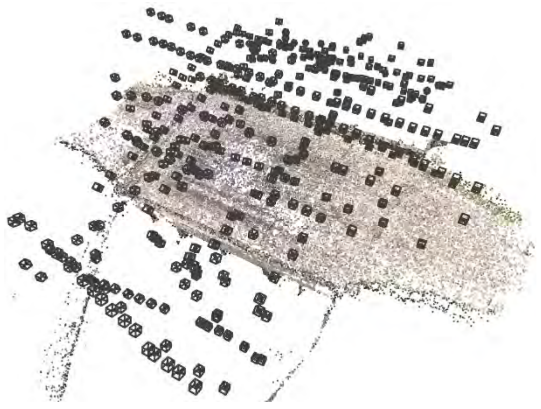
The camera pose is a dot product of a translation vector, C, and a rotation matrix, R_C, in world coordinates, and describes the position and orientation of the camera in camera coordinates. For future decomposition, as discussed in Section 4.3, the matrix is typically made square by adding a row of ***[0, 0, 0, 1]*** to the bottom [52]. This matrix is found for each image in the dataset and is unique for each image. With every camera pose and the matched key points known for each image, 3-D points can be created [49, 53, 54]. The created 3-D point cloud, which is the output from Meshroom, contains points of not only the bridge but also parts of the background and surroundings that were not of importance. To eliminate the points irrelevant to the bridge, an automated algorithm was developed to identify and delete the extraneous points. First, points of the bridge were relatively dense compared with the outlier points; therefore, the points that did not have a specified density, *j*, were deleted. Next, the average value of ***[x, y, z]*** was found to estimate the center point of the bridge. Based on the threshold *k*, the points farther than *k* away were deleted. The final point clouds are shown in Figure 3.1. Figure 3.1a and Figure 3.1b show a rendering of the Palmer Bridge and Oxbow, respectively. Figure 3.1c is an overview of the entire Palmer Bridge with the camera poses from the images collected highlighted with dark grey cubes.



(a)



(b)



(c)

Figure 3.1 Point cloud renderings: (a) Palmer Bridge, (b) Oxbow Bridge, and (c) overview of Palmer Bridge with camera poses identified

With a 12-core Intel® i7 Processor with 32-GB of RAM, the feature extraction and matching took approximately 10 minutes and the creation of the dense point cloud model from about 450 images took approximately 4.30 hours of running time. Because the number of operations to compare each key-point descriptor to each image increases superlinearly, and not linearly, to the number of input images, the processing requirements are not linearly related to the number of input images. Having a smaller overlap ratio as recommended (i.e., 50% to 60%) and therefore, fewer images, is expected to considerably reduce this processing time. If 250 images were used as input, instead of all 450, a 50% efficiency increase is expected. The computational efficiency of the point cloud creation is shown in Table 3.2.

Table 3.2 Computational efficiencies of the 3-D point cloud and photorealistic model generation

	Efficiency		Time	
	Order	Type	(*minutes*)	(*hours*)
Feature Extraction and Image Matching	$\mathbb{O}(n \log n)$	Superlinear	10	0.167
Dense Point cloud Model	$\mathbb{O}(n \log n)$	Superlinear	258	4.30
Mesh	$\mathbb{O}(n)$	Linear	27	0.450
Photorealistic Model	$\mathbb{O}(n \log n)$	Superlinear	26	0.433
		Total:	**321**	**5.35**

3.3. Development of Photorealistic Model

After the point cloud is obtained, the photorealistic model can be developed. This model will be used to help visualize the structure in full detail. Using the 3-D points generated from the SfM algorithm, faces were calculated to construct a "water-tight" triangular mesh around the 3-D points using the screened Poisson surface reconstruction algorithm [48, 49, 55]. Next, the images are stitched together to produce a texture that is overlaid on the mesh. Using GPS ground-truth points collected during the field tests as a reference, the 3-D point cloud is scaled to real-world coordinates and georeferenced, which enables the integration of the new inspection system with a bridge manager's current GIS-based data management system. The final photorealistic model is shown in Figure 3.2. An overview from images to the 3-D point cloud to the 3-D photorealistic model of both bridges is shown in Figure 3.3.

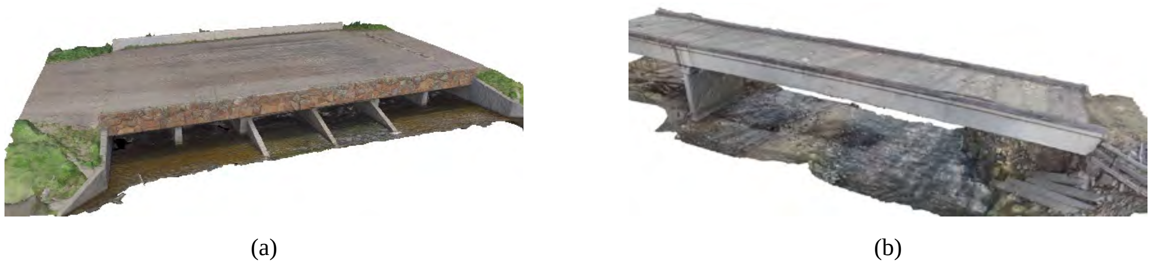


Figure 3.2 Photorealistic model renderings: (a) Palmer Bridge, and (b) Oxbow Bridge

The computational time required to generate this mesh was 27 minutes and the texture and final photorealistic model was approximately 26 minutes. The mesh creation, however, is a function of the number of points in the 3-D model and not of the number of images. The texture generation is reliant on the number of input images. The computational efficiency of the photorealistic model generation is summarized in Table 3.2.

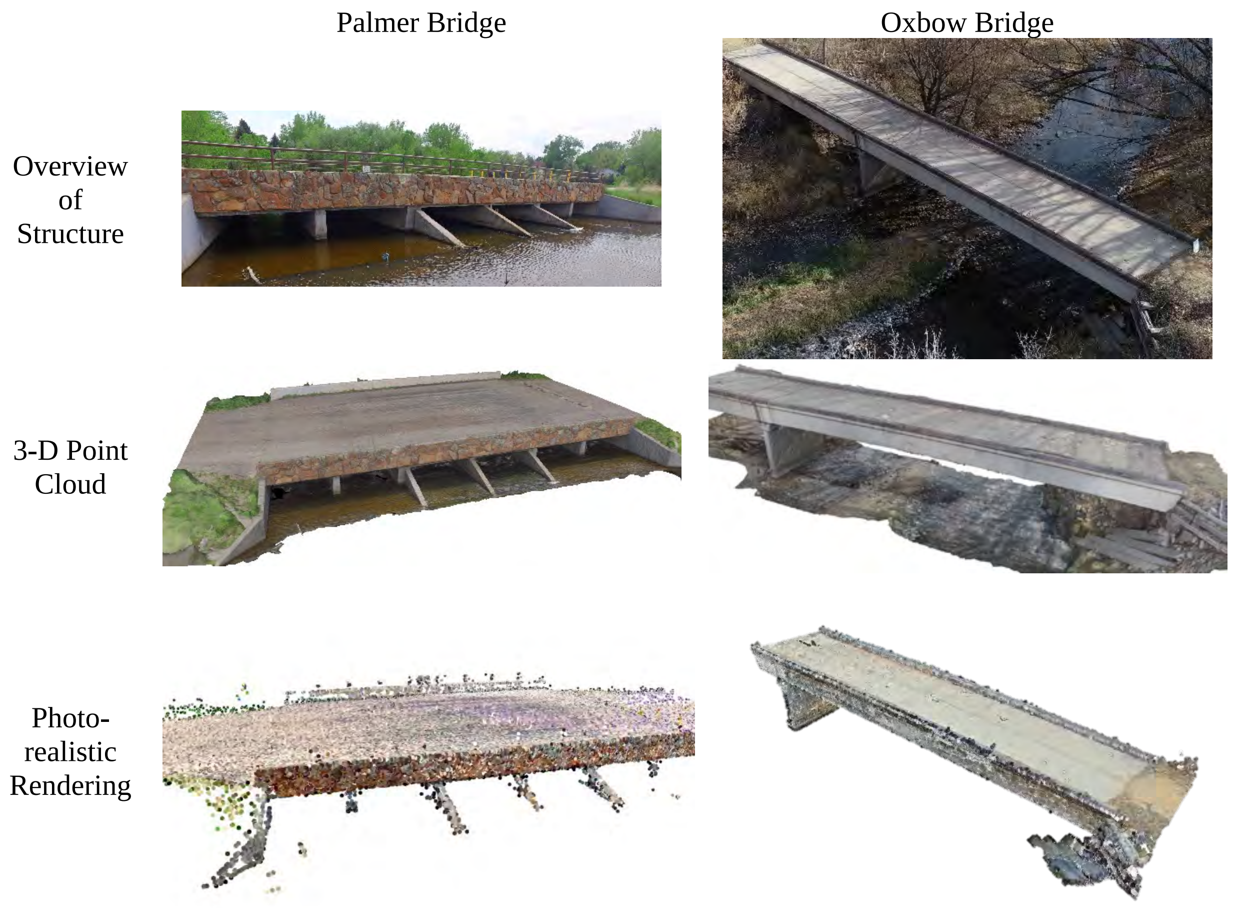


Figure 3.3 Test sites, 3-D point clouds, and photorealistic models

3.4. Element Identification

Current inspection practices regulated by FHWA and AASHTO require the condition rating for each structural element. To ensure seamless merging of the proposed bridge inspection system with current practices, a novel element identification technique is developed to effectively segment the elements of the bridge (i.e., deck, beams, piers, etc.) from the 3-D point cloud, which allows the subsequent automatic mapping of damage information to the corresponding element of the bridge. With the resulting element-wise damage quantification, existing bridge management software, which uses element data to assess repair needs and costs and makes recommendations for maintenance programs, projects, and action types, can be readily utilized. However, identifying structural elements is a challenging step, due to the complexity of the point cloud information, the inconsistencies of points planarity (i.e., "noisy" points), and irregular shapes of the elements. This section will discuss the proposed technique that tackles these challenges.

Using the point cloud for the element segmentation instead of images not only eliminates the need for labeled training images but also provides additional 3-D localization information. Research on the segmentation of point cloud data from existing structures or construction sites is in its early stages; however, some studies have shown success in identifying walls, ceilings, floors, windows, and lights [22–25]. These techniques mainly use denoising of the point cloud to make the points more planar and identify straight flat regions. The denoising technique worked effectively to segment linear and flat features, which lends itself well to rooms and buildings with walls and windows, but may not apply to bridges, as the geometry of a bridge is often more complex and different. Lu et al. proposed a top-down approach to segment the bridge elements of point clouds [21]. This study investigated the separation of piers, pier-caps, and decks using terrestrial laser scan data and found less than a 5% false-positive rate in the segmentation of the piers; however, these terrestrial LiDAR data are more difficult and expensive to obtain, and the proposed algorithm relies on high point density and point planarity to identify the nuances of the data, which may limit its applicability for low-density point clouds obtained from photogrammetry. Moreover, this technique proved successful in a specific type of RC bridge; however, it does not show generality in additional bridge types and geometries.

Given the limitations of the existing approaches, a new point cloud segmentation process is proposed. This process employs two unsupervised machine learning (ML) techniques i.e., Gaussian mixture model (GMM) and agglomerative clustering. The advantage of these unsupervised ML techniques is that they do not need model training and, in turn, do not require a pre-labeled training, validation, or test dataset. To implement these two ML techniques, Python's Scikit-learn package was adopted [56]. A third clustering technique was developed to cluster the 3-D points, which are based on the surface normals of each point within the point cloud. In the following, the detailed methodologies of the two unsupervised ML techniques are introduced first. The GMM assumes the clusters are generated from a Gaussian distribution and creates a probabilistic model of the clusters. The GMM is a Bayesian updating process with unknown parameters, $\boldsymbol{\theta}$, and the prior distribution defined as,

$$p(\boldsymbol{\theta}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\mu_i, \sigma_i) \tag{3}$$

where i is each 3-D point in the point cloud with K elements, weights of ϕ_i, and a normal distribution, $\mathcal{N}$, with mean of μ_i and covariance matrices of σ_i. Therefore, the posterior distribution, which is also the GMM, is defined as,

$$p(\theta|x) = \sum_{i=1}^{K} \tilde{\phi}_i \, \mathcal{N}(\tilde{\mu}_i, \tilde{\sigma}_i) \tag{4}$$

where x is the evidence or each point occurrence. It is then updated using the expectation-maximization algorithm. When using the GMM in this study, it is assumed that each cluster has its unique general covariance matrix. The GMM was chosen because it is most effective for distributions and groups with different, irregular, or oblong shapes such as those found in a point cloud of a bridge. Contrastingly, agglomerative clustering generates a hierarchical dendrogram of the points. This dendrogram recursively merges pairs of the 3-D points based on the minimum Euclidean distances to separate data circumscribable in a sphere; therefore, it is more suitable for clustering more centralized points. In addition to these two unsupervised ML techniques, the surface normals of the point cloud were calculated and used to cluster points belonging to a flat surface. The normals are typically used for visualization to determine the reflectance of light; however, they also give information about the direction of the point by calculating the principal axis of adjacent points using covariance analysis [20]. It is assumed that certain elements of the bridge (i.e., the bridge deck) are relatively level, flat, and smooth surfaces; therefore, the normals would point in relatively the same direction and the angles between the normal vectors would be relatively small. Calculating the angles between the normal vectors of a specified point and of all the remaining points and thresholding the angle by a pre-defined small value allows clustering all the points that have similar surface normals as the specified point. Examples of the surface normals of the bridge deck and a beam are shown in Figure 3.4.

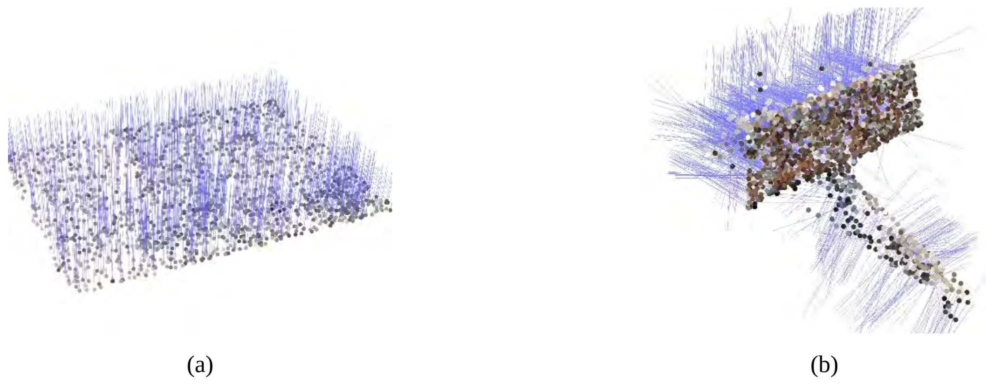


Figure 3.4 Surface normals: (a) bridge deck, and (b) beam and sidewall guard

Utilizing these three tools, a human-in-the-loop machine learning process is proposed. First, the bridge deck will be segmented. Since the bridge deck is assumed to be a relatively level, flat, and smooth surface, the surface normal-based clustering is used. To start the clustering, a user selects only one point on the bridge deck, then the points with the small surface normal angle are automatically clustered together to identify the bridge deck. With the bridge deck segmented out, GMM and agglomerative clustering can be implemented to segment the remaining elements. The exact shape of the structure dictates the order in which to perform the GMM and agglomerative clustering; however, from the two bridges tested, the order remained consistent. A GMM is performed over the entire structure, segmenting the substructure from the superstructure. Next, agglomerative clustering is used to further segment the elements of the superstructure. Since the remaining points are more centralized and can be circumscribed in a sphere, agglomerative clustering produces better and faster results when compared with GMM, K-Means, Ward, and BIRCH clustering for clustering the remaining elements of the superstructure. Additionally, if a pier had a similar relatively flat, planar appearance, such as in the Oxbow bridge, surface normals could again be implemented to segment out the pier. With human-in-the-loop, the order of implementation of the three clustering techniques can be adapted easily by a user to allow the application of the proposed technique to a variety of bridges. This proposed element identification technique incorporates both the power of machine learning and the benefit of minimal user inputs to enhance its applicability to generalized bridge structures. In this element identification process, an initial segmentation of the point cloud (first row of Figure 3.5) is achieved in less than three minutes. Further refinement utilizing automated GMM and agglomerative clustering resulted in the final segmentation shown in the second row of Figure 3.5, where each color group represents each identified element. This process is much more efficient compared with a manual approach (i.e., manually selecting the points and segmenting the point cloud), which would take up to one hour. Using a manually segmented and labeled point cloud for validation, the proposed method achieved high segmentation accuracy for the Oxbow and Palmer Bridge, 99.5% and 99.4%, respectively, with a total processing time of about 10 minutes (including the user's input time). The accuracy could continuously be increased, but with the cost of added user time. This integrated clustering technique performs much better than adopting a single cluster method, such as only using GMM or agglomerative clustering, where the accuracy is significantly lower (below 50%). Note that after identifying the elements using ML, a user still needs to manually label the element as a "girder," "pier," etc., which is the common consequence of unsupervised ML.

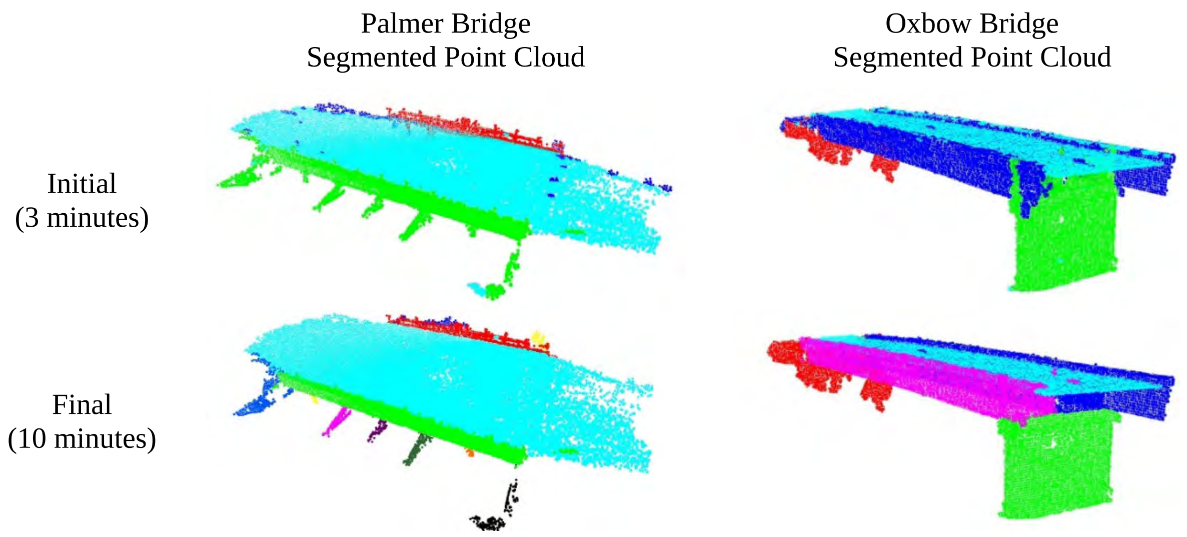


Figure 3.5 Element segmentation of point clouds

4. DAMAGE IDENTIFICATION, GROWTH TRACKING, AND LOCATION MAPPING

In the current human-based bridge inspection practice, the damage ratings mostly come from qualitative categorical designations; the severity of damage is not explicitly delineated on a rating scale. Making maintenance and repair decisions in this seemingly ambiguous setting is cumbersome and may require additional field inspections, increasing management costs. To address this shortcoming and enhance the efficiency of UAV-based inspection, this study proposes an automated damage detection algorithm to identify defects and measure the defect's growth over time in concrete structures from images. Furthermore, these detected defects are mapped onto the 3-D point cloud to obtain location and size information in real-world coordinates, which will provide bridge managers with quantified information to better assess the structural condition and more precisely track the progression of defects at specific locations over time.

4.1. Damage Identification

Detecting and measuring defects, such as cracks and spalling in concrete, from images can be challenging due to the low luminance of the defect and small widths of the cracks. There are several studies on defect detection for concrete surfaces [17, 18, 57]. Because the methods proposed in these studies were tailored to their specific applications (i.e., exterior concrete facades and laboratory concrete beam tests), they worked well in their respective environments but did not produce satisfactory results when used with the images of concrete bridges collected by a UAV. Recently, convolutional neural networks (CNNs) have made tremendous progress in image classification and recognition [19, 58–62], thus have become another alternative for damage/defect identification. Using CNNs on the images from the UAV flight requires first labeling a large set of images to create a training and validation set, which, however, would negate the purpose of creating an automatic defect detection algorithm. Therefore, until large enough datasets of concrete defects are assembled to allow CNNs become more generalizable, a new defect detection algorithm that does not require a significant amount of training data is proposed for this study. This method relies on computer vision and is optimized for the defects typically found in concrete bridges with the process shown in Figure 4.1.

The proposed damage detection algorithm uses a black hat transform and canny edge detector to identify defects in an image. It was developed in Python and implemented functions in the open-source OpenCV library [63]. In the following, the steps of this algorithm are introduced with the outputs of each step shown using three examples: 1) the north side of the Oxbow Bridge's double tee beam (column 1 in Figure 4.1); 2) the south side of the Palmer Bridge (column 2 in Figure 4.1); and 3) the eastern top road deck of the Palmer Bridge (column 3 in Figure 4.1). The algorithm and its parameters remained the same between the two bridges. In the first step, automatic preprocessing of the image was performed, which enabled future steps to produce better results by emphasizing the colors in potential defect regions. The original images were changed to gray scale and minor color correction was performed, i.e., the contrast was increased while the brightness was reduced. To finish the preprocessing step, OpenCV's "non-local means" denoising and a Gaussian blur convolution were used to smooth the image and allow for the defects to become more visible (second row in Figure 4.1). This denoising technique replaces the pixel color with an average of the colors of similar pixels. The idea behind non-local means denoising is assuming that the most similar pixels do not necessarily lie near the pixel in question and could be located anywhere in the image [64]. Therefore, a "research window" is used to search for the most similar pixels. The pixel-wise implementation of non-local means denoising is shown in Eq. (5).

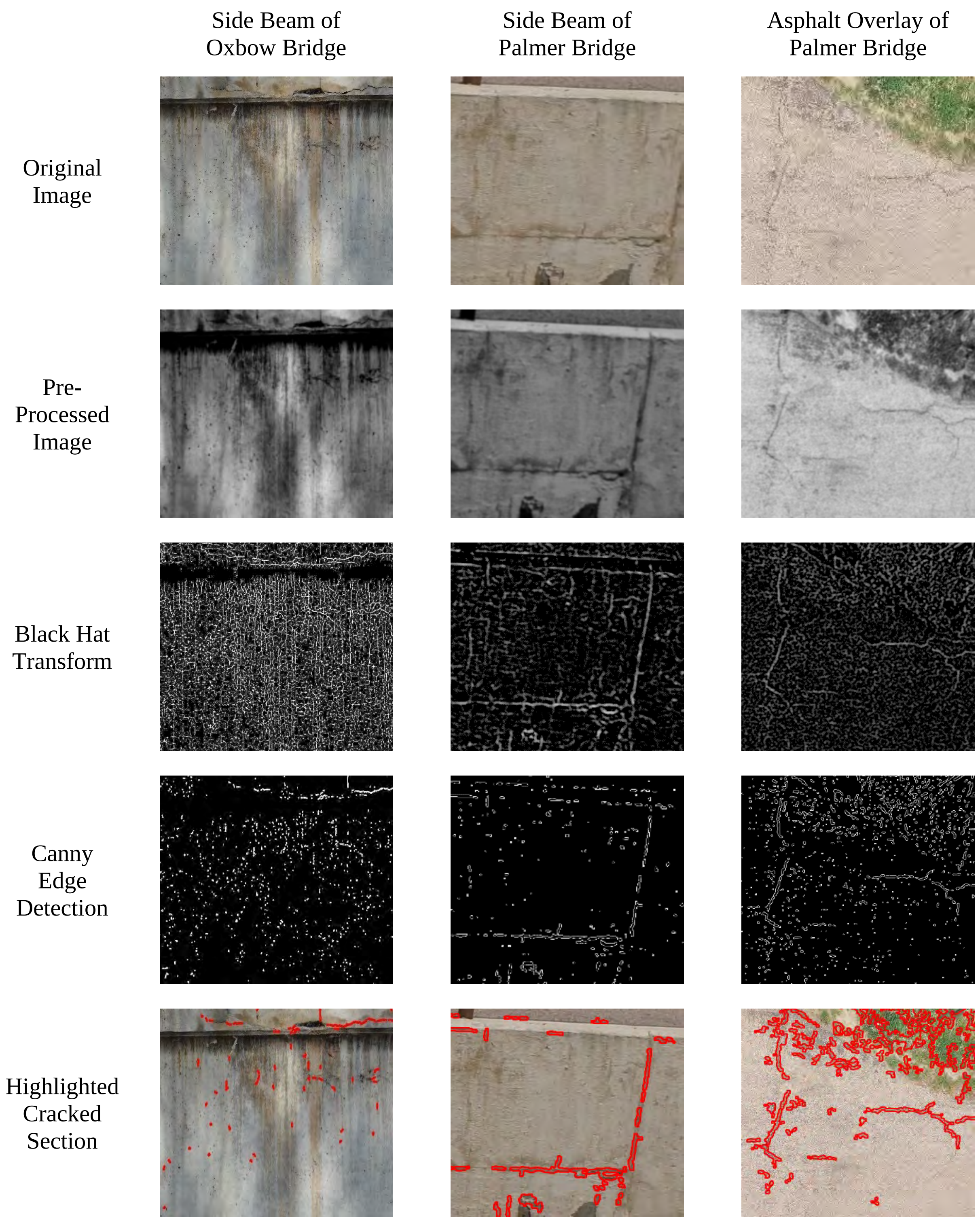

Side Beam of Oxbow Bridge
Side Beam of Palmer Bridge
Asphalt Overlay of Palmer Bridge
Original Image
Pre-Processed Image
Black Hat Transform
Canny Edge Detection
Highlighted Cracked Section

Figure 4.1 Damage detection algorithm output

$$\hat{u}(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u(q) \cdot w(p,q) \tag{5}$$

<u>Where:</u>

$$C(p) = \sum_{q \in B(p,r)} w(p,q)$$

where $\hat{u}(p)$ is the new pixel color, $u(q)$ is the original pixel color in question, $B(p,r)$ is the research window, $w(p,q)$ is the weight factor, and $C(p)$ is the normalization constant composed of the sum of all the weight factors in the research window. To finish the pre-processing step, a Gaussian blur was placed over the image to also help in reducing the noise in the image. This is accomplished by convolving the image with a Gaussian kernel. It was found through trial and error that the 4×4 kernel in Eq. (6) worked best.

$$K = \begin{bmatrix} 0.0145 & 0.0467 & 0.0467 & 0.0154 \\ 0.0467 & 0.1413 & 0.1413 & 0.0467 \\ 0.0467 & 0.1413 & 0.1413 & 0.0467 \\ 0.0154 & 0.0467 & 0.0467 & 0.0154 \end{bmatrix} \tag{6}$$

In the second step, a black hat transform was performed on the pre-processed images. A black hat transform is based on mathematical morphology and is the difference between the closing operation of an image and the input image,

$$T_{BHT} = f \bullet b - f, \tag{7}$$

where f is the original, gray scale image, b is the rectangular structuring element with a size of (x', y'), and $\bullet$ is the closing operation, which is defined as a dilation followed by an erosion of the image. The dilation function is a convolution based on b, defined as,

$$dilation(x,y) = \underbrace{\max}_{(x',y'):element(x',y') \neq 0} f(x + x', y + y') \tag{8}$$

and the erosion is defined as,

$$erosion(x,y) = \underbrace{\min}_{(x',y'):element(x',y') \neq 0} f(x + x', y + y') \tag{9}$$

The black hat transform was performed twice with two different structuring elements with size 3×1 and 1×3 rectangles as recommended by [57]. The black hat transform produces a gray scale image where the white portions are regions of greater contrast compared with adjacent pixels and highlights potential defect areas (row 3 of Figure 4.1). To help reduce anomalies due to the colors of the image or dirt on the structure, a Gaussian blur is convoluted over the transformed black and white image. Lastly, a canny edge detector [65] outlines the white regions within the transformed black and white image (row 4 of

Figure 4.1). To perform the canny edge detection, first, the derivatives in the horizontal, x, and vertical, y, directions need to be approximated. This is found using the Sobel operator defined as,

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \otimes f \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \otimes f \tag{10}$$

where $\otimes$ is the convolution operator and G is the approximation of the derivative in each direction. Next, the magnitude, G,

$$G = \sqrt{G_x^2 + G_y^2} \tag{11}$$

and angle, θ,

$$\theta = \arctan\left(\frac{G_x}{G_y}\right) \tag{12}$$

are to be found. The angle is rounded to one of four directions representing the horizontal, vertical, and the two diagonals. Non-maximum suppression is used to decide if a pixel is a local maximum within its neighbors and, therefore, represents an edge. Last, the non-suppressed pixels are thresholded to determine the strongest edges with the assumption that edges are long lines. For this application, the minimum and maximum values for the thresholding are 50 and 150 out of 255-pixel intensity. The obtained outlines include all the possible defects in the image, but there is still a considerable amount of extraneous detection associated with the output. Therefore, a minimum threshold area of the outlines was set to 20 pixels in this study, which equates to about 2.5 cm^2 in an average sense (the exact area is dependent upon the camera's parameters and distance from the structure). Defect areas less than this threshold were discarded, eliminating most of the extraneous outlines while preserving the defects. The final output images with identified defects are shown in the last row of Figure 4.1.

4.2. Damage Growth Tracking

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 UAVs every two years (the typical inspection frequency in the United States), the image sets from these successive inspections are comparable and 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.

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 developed 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 aforementioned SIFT algorithm is utilized [50]. SIFT key-points 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 key-points in the previous and current images, the linear transformation is defined as,

$$T = M \times X; \quad X = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{13}$$

where T is a matched key-point from the current image, X is a matched key-point of the previous image, and M is the linear transformation matrix. M is first solved using the known matches of the key-points, and then the linear transformation is applied over the entire image.

After the alignment, the developed crack detection algorithm introduced earlier is performed on both the previous and current images; however, the algorithm is stopped after the black hat transform. Then, the resulting black and white image of the current state is subtracted from the previous image to find the remaining black hat transforms (i.e., the difference between the two images, namely the “difference image” in the following), which indicates the damage growth. In the end, the Gaussian blur convolution, thresholding, and outlining steps (as described in the defect identification algorithm) are performed on the difference image to highlight the areas of crack growth; thus, the extent of crack growth is quantified.

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 ASTM C496 [66]. 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 4.2). This compressive force creates tension in the center of the cylinder, which causes a crack to gradually develop within the specimen. To record the crack growth, two UAVs (DJI Mavic 2 Zoom [UAV #1 in Figure 4.2] and DJI Mavic 2 Pro [UAV #2 in Figure 4.2]) 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 UAVs in the experiment intends to simulate the real-world scenario, where one UAV might take different paths during the two inspections or different UAVs, with different cameras, may be used in the two inspections.

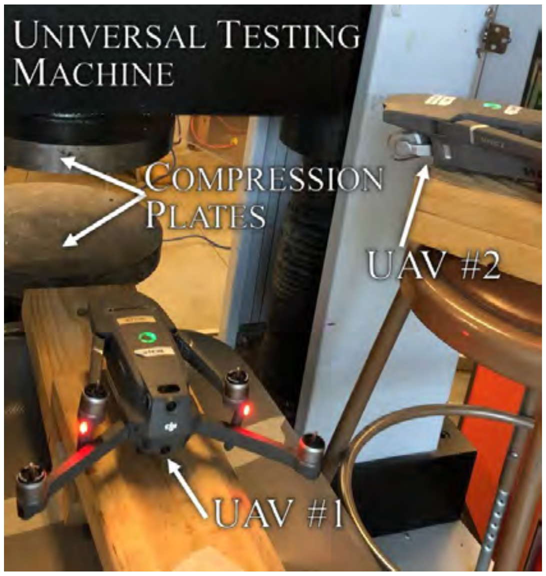


Figure 4.2 Setup of crack-growth experimental data collection

Data were collected from two split cylinder tests and used for validation of the proposed method. The first test used two images (Figure 4.3a and Figure 4.3b) taken by the same UAV (UAV #1 in Figure 4.2) 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, the black hat transform, and finding the difference as explained earlier, Figure 4.3c 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 previous and current images are taken at different angles by different UAVs during different inspections. In the second test, the initial image (Figure 4.4a) was still obtained from UAV #1 (Figure 4.2), while UAV #2 (Figure 4.2) was used to collect the image with the progressed cracks over time (Figure 4.4c). The affine transform was applied to the initial image (Figure 4.4a), resulting in the warped image (Figure 4.4b) that can be readily compared to the current image (Figure 4.4c). Figure 4.4d shows the identified crack growth. From these two tests, it is shown 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. One should note that the basis of the defect growth tracking algorithm is the damage detection algorithm, which has been validated on two tested bridges. 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 key-points. SIFT key-point matching has been used to create 3-D point clouds and has proven 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.

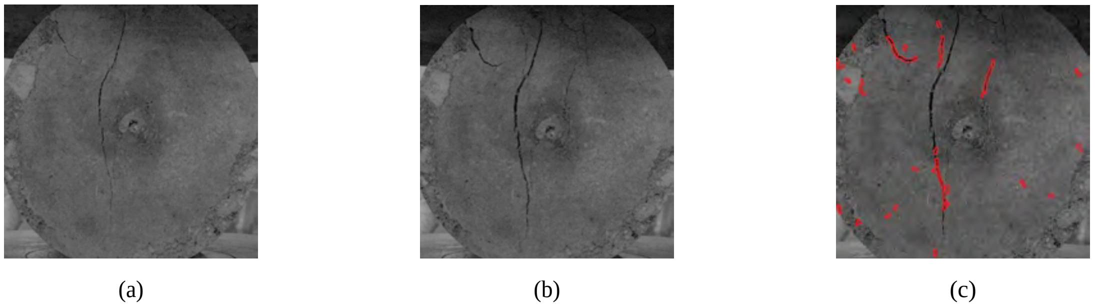


Figure 4.3 Crack growth experiment 1 results: (a) previous image, (b) current image, and (c) difference image (crack growth highlighted)

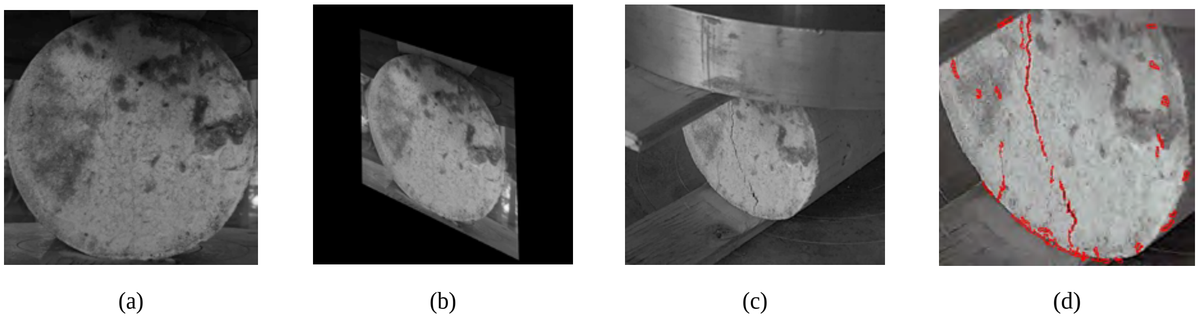


Figure 4.4 Crack growth experiment 2 results: (a) previous image, (b) affine transformed previous image, (c) current image, and (d) difference image (crack growth highlighted)

4.3. Damage Location Mapping

Beyond the image-based damage detection and growth tracking, this study developed a damage mapping technique to relate the defects on 2-D images of arbitrary scale to a 3-D point cloud of real-world scale, aimed at recovering the damage size and location information that is critical for structural condition assessment.

Two techniques were tested to map the detected defects, found using the proposed algorithm discussed in Section 4.1, from the 2-D images onto the 3-D point cloud. In the first technique, the created texture from the photorealistic model development was passed through the defect detection algorithm. The second technique was to use the generated camera pose matrix from the point cloud creation process. Therefore, the techniques will be discussed and analyzed in Section 4.3.1 and Section 4.3.2, respectively.

4.3.1. Damage Mapping Based on Texture Extraction

The first technique to attempt to map the defects detected in a 2-D image into 3-D space was to use the texture image created in the process of building the photorealistic model, as discussed in Section 3.3. This texture is a set of images, originally used to create the 3-D point cloud, stitched together, and essentially overlaid on the mesh of the model (shown in Figure 3.2). The texture image can have a fairly high pixel resolution of 268 MP or more. A sample of one of the two textures used to create Figure 3.2 is shown in Figure 4.5.

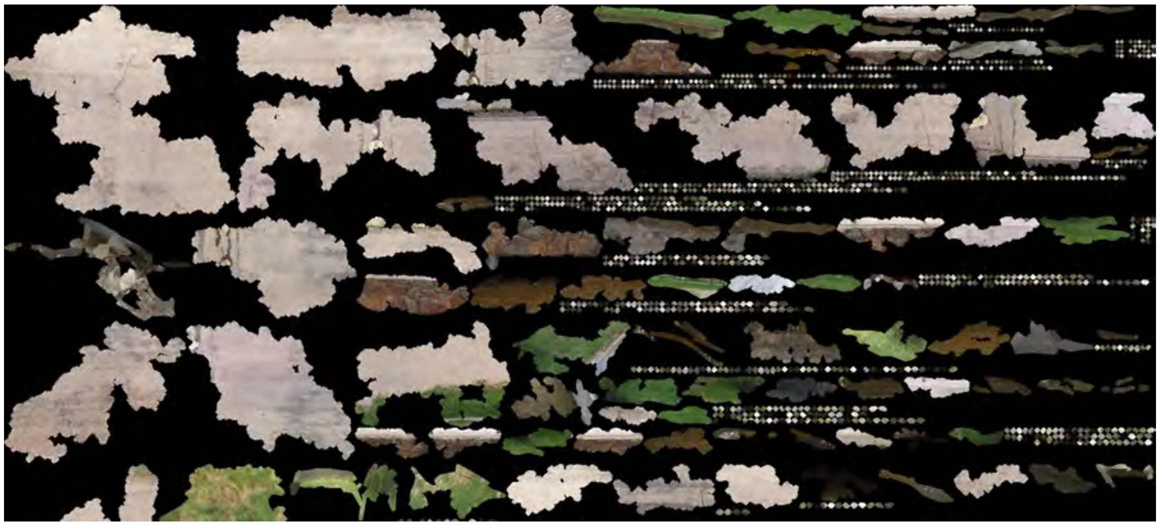


Figure 4.5 Original texture used to create Figure 3.2

As opposed to a subset of the image collected by the UAV passing through the damage detection algorithm to highlight the defects, only the high-resolution texture image was used and the defects on the texture image were highlighted. Using the same parameters and technique as discussed in Section 4.1, the detected defects were highlighted in red. The texture with highlighted defects is shown in Figure 4.6.

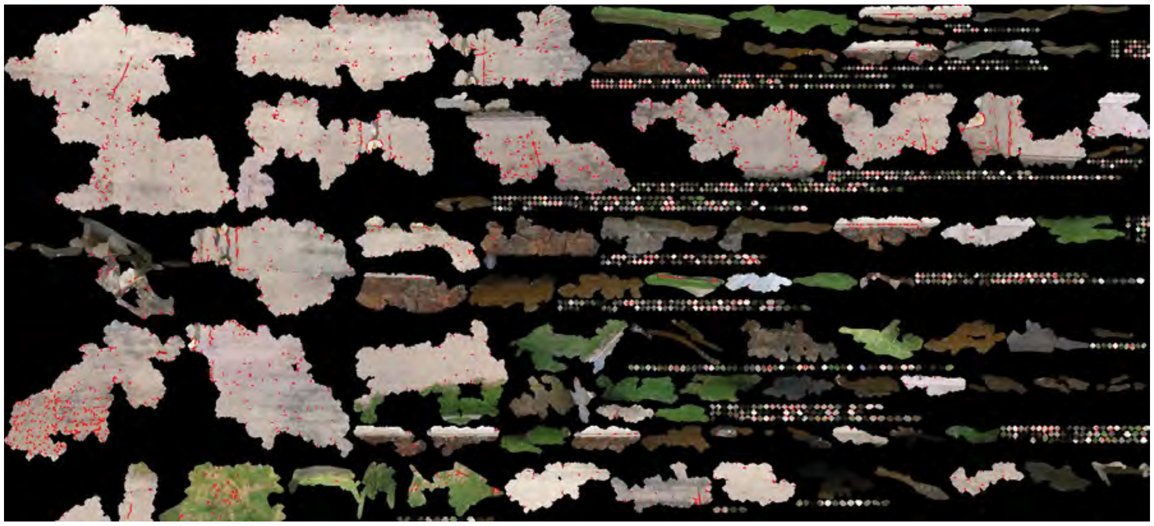


Figure 4.6 Detected defects in the texture of the photorealistic model

Once the defects were detected in the two textures, they were used to overlay on the generated mesh with the defects highlighted. After this, the defects are mapped into 3-D real-world coordinates and the location and size of the defects are known. The result is shown in Figure 4.7.

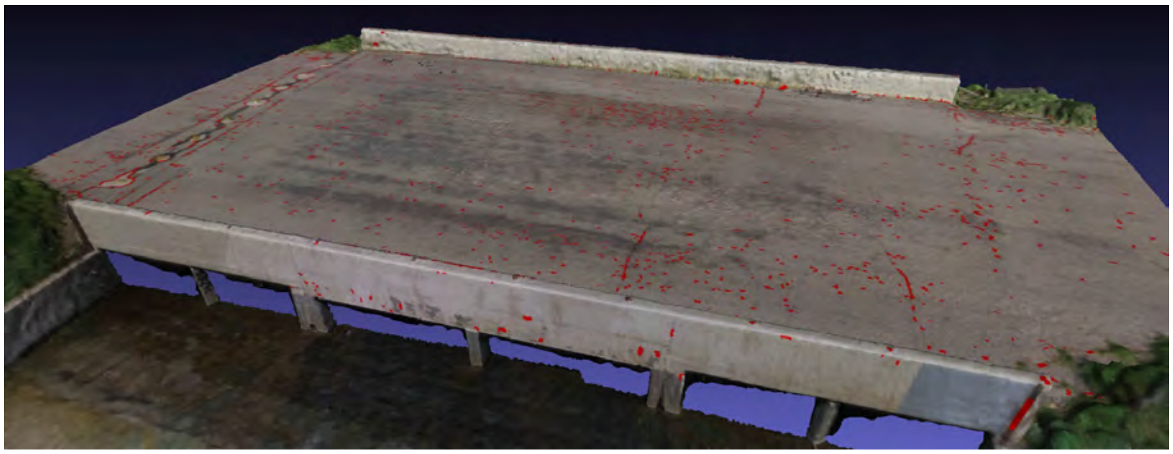
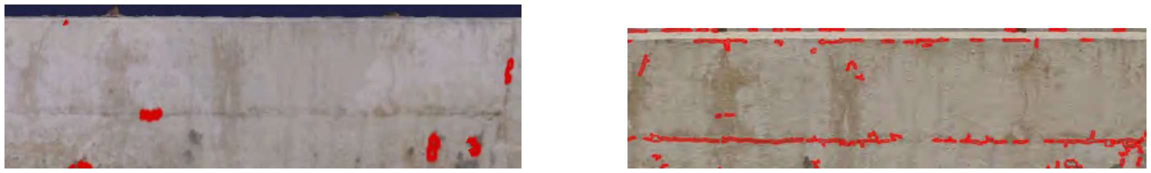


Figure 4.7 The photorealistic model with the texture passed through the damage detection algorithm discussed in Section 4.1

There were some problems with this technique. Even though the texture has a substantially higher resolution than the original image (268 MP in the texture as opposed to 12.4 MP in the original images), the texture was created by stitching the original images together. Inherently, there was some loss of quality in the images, and either there was significantly more noise in the image or there was not enough information left in the stitched images. One example of this is in Figure 4.8. Figure 4.8a is a close-up on the side of the photorealistic model with the texture passed through the defect detection algorithm. Figure 4.8b is the same area as the original image taken by the UAV passed through the defect detection algorithm. There is significantly more detail and better damage identification in Figure 4.8b when compared with Figure 4.8a.



(a) (b)

Figure 4.8 Comparison of defect detection using: (a) texture mapping technique, and (b) the originally proposed technique

The comparison in Figure 4.8 indicates a better solution is needed to map the detected defects from the 2-D images into 3-D real-world coordinates. Section 4.3.2 proposes a different and more successful technique to accomplish this.

4.3.2. Damage Mapping Based on the Unique Camera Poses

To take full advantage of the higher resolution images from the UAV, the next technique uses the original image, the image passed through the damage detection algorithm, the 3-D point cloud, and the camera pose of the image to map the 2-D image onto the 3-D point cloud. The camera pose is a matrix that relates the 2-D image in pixel space to the 3-D environment in camera coordinates. Each camera pose was recovered for each image during the construction of the point cloud. Knowing the camera pose, the proposed technique relates each image to the 3-D point cloud. In this technique, the 3-D points are first indexed and then transformed into 2-D points in pixel space using the camera poses of each UAV-taken image, which were calculated during the creation of the 3-D point cloud, to create a new set of 2-D "point-images." Through one-to-one matching, the pixel points of the defects in the UAV-taken image are mapped to the 2-D points in the point-image. Then the 2-D point-image is transformed back into the 3-D point cloud with the defects mapped in 3-D space. In the following, the procedure of this technique is introduced in detail.

Before discussing the specific algorithm used to automatically relate the defects in 2-D pixel space to the 3-D real-world coordinates, the camera matrix, P, must first be defined. The camera matrix relates the 2-D pixels in pixel space to the real-world coordinates. It can be decomposed into two matrices, the *Intrinsic*, K, and *Extrinsic*, E, Matrices, Eq. (14).

$$[P]_{3\times4} = [K]_{3\times4} \times [E]_{4\times4} \quad \textbf{(14)}$$

The *Intrinsic Matrix*, K, gives information about the camera itself and the *Extrinsic Matrix* is a transformation of real-world coordinates. First, the *Intrinsic Matrix* is composed of the focal length (f_x & f_y), the optical center (x_0 & y_0), and the shear factor (s). These three parameters are a function of the geometry of the camera that captured the image and, therefore, remains constant in the dataset. The general intrinsic matrix is shown in Eq. (15). To make the matrix multiplication work with the *Extrinsic Matrix*, E, discussed next, a column-vector of *0*s is added to the end [67].

$$\text{Intrinsic Matrix:} \quad K = \underbrace{\begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{2-D Translation} \\ \text{Matrix}}} \times \underbrace{\begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{2-D Scaling} \\ \text{Matrix}}} \times \underbrace{\begin{bmatrix} 1 & \frac{s}{f_x} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\substack{\text{2-D Shear} \\ \text{Matrix}}}$$

$$= \left[\begin{array}{ccc|c} f_x & s & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right]_{3\times4} \quad \textbf{(15)}$$

Next, the *Extrinsic Matrix*, E, needs to be found to completely describe the camera matrix. This matrix describes how the real-world coordinates are transformed relative to the camera. It is composed of a 3×3 Rotation Matrix, R, (the direction of the real-world axis in camera coordinates) and a 3×1 translation column-vector, t, (the position of the real-world origin in camera coordinates) Eq. **(**16**)**. Since it describes the transformation of the 3-D coordinates from the camera's perspective, it is unique to every image. Often, this matrix is made square for future decomposition by adding a row of ***[0, 0, 0, 1]*** at the end of the matrix [52].

$$\text{Extrinsic Matrix:} \quad E = \underbrace{\left[\begin{array}{c|c} I & t \\ \hline 0 & 1 \end{array}\right]}_{\substack{\text{Real-World Origin Position in} \\ \text{Camera Coordinates}}} \times \underbrace{\left[\begin{array}{c|c} R & 0 \\ \hline 0 & 1 \end{array}\right]}_{\substack{\text{Real-World Axis Directions in} \\ \text{Camera Coordinates}}}$$

$$= \left[\begin{array}{ccc|c} R_{1,1} & R_{1,2} & R_{1,3} & t_1 \\ R_{2,1} & R_{2,2} & R_{2,3} & t_2 \\ R_{3,1} & R_{3,2} & R_{3,2} & t_3 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4\times4} \quad \textbf{(16)}$$

With the camera matrix now completely defined, the algorithm used to automatically project the 2-D images with detected damage onto the 3-D point cloud to locate the position of the defect can be described. For this process, it is assumed that an image has been passed through the defect detection algorithm (an example image is shown in Figure 4.9d) and the camera pose matrix has already been recovered using the process discussed in Section 2.2. This camera pose matrix, in Eq. **(**17**)**, describes the position and orientation of the camera in real-world coordinates and is composed of a column-vector of the camera position in real-world coordinates, *C*, and the rotational matrix of the camera's orientation in real-world coordinates, R_C. Again, a row of ***[0, 0, 0, 1]*** is added to make the matrix square [52]. This matrix is known and unique for each image.

$$\begin{bmatrix} R_C & C \\ 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} I & C \\ 0 & 1 \end{bmatrix}}_{\substack{\text{Translation Matrix in} \\ \text{Real−World Coordinates}}} \times \underbrace{\begin{bmatrix} R_C & 0 \\ 0 & 1 \end{bmatrix}}_{\substack{\text{Rotation Matrix in} \\ \text{Real−World Coordinates}}}$$

$$= \left[\begin{array}{ccc|c} R_{C1,1} & R_{C1,2} & R_{C1,3} & x \\ R_{C2,1} & R_{C2,2} & R_{C2,3} & y \\ R_{C3,1} & R_{C3,2} & R_{C3,2} & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]_{4\times4} \tag{17}$$

To find the extrinsic matrix given the camera pose matrix, take the inverse of the camera pose matrix and simplify Eq. **(**18**)**.

$$\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_C & C \\ 0 & 1 \end{bmatrix}^{-1} \tag{18}$$

$$= \left[\begin{bmatrix} I & C \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} R_C & 0 \\ 0 & 1 \end{bmatrix}\right]^{-1}$$

$$= \begin{bmatrix} R_C & 0 \\ 0 & 1 \end{bmatrix}^{-1} \times \begin{bmatrix} I & C \\ 0 & 1 \end{bmatrix}^{-0}$$

$$= \begin{bmatrix} R_C^T & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} I & -C \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} R_C^T & -R_C^T \cdot C \\ 0 & 1 \end{bmatrix}_{4\times4}$$

With Eq. **(**18**)**, the relationship between real-world coordinates and the camera coordinates is shown in Eq. **(**19**)** and Eq. **(**20**)**. The *Extrinsic Matrix* in real-world coordinates described by the camera pose matrix composed of R_C and C is shown in Eq. **(**21**)**.

$$R = R_C^T \tag{19}$$

$$t = -R \cdot C \tag{20}$$

$$\text{Extrinsic Matrix: } E = \underbrace{\begin{bmatrix} R_C^T & -R_C^T \cdot C \\ 0 & 1 \end{bmatrix}}_{\substack{\text{Extrinsic Matrix in} \\ \text{Real−World Coordinates}}} \tag{21}$$

Finally, combining the *Intrinsic Matrix* with the known camera parameters and *Extrinsic Matrix* in real-world coordinates, described by the camera pose matrix composed of R_C and C, the unique *Camera Matrix*, P, for each image can be assembled Eq. **(**22**)**.

$$[P]_{3\times4} = \begin{bmatrix} f_x & s & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{3\times4} \times \left[\begin{array}{c|c} R_C^T & -R_C^T \cdot C \\ \hline 0 & 1 \end{array}\right]_{4\times4} \tag{22}$$

Now, using the composed *Camera Matrix* Eq. **(**22**)**, P, with all values known from the camera's parameters and pose matrix, the 3-D point cloud in real-world coordinates (obtained from the point cloud discussed in Section 2.2) can be projected into 3-D pixel space by multiplying the column-vector of a given point by the *Camera Matrix* shown in Eq. **(**23**)**. $[x_1 \quad x_2 \quad x_3]^T$ is one 3-D point from the 3-D point cloud.

$$\underbrace{\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix}}_{\substack{\text{3−D Point in} \\ \text{Pixel Space}}} = [P]_{3\times4} \times \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix}}_{\substack{\text{3−D Point in} \\ \text{Real−World Coordinates}}} \tag{23}$$

The transformed 3-D point, the output from Eq. **(**23**)**, is now in 3-D pixel space. Next, the points are transformed again from 3-D pixel space into 2-D pixel space by Eq. **(**24**)** and give the x_1' and x_2' each point. x_1' and x_2' are the final results of the projection of the 3-D point cloud into 2-D points in pixel space.

$$\underbrace{\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}}_{\substack{\text{2−D Point in} \\ \text{Pixel Space}}} = \begin{bmatrix} {}^{x_1}/_{x_3} \\ {}^{x_2}/_{x_3} \end{bmatrix} \tag{24}$$

Using the technique previously mentioned, a 3-D point cloud in real-world coordinates can be readily mapped into 2-D pixel space. The projected 3-D point cloud has the same dimensions as the original image (i.e., if the image is 3,000×4,000 pixels, the projected point cloud will be 3,000×4,000 units); therefore, the points on the projected 2-D point cloud will closely match with the pixels in the original image. Two example output images are shown in Figure 4.9. Figure 4.9a is the original image captured by the UAV during flight. Using the technique from above and knowing the camera's pose and parameters composed of R_C and C, Figure 4.9b is the projected point cloud in 2-D pixel space.

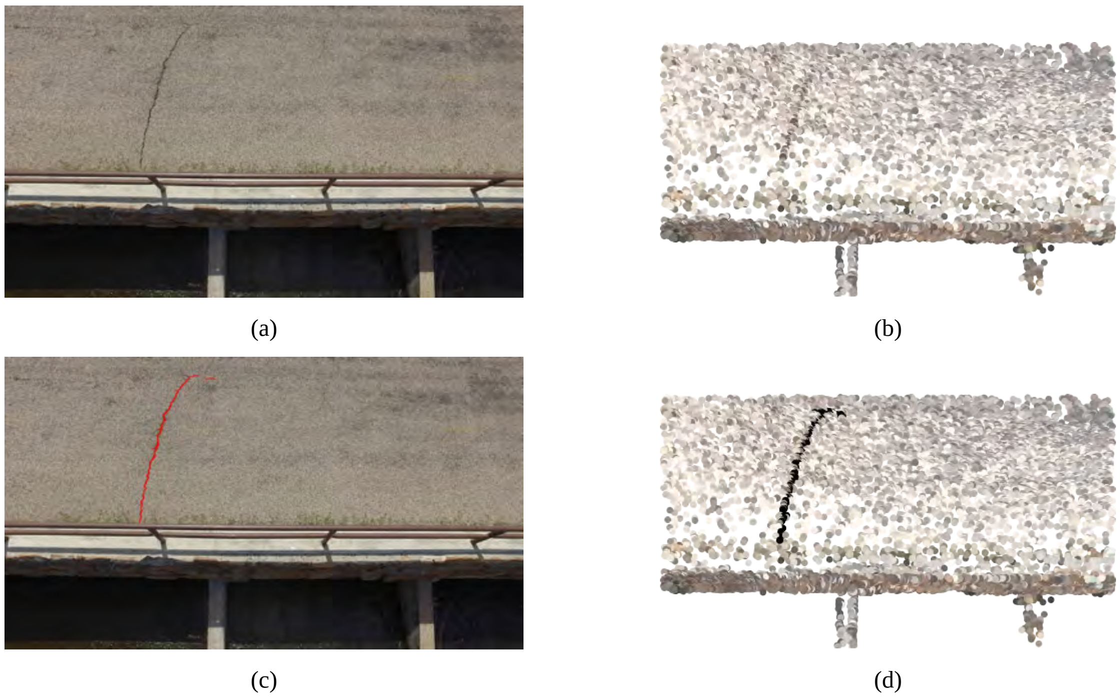


Figure 4.9 Sample images of the projection of the point cloud using the camera pose matrix composed of R_C and R: (a) image taken by the UAV, (b) projection of the point cloud in 2-D pixel space using the camera's pose matrix and parameters, (c) detected damage from the image using the algorithm discussed in Section 4.1, and (d) detected defects mapped onto the points

A 2-D pixel from an image can be readily mapped to the corresponding point and related to the point cloud in 3-D real-world coordinates. The detected defect(s) from the algorithm (discussed in Section 4.1 and an example shown in Figure 4.9c) are located in this 2-D pixel space with red pixels/markers. Using Euclidean distance, the closest point from the point cloud can be matched with the red pixels/markers to identify the defected points. Before the transformation, each point is indexed. Once the defected points of the point cloud are identified, the index number is related to the non-transformed points in the 3-D real-world coordinates. Thus, the 3-D position of the defect is known. An example of the full process is shown in Figure 4.9. Figure 4.9a is the original image taken by the UAV. The 3-D points are then projected into 2-D pixel space (Figure 4.9b). After the image is passed through the defect detection algorithm, discussed in Section 4.1 (Figure 4.9c), the defects are readily mapped onto the projected points shown in Figure 4.9d. Finally, the projected points are mapped back to the 3-D point cloud, and location and size are known in real-world coordinates. The final projected defects are shown in Figure 4.10. The red cube is the position and orientation of the camera at the time of the image (i.e., the camera's pose).

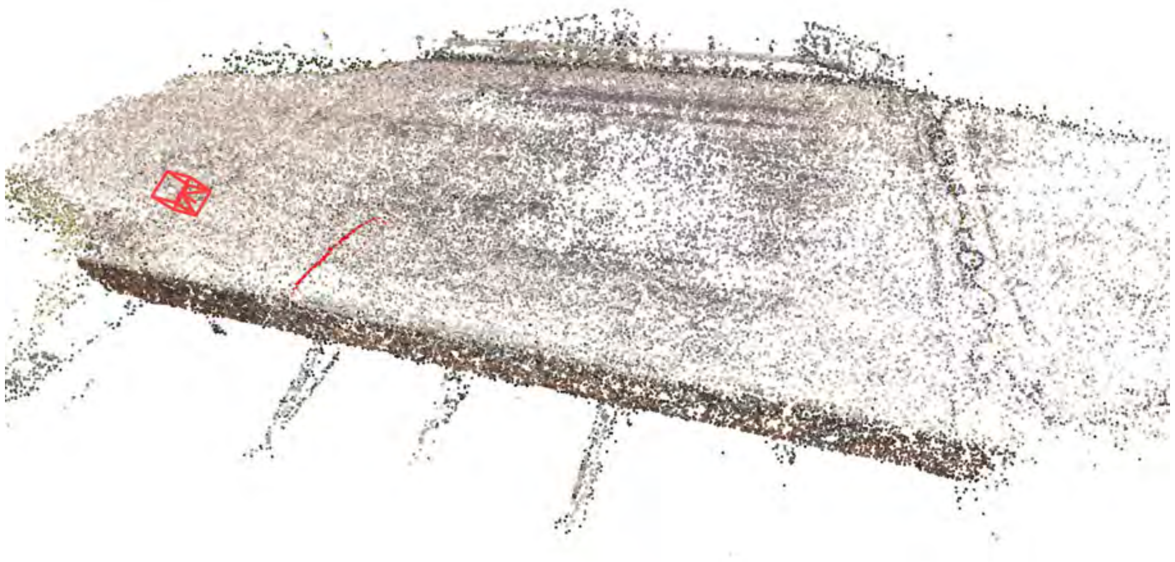


Figure 4.10 3-D point cloud with the detected defect mapped in real-world coordinates

The proposed technique takes about 10 seconds for each image to map onto the 3-D point cloud. The most computational involved portion of the algorithm is finding the closest points from the 2-D image to the projected 2-D points. This matching portion is currently done by finding a projected point with the shortest Euclidean distance to the 2-D red pixel/marker in the image. It was found that the first method using the texture image mapped the defects in 3-D real-world coordinates quickly and easily but missed the majority of the defects when passed through the defect detection algorithm; however, the technique based on camera pose successfully projected the defects located in 2-D pixel space into 3-D real-world coordinates.

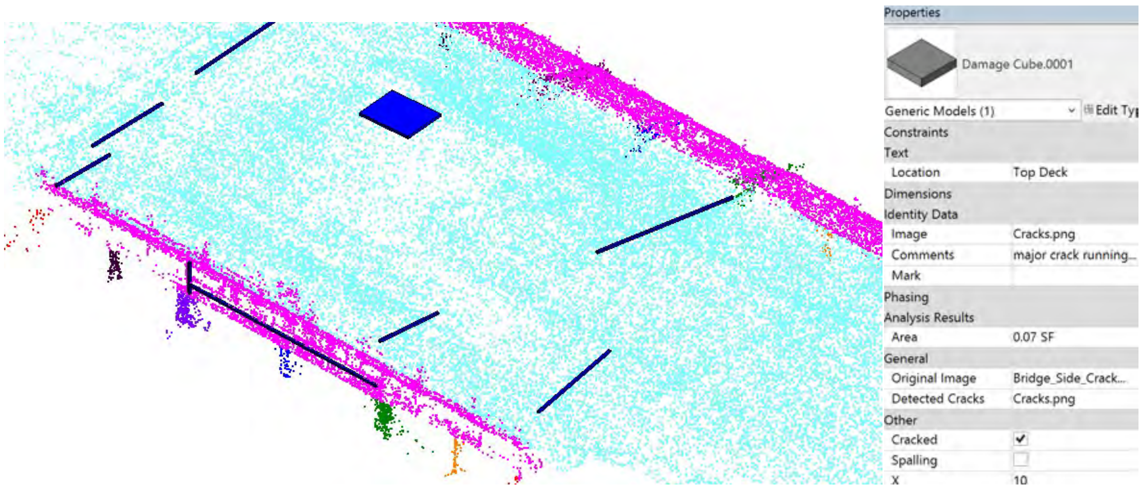
Using this technique, the identified defects in the images are automatically mapped onto the point cloud and located in 3-D space. This allows a bridge manager to easily visualize the location and size of all the identified defects on the structure. Furthermore, this quantified localization and size information of damage enables the element-wise bridge condition assessment through AB-BrIM modeling, which will be discussed in Section 5.

5. DEVELOPMENT OF AS-BUILT BRIDGE INFORMATION MODEL (AB-BRIM)

An AB-BrIM is developed based on the point cloud to document the quantified and localized damage information for structural elements in a 3-D model. To build the AB-BrIM, a base software program had to be chosen. For this particular application, a robust BIM package that could handle point cloud data and input customizable, user-defined parameters/objects was needed. Three BIM software suites were considered: LEAP Bridge, Tekla Structures, and Revit [68–70]. Based on a comparison study by McGuire et al., the Revit® software suite, designed by Autodesk, Inc., was chosen for this need, as it was able to associate any user-defined information (i.e., damage information, images, etc.) to any specific element or object, which can then be easily queried and accessed [71]. Note that commercial software was only used to assist the authors to store and visualize the AB-BrIM; however, it is not required for the implementation of the proposed system, as a proprietary or openly sourced program for BrIM could be easily built. To easily build, access, and query the user-defined information, a "family" within Revit was created. Additionally, the Revit® software can automatically generate reports or schedules of these user-defined families. For instance, with a few clicks in the user interface, all the damage information for each element (i.e., girder, deck, guardrail, etc.) can be listed and quantified in a formatted schedule. This automatic report creation ability combined with the use of Revit families to store and locate information is convenient for a bridge owner or manager to quickly quantify and report all the damage on the bridge or on a particular element of the structure.

5.1. Automatic Information Upload and Model Establishment in Revit

To begin the automatic information upload to Revit, the 3-D point cloud is scaled to real-world coordinates and georeferenced in the Revit project using ground-truth data from GPS points. The segmented elements (output of the process described in Section 3.4) were also uploaded and labeled in this process. This step of uploading, scaling, and geo-referencing can also be automated using the process described later in this section. Next, the concept of "damage cubes," proposed by McGuire et al., was adapted here for documentation of each detected defect [71]. A damage cube is a generic, user-defined Revit family object that can have a variably defined shape and size. The damage cube is placed on an element where damage was identified and contains any user-defined information for the specified location or defect; the proposed stored parameters are found in Table 5.1. The damage cubes are represented by the blue objects in Figure 5.1, where their size, shape, and location are consistent with the actual damage area, pattern, and location in the structure. Each point cluster in Figure 5.1 of the same color is the segmented elements of the bridge (i.e., cyan is bridge deck, magenta is sides/guardrails, red is wing walls, etc.).



Properties

Damage Cube.0001

Generic Models (1) | Edit Typ

Constraints	
Text	
Location	Top Deck
Dimensions	
Identity Data	
Image	Cracks.png
Comments	major crack running...
Mark	
Phasing	
Analysis Results	
Area	0.07 SF
General	
Original Image	Bridge_Side_Crack...
Detected Cracks	Cracks.png
Other	
Cracked	✓
Spalling	☐
X	10

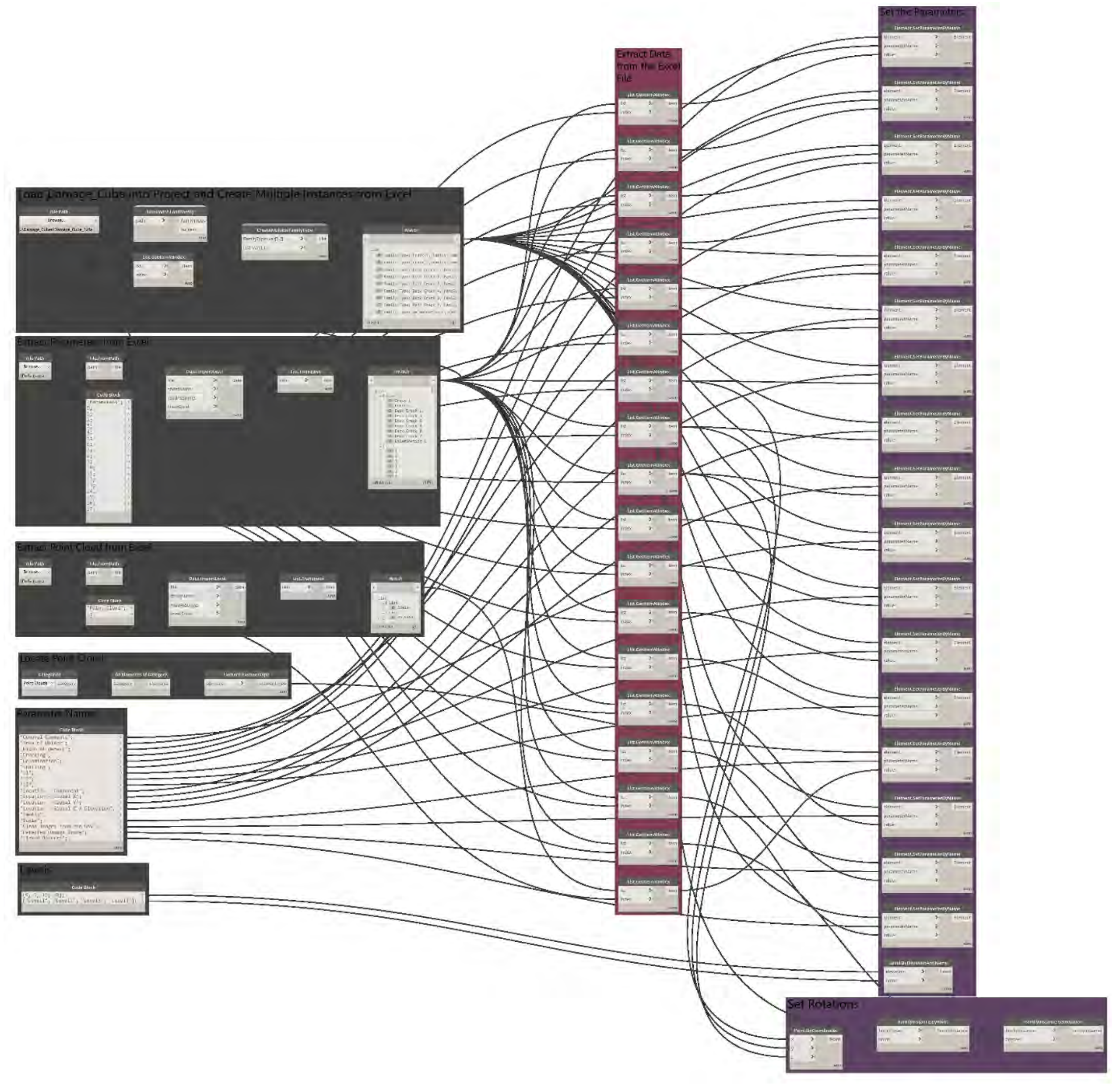
(a) (b)

Figure 5.1 Example damage cubes (blue objects) placed on the segmented point cloud (cyan is bridge deck, magenta is side/guardrail, red is wing wall, etc.): (a) AB-BrIM overview, and (b) property menu of selected damage cube

Table 5.1 Proposed and example parameters for the properties of a damage cube

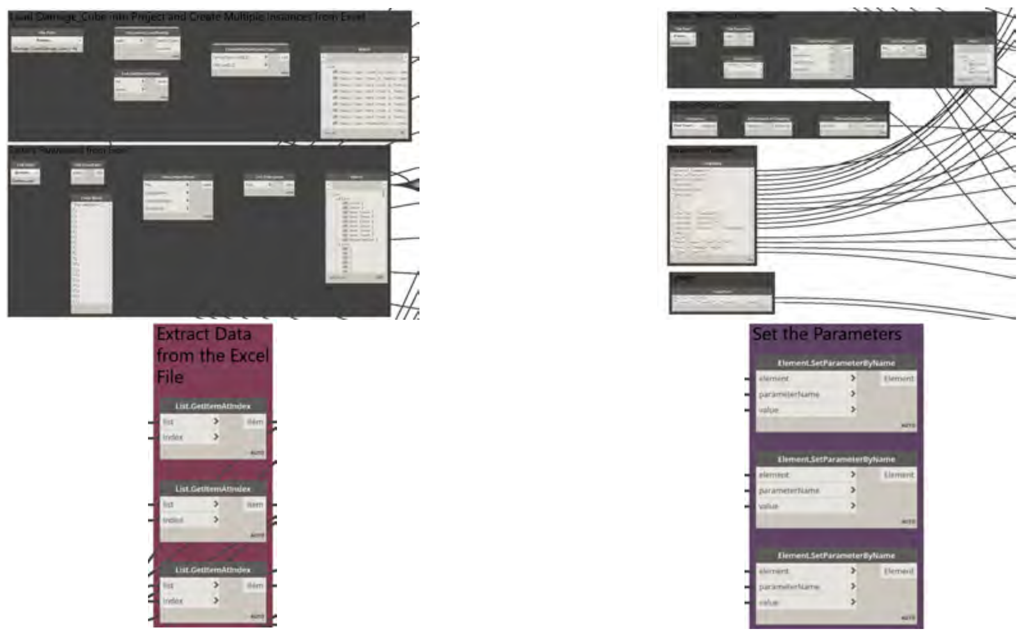
Proposed Damage Cube Parameters	Example Values
General Comments	"Suspected Delamination"
Area of Defect (*inches2*)	1728
Width of Defect (*inches*)	N/A
Cracking	Unchecked Checkbox
Delamination	Checked Checkbox
Spalling	Unchecked Checkbox
Dimension for Visualization	5-feet × 4-feet × *2*-inches
Location – Component	"Deck"
Location – Global	(-2.5091, 15.2083, 0.5000)
Original Image from UAV	"C:\0716.JPEG"
Damage Detected Image	"C:\07166_DD.JPEG"
Linked Report	"C:\FCPALM_201506.PDF

With the damage cube established and created to store localized information about a particular defect and to easily visualize where the damage exists on a structure, the damage information and any other information about the defect is automatically uploaded into Revit and associated with a damage cube. Revit uses Dynamo to allow for the automation of commands and functions. Dynamo is a node-based, visual programming language with each node representing either a variable or a function. The nodes, which can act as an input or output, are connected to create an automated program for model establishment. Using Dynamo, a script can be created to automatically create and place the damage cubes onto the segmented bridge elements. Assuming the damage location and size, which are found using the algorithm discussed in Section 4.1, and/or any other information that is relevant to a specific damage cube (i.e., inspector's comments, additional images, etc.) is stored in a .CSV file, Dynamo can read this information and create a custom damage cubes for each damage location. The created Dynamo visual program is shown in Figure 5.2 with each section zoomed into and discussed in Figure 5.3.



Set the Parameters
Extract Data from the Excel File
Load Damage_Cubes into Project and Create Multiple Instances from Excel
Extract Parameters from Excel
Extract Point Cloud from Excel
Locate Point Cloud
Parameter Names
Levels
Set Rotations

Figure 5.2 Full dynamo program to automatically create the damage cubes



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

Figure 5.3 Node grouping: (a) nodes for loading in damage cube family and .CSV file with each damage cube instance information, (b) nodes for loading in the point cloud and some constants used, (c) nodes for extracting the information from the .CSV file, and (d) nodes for setting the parameters of the damage cube instance

After the point cloud is uploaded, scaled, and georeferenced in the Revit project, the damage cubes can be automatically placed on the segmented elements with the stored damage information using Dynamo and the program described in Figure 5.2 and Figure 5.3. First, the damage cube family and the .CSV file are loaded in the project. Dynamo creates a generic damage cube for each instance in the file. Next, Dynamo extracts all other information in the file associated with each damage cube instance and stores it in a list. With each generic damage cube instance created, the information in the extracted list from the .CSV file can be set as the damage cube's parameters. Once the parameters of the damage cube are set, the size, shape, location, and stored information of the damage cube is updated; finally, Dynamo draws each damage cube with the assigned information to create Figure 5.1.

Changes made in Revit to the point cloud, damage cube, or information in the damage cube, can automatically be updated in the .CSV file for later use. Additionally, since the parameters are known for scaling and georeferencing the point cloud, they can readily be stored in the .CSV file, which would further promote automated AB-BrIM generation. Having all the information stored in a .CSV file could also condense the file size; instead of saving the entire Revit file, the information could be stored separately and the Revit file could be compiled and built when opening.

With the final AB-BrIM assembled and usable, file size and storage become a concern. Table 5.2 shows the file required to run and build the AB-BrIM. Under current inspection practices, only a PDF report is stored with a size of about 1,200 KB. With the proposed element-wise AB-BrIM, significantly more disk space is required for storage; the majority of the disk space is the large number of high-resolution images. However, not all the images are required to be saved to view the model and can be kept for documentation purposes; therefore, keeping 20% or less will still provide more imagery and data than when stored in the current PDF reports. Furthermore, the full resolution of the images is not required after the point cloud generation and damage detection steps of the framework are complete. The images can be compressed after these two steps to save on storage. For example, reducing the image resolution by 50% reduces the image storage requirement by 87%. Lastly, having more imagery, data, and a robust AB-BrIM that bridge owners and managers can use justifies the additional storage expense.

Table 5.2 Storage sizes for AB-BrIM using Revit

	Storage Size	
Assembled Revit Model	7,500-*KB*	7.50-*MB*
.CSV File	25-*KB*	0.0250-*MB*
Point Cloud Model	7,000-*KB*	7.00-*MB*
Photorealistic Model	490,000-*KB*	490-*MB*
Full Palmer Bridge Image Set (449 JPEG Images)	2,160,000-*KB*	2,160-*MB*
Palmer Bridge Image Subset (82 JPEG Images)	424,000-*KB*	424-*MB*
Palmer Bridge 50% Reduced Image Subset (82 JPEG Images)	55,000-*KB*	55-*MB*
Assemble to View Process	62,025-*KB*	62.025-*MB*
Self-Contained Revit File	62,500-*KB*	62.50-*MB*
Current Report	1,200-*KB*	1.2-*MB*

Using this AB-BrIM, visualization of the overall condition of the structure is more straightforward. The size and pattern of the damage cubes displayed in the 3-D model highlight the important areas where attention is needed (Figure 5.1a). A user can click on a damage cube to check detailed damage information in a properties box (Figure 5.1b); an example of all the proposed parameters within the damage cube is shown in Table 5.1. Such realized damage visualization can help bridge managers make more informed decisions in bridge management. The automated damage modeling program allows efficient model updating when new inspection data become available.

6. DISCUSSION OF RESULTS

To evaluate the efficiency and advantage of the proposed bridge inspection system, it was compared with two existing techniques. The first one is the traditional human-based inspection and represents the current state of practice in routine bridge inspection. The second one is the UAV-assisted inspection without automated image processing, which reflects the recent exploration of potential ways of using UAVs. The advantages and shortcomings of each approach will be discussed in the subsequent subsections. The results are summarized in Table 6.1.

Table 6.1 Comparison of bridge inspection techniques

	Traditional Human-Based Techniques	UAV-Assisted Inspection with Manual Defect Detection from Images (Second Approach)	Proposed Automated Bridge Inspection System Implementation
On-Site Time	2 weeks	2 hours	2 hours
Equipment	"Snooper Trucks," and/or Scaffolding	UAV Platform	UAV Platform
In-Office Time	Minimal	Significant	Automated
Deliverable	PDF Report	AB-BrIM	AB-BrIM
Damage Information	Manually and Qualitatively Documented	Manually Identified and Measured from Images	Automatically Detected
Safety	Special Equipment; Adjacent to Traffic	Outside of Traffic	Outside of Traffic
Cost[3]	\$55,000+	\$20,000+	\$20,000

[3] Based on a contract estimation by [7]

6.1. Traditional Human-Based Technique

In this approach, an inspector and crew use visual inspection to manually identify and document cracks and other defects. Apart from the shortcomings discussed in Section 1.1, such as subjectivity, safety concerns, low efficiency, and signification cost, these conventional techniques have an additional major drawback, i.e., the tracking and visualization of the damage growth/structural condition change in the future are not convenient. The inspection report typically consists of a table with numerated information and some appended images of the surrounding areas and elements. Occasionally, markings or other notes will be recorded on the structure to assist future inspectors to evaluate the progression of existing cracks and other defects over time. However, a quantitative measure of the rate of defect growth over time is not feasible using this limited information. With only a couple of images collected for the structure, it is often difficult to visualize the severity and location of the defects or to compare it with a previous condition. Additionally, different crews may perform future inspections, leading to the possibility of inconsistencies and misinterpretations of the structure's changing condition.

6.2. UAV-Assisted Inspections: Current Practice

There are two existing techniques to implement UAVs to assist bridge inspectors in current practice. The first way is to use UAV to provide "eyes-in-the-sky." The second is to use UAVs to collect the images and then manually process the images and identify damage. Each of these two approaches will be discussed in this subsection.

In the first method, UAVs are used to perform pre-survey and/or enhance data collection during a bridge inspection. A UAV with optical sensors will fly around the structure collecting pictures or be used as a real-time surveillance system of the structure. Using the images, the on-site inspectors can visualize the most important areas of interest and have insight on where and how to perform a more in-depth inspection. Additionally, the images could be included in the report and provide more data to be used in future decision making. This approach will assist the inspectors to better utilize the on-site time by focusing on the important areas identified through UAV surveillance and provide more data; however, it does not necessarily save on-site time when compared with the traditional techniques, nor does it take full advantage of the current technology in image processing.

The second approach is to conduct the damage detection, quantification, and mapping manually from the images taken from the UAV. High-resolution images will be used to generate a 3-D point cloud and photorealistic model. An inspector will review the images and photorealistic model to identify structural damage. Because of the high overlap of the images, the inspector is not required to review all the images to identify the defects. Once a defect is identified, it is marked and tagged to the 3-D point cloud and photorealistic model. Any other relevant information can be associated with this damaged spot by manually establishing AB-BrIM and a damage cube. The inspector and crew could spend significantly less time on the job site, for example, reducing the time from a week or more to a couple of hours. Traffic control may still be required for certain areas, but the duration of this control is significantly reduced. With this approach, substantially more time is needed for the final report and data processing when compared with the traditional human-based approach. This report creation time could increase from a day to multiple days in the office. However, the final deliverable contains more damage information to facilitate better condition assessments as well as more complete information for future inspectors to consider when measuring the change of the structure over time.

6.3. Integration of UAV-based Inspection with Computer Vision and Machine Learning Techniques: Proposed System

Compared with current UAV-assisted inspection practices, where manual data processing is necessary, the proposed system in this study takes full advantage of the advancements in UAVs as well as computer vision and machine learning techniques to significantly improve the efficiency and quality of data processing/interpretation. Using this new system, the UAV collects data around the structure; then, image processing and machine learning algorithms developed in this study are used to build a point cloud and photorealistic base for an AB-BrIM, detect, track, and quantify defects from the images, identify the structural elements, and map the defects information on to the element-wise 3-D AB-BrIM. In the entire data processing procedure, only minimal user inputs are required in the element identification module, while all the other modules are fully automated. Therefore, the data processing burden on the inspector and crew resulting from the big data collected by UAV is greatly reduced.

Once the high-resolution images are captured, the proposed data processing procedure will be applied. Although computing time is increased when compared with human-based or existing UAV-assisted techniques, the algorithms can run autonomously or be left overnight, thus drastically reducing report generation time for the inspector. The inspector will also perform a quality control assessment afterward to ensure the proper performance of the algorithms. Once all this information is processed and mapped, future inspectors and decision-makers can then revisit the 3-D point cloud model and visualize the information more cohesively. Also, successive inspections become more comparable with the additional photos and quantified damage information. Although traffic control is still required when using the proposed inspection system, the time of the traffic interruption, as compared with conventional techniques, will be reduced from days/weeks to a couple of hours. The proposed inspection procedure requires less time in the field, thus increasing the number of bridges that an inspector and crew can inspect in a season while providing more quantified information to be used for more robust decision-making.

7. CONCLUDING REMARKS AND FUTURE WORK

Using the current human-based techniques outlined by AASHTO and FHWA, in-depth inspection of large-scale bridges is relatively slow, costly, and potentially dangerous. Additionally, the extent and severity of the defects are not rigorously delineated on a rating scale, thus the condition assessment results may be subjective and inconsistent. Once the final reports are created, it is often difficult to know exactly where each defect is located and to track the growth of the identified defects over time. To overcome these challenges, present in current practice, an automated bridge inspection system is proposed by leveraging the most recent advances in UAV technologies, computer vision, and machine learning techniques. Implementation of the proposed system has been streamlined with element-wise damage information that is consistent with typical practices and a geo-referenced AB-BrIM to fit seamlessly into state DOTs' GIS-enabled assets management systems.

The comparison of the proposed system with the human-based and the existing UAV-assisted bridge inspection techniques highlights the unique advantages of the new system in terms of in-depth data analytics, which is accomplished through the development of automated algorithms for damage quantification and visualization. The proposed automated defect detection, defect growth tracking, and defect location mapping algorithms, in conjunction with the element identification algorithm, identify, locate, and size defects in the structure and map this information to an element-wise AB-BrIM for easy visualization. This 3-D visualization and damage documentation will enable bridge managers to plan for future maintenance and rehabilitation projects more efficiently. Also note that although the damage identification module was demonstrated only on concrete surface cracks, the proposed framework can be readily used to quantify other types of damage, such as section loss, spalling, and delamination, using the geometric information of the point clouds and the visual information of the images. The quantitative inspection achieved using this framework can provide important structural condition information (such as as-built geometry of structural elements, damage identification, concrete section loss, etc.) that is essential for accurate structural analysis and load rating [72]. This information is valuable, as it would otherwise not be available from current human visual inspection techniques.

7.1. Future Study

Future work of this project would continue to promote more robust inspection techniques by leveraging the advancements of optical sensors and additional payloads available. A module for analyzing thermal imagery to detect subsurface delamination of the bridge deck can be readily incorporated in the current framework. In addition, the integration of autonomous UAV flights with elevation changes will be explored. Autonomous flight of the UAV will promote more consistent data collection and enable more frequent and repeated inspections. Additionally, an autonomous flight would be more efficient for data collection by allowing for the optimal overlap of the images and by calculating the most efficient path of the UAV. Automatic tracking of structure changes over time becomes much easier with similar image collection. This information is valuable for conducting life-cycle analysis or deterioration modeling as defects that change over time are often handled differently than defects that remain consistent. Finally, as LiDAR systems become more affordable, they can be incorporated to allow for real-time data processing.

REFERENCES

[1] T.W. Ryan, E. Mann, Z.M. & Chill, B.T. Ott, *Bridge Inspector's Reference Manual*, Washington, D.C., 2012. www.nhi.fhwa.dot.gov.

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

[3] T. Omar, & M.L. Nehdi, "Remote sensing of concrete bridge decks using unmanned aerial vehicle infrared thermography," *Autom. Constr.* 83 (2017) 360–371. https://doi.org/10.1016/j.autcon.2017.06.024.

[4] M. Moore, B. Phares, B. Graybeal, D. Rolander, & G. Washer, "Reliability of Visual Inspection for Highway Bridges, Volume I: Final Report," McLean, VA, 2001. http://www.tfhrc.gov/hnr20/nde/01020.htm.

[5] S. Spray, "Rope Access Surveys," *Stuart Spray Wildl. Consult.* (2015). http://www.stuartspraywildlife.co.uk/rope-access-surveys/ (accessed November 10, 2018).

[6] Branson, "Inspectors Snoop Under Bridges to Check Conditions," Missouri Dep. Transp. (2014). http://modotblog.blogspot.com/2014/06/inspectors-snoop-under-bridges-to-check.html (accessed November 10, 2018).

[7] J. Wells, & B. Lovelace, "Unmanned Aircraft System Bridge Inspection Demonstration Project Phase II Final Report," St. Paul, MN, 2017.

[8] I. Hernandez, T. Fields, & J. Kevern, "Overcoming the Challenges of Using Unmanned Aircraft for Bridge Inspections," in: *AIAA Atmos. Flight Mech. Conf., American Institute of Aeronautics and Astronautics*, Reston, Virginia, 2016. https://doi.org/10.2514/6.2016-3396.

[9] D.T. Gillins, C. Parrish, M.N. Gillins, & C. Simpson, "Eyes in the Sky: Bridge Inspections With Unmanned Aerial Vehicles," Salem, OR, 2018. https://www.oregon.gov/ODOT/Programs/Pages/Research-Publications.aspx.

[10] J. Wells, & B. Lovelace, "Improving the Quality of Bridge Inspections Using Unmanned Aircraft Systems (UAS)," St. Paul, MN, 2018. http://www.dot.state.mn.us/research/reports/2018/201826.pdf.

[11] L. Duque, J. Seo, & J. Wacker, "Bridge Deterioration Quantification Protocol Using UAV," *J. Bridg. Eng.* 23 (2018) 04018080. https://doi.org/10.1061/(ASCE)BE.1943-5592.0001289.

[12] Y. Xu, Y. & Turkan, "BrIM and UAS for bridge inspections and management," *Eng. Constr. Archit. Manag.* 27 (2019) 785–807. https://doi.org/10.1108/ECAM-12-2018-0556.

[13] X.W. Ye, Y.Q. Ni, T.T. Wai, K.Y. Wong, X.M. Zhang, & F. Xu, "A vision-based system for dynamic displacement measurement of long-span bridges: algorithm and verification," *Smart Struct. Syst.* 12 (2013) 363–379. https://doi.org/10.12989/sss.2013.12.3_4.363.

[14] X.W. Ye, C.Z. Dong, & T. Liu, "Image-based structural dynamic displacement measurement using different multi-object tracking algorithms," *Smart Struct. Syst.* 17 (2016) 935–956. https://doi.org/10.12989/sss.2016.17.6.935.

[15] X.W. Ye, T.-H. Yi, C.Z. Dong, & T. Liu, "Vision-based structural displacement measurement: System performance evaluation and influence factor analysis," *Measurement.* 88 (2016) 372–384. https://doi.org/10.1016/j.measurement.2016.01.024.

[16] X.W. Ye, T.-H. Yi, C.Z. Dong, T. Liu, & H. Bai, "Multi-point displacement monitoring of bridges using a vision-based approach," *Wind Struct.* 20 (2015) 315–326. https://doi.org/10.12989/was.2015.20.2.315.

[17] M.R. Jahanshahi, & S.F. Masri, "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures," *Autom. Constr.* 22 (2012) 567–576. https://doi.org/10.1016/j.autcon.2011.11.018.

[18] A.M.A. Talab, Z. Huang, F. Xi, & L. HaiMing, "Detection crack in image using Otsu method and multiple filtering in image processing techniques," *Optik (Stuttg).* 127 (2016) 1030–1033. https://doi.org/10.1016/j.ijleo.2015.09.147.

[19] A. Reddy, V. Indragandhi, L. Ravi, & V. Subramaniyaswamy, "Detection of Cracks and damage in wind turbine blades using artificial intelligence-based image analytics," *Measurement.* (2019). https://doi.org/10.1016/j.measurement.2019.07.051.

[20] N.J. Mitra, & A. Nguyen, "Estimating Surface Normals in Noisy Point Cloud Data," in: *Symp. Comput. Geom.*, SoCG 2003, San Diego, CA, 2003: pp. 322–328.

[21] R. Lu, I. Brilakis, & C.R. Middleton, "Detection of Structural Components in Point Clouds of Existing RC Bridges," *Comput. Civ. Infrastruct. Eng*. 34 (2019) 191–212. https://doi.org/10.1111/mice.12407.

[22] T. Czerniawski, B. Sankaran, M. Nahangi, C. Haas, & F. Leite, "6D DBSCAN-based segmentation of building point clouds for planar object classification," *Autom. Constr*. 88 (2018) 44–58. https://doi.org/10.1016/j.autcon.2017.12.029.

[23] H. Macher, T. Landes, & P. Grussenmeyer, "From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings," *Appl. Sci. 7* (2017) 1030. https://doi.org/10.3390/app7101030.

[24] R. Maalek, D. Lichti, & J. Ruwanpura, "Robust Segmentation of Planar and Linear Features of Terrestrial Laser Scanner Point Clouds Acquired from Construction Sites," *Sensors.* 18 (2018) 819. https://doi.org/10.3390/s18030819.

[25] L. Díaz-Vilariño, H. González-Jorge, J. Martínez-Sánchez, & H. Lorenzo, "Automatic LiDAR-based lighting inventory in buildings," *Meas. J. Int. Meas. Confed.* (2015). https://doi.org/10.1016/j.measurement.2015.06.009.

[26] J. Mulakala, "Measurement Accuracy of the DJI Phantom 4 RTK & Photogrammetry," San Francisco, CA, 2018. https://docs.djicdn.com/DJI+Enterprise/measurement-accuracy-dji-phantom-4-rtk-whitepaper-f.pdf.

[27] J. Seo, L. Duque, & J.P. Wacker, "Field Application of UAS-Based Bridge Inspection," *Transp. Res. Rec.* 2672 (2018) 72–81. https://doi.org/10.1177/0361198118780825.

[28] S. Bang, H. Kim, & H. Kim, "UAV-based automatic generation of high-resolution panorama at a construction site with a focus on preprocessing for image stitching," *Autom. Constr.* 84 (2017) 70–80. https://doi.org/10.1016/j.autcon.2017.08.031.

[29] W.W. Greenwood, J.P. Lynch, & D. Zekkos, Applications of UAVs in Civil Infrastructure, J. Infrastruct. Syst. 25 (2019) 04019002. https://doi.org/10.1061/(asce)is.1943-555x.0000464.

[30] C. Brooks, R. Dobson, D. Banach, T. Oommen, K. Zhang, A. Mukherjee, T. Havens, T. Ahlborn, R. Escobar-Wolf, C. Bhat, S. Zhao, Q. Lyu, & N. Marion, "Implementation of Unmanned Aerial Vehicles (UAVs) for Assessment of Transportation Infrastructure Phase II," Houghton, MI, 2018.

[31] S. Dorafshan, M. Maguire, N. V. Hoffer, & C. Coopmans, "Challenges in bridge inspection using small unmanned aerial systems: Results and lessons learned," in: 2017 Int. Conf. Unmanned Aircr. Syst., IEEE, 2017: pp. 1722–1730. https://doi.org/10.1109/ICUAS.2017.7991459.

[32] B.\& H.P. and Video, Mavic, (2019). https://static.bhphoto.com/images/images1000x1000/1535023310_1430448.jpg (accessed April 11, 2019).

[33] B.\& H.P. and Video, Phantom, (2019). https://static.bhphoto.com/images/images1000x1000/1479219684_1298124.jpg (accessed April 11, 2019).

[34] B \& H Photo and Video, Matrice, (2019). https://static.bhphoto.com/images/images1000x1000/1479120343_1297339.jpg (accessed April 11, 2019).

[35] P. Office, "FAA Drone Registry Tops One Million," Transportation. Gov. (2018). https://www.transportation.gov/briefing-room/faa-drone-registry-tops-one-million (accessed November 10, 2018).

[36] W.K. Chiu, W.H. Ong, T. Kuen, & F. Courtney, "Large Structures Monitoring Using Unmanned Aerial Vehicles," *Procedia Eng. 188* (2017) 415–423. https://doi.org/10.1016/j.proeng.2017.04.503.

[37] S. Shahnaz, "Gravel Road Condition Monitoring Using Unmanned Aerial Vehicle (UAV) Technology," South Dakota State University, 2010.

[38] PrecisionHawk, Precision Flight, (2018).

[39] Skycatch, Flights, (2018). https://www.skycatch.com/products/.

[40] Pix4D, Pix4D Capture, Pix4D. (2019). https://www.pix4d.com/product/pix4dcapture.

[41] American Society of Civil Engineering, "2017 Infrastructure Report Card, A Comprehensive Assessment of America's Infrastructure," ASCE, Reston, VA, 2018. https://www.infrastructurereportcard.org/.

[42] M.J. Lato, D. Gauthier, & D.J. Hutchinson, "Selecting the optimal 3D remote sensing technology for the mapping, monitoring and management of steep rock slopes along transportation corridors," in: Transp. Res. Board 2015 Annu. Meet., TRB 2015 Annual Meeting, Washington D.C., 2015: pp. 1–15.

[43] D. Roca, J. Armesto, S. Lagüela, & L. Díaz-Vilariño, "Lidar-equipped UAV for building information modelling," *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* XL–5 (2014) 523–527. https://doi.org/10.5194/isprsarchives-XL-5-523-2014.

[44] A.J. Cawood, C.E. Bond, J.A. Howell, R.W.H. Butler, & Y. Totake, "LiDAR, UAV or compass-clinometer? Accuracy, coverage and the effects on structural models," *J. Struct. Geol.* 98 (2017) 67–82. https://doi.org/10.1016/j.jsg.2017.04.004.

[45] G.Y. Jeong, T.N. Nguyen, D.K. Tran, & T.B.H. Hoang, "Applying unmanned aerial vehicle photogrammetry for measuring dimension of structural elements in traditional timber building," *Measurement.* 153 (2020) 107386. https://doi.org/10.1016/j.measurement.2019.107386.

[46] F. Carvajal-Ramírez, A.D. Navarro-Ortega, F. Agüera-Vega, P. Martínez-Carricondo, & F. Mancini, "Virtual Reconstruction of Damaged Archaeological Sites based on Unmanned Aerial Vehicle Photogrammetry and 3D Modelling. Study Case of a Southeastern Iberia Production Area

in the Bronze Age," *Measurement.* 136 (2019) 225–236. https://doi.org/10.1016/j.measurement.2018.12.092.

[47] L. Díaz-Vilariño, H. González-Jorge, J. Martínez-Sánchez, M. Bueno, & P. Arias, "Determining the limits of unmanned aerial photogrammetry for the evaluation of road runoff," *Meas. J. Int. Meas. Confed*. (2016). https://doi.org/10.1016/j.measurement.2016.02.030.

[48] M. Jancosek, & T. Pajdla, "Multi-view reconstruction preserving weakly-supported surfaces," in: CVPR 2011, IEEE, 2011: pp. 3121–3128. https://doi.org/10.1109/CVPR.2011.5995693.

[49] P. Moulon, P. Monasse, & R. Marlet, "Adaptive Structure from Motion with a Contrario Model Estimation," in: K.M. Lee, Y. Matsushita, J.M. Rehg, Z. Hu (Eds.), Asian Conf. Comput. Vis. 2012, ACCV 2012, Daejeon Korea, 2013: pp. 257–270. https://doi.org/10.1007/978-3-642-37447-0_20.

[50] D.G. Lowe, "Object recognition from local scale-invariant features," in: Proc. Seventh IEEE Int. Conf. Comput. Vis., IEEE, Kerkyra, Greece, 1999: pp. 1150–1157 vol. 2. https://doi.org/10.1109/ICCV.1999.790410.

[51] M. Muja & D.G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in: VISAPP 2009 - Proc. 4th Int. Conf. Comput. Vis. Theory Appl., 2009: pp. 331–340.

[52] K. Simek & M. Reid, "Dissection the Camera Matrix," Sightations A Comput. Vis. Blog. (2013). http://ksimek.github.io/2013/08/13/intrinsic/ (accessed March 20, 2019).

[53] C. Wu, "Towards Linear-Time Incremental Structure from Motion," in: 2013 Int. Conf. 3D Vis., IEEE, 2013: pp. 127–134. https://doi.org/10.1109/3DV.2013.25.

[54] J.L. Schonberger & J.-M. Frahm, "Structure-from-Motion Revisited," in: 2016 IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, Las Vegas, NV, 2016: pp. 4104–4113. https://doi.org/10.1109/CVPR.2016.445.

[55] M. Kazhdan & H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph*. 32 (2013) 1–13. https://doi.org/10.1145/2487228.2487237.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhfer, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, & E. Duchesnay, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.* 12 (2011) 2825–2830. https://doi.org/10.1007/s13398-014-0173-7.2.

[57] S. Sankarasrinivasan, E. Balasubramanian, K. Karthik, U. Chandrasekar, & R. Gupta, "Health Monitoring of Civil Structures with Integrated UAV and Image Processing System," *Procedia Comput. Sci.* 54 (2015) 508–515. https://doi.org/10.1016/j.procs.2015.06.058.

[58] Q. Zhou, J. Park, & V. Koltun, "Open3D: A Modern Library for 3D Data Processing," *Open3D.* (2018). http://arxiv.org/abs/1801.09847.

[59] J. Li, H.-C. Wong, S.-L. Lo, & Y. Xin, "Multiple Object Detection by a Deformable Part-Based Model and an R-CNN," *IEEE Signal Process. Lett.* 25 (2018) 288–292. https://doi.org/10.1109/LSP.2017.2789325.

[60] S. Dorafshan, R.J. Thomas, & M. Maguire, "Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete," *Constr. Build. Mater.* 186 (2018) 1031–1045. https://doi.org/10.1016/j.conbuildmat.2018.08.011.

[61] H. Nhat-Duc, Q.-L. & Nguyen, V.-D. Tran, "Automatic recognition of asphalt pavement cracks using metaheuristic optimized edge detection algorithms and convolution neural network," *Autom. Constr.* 94 (2018) 203–213. https://doi.org/10.1016/j.autcon.2018.07.008.

[62] X.W. Ye, T. Jin, & C.B. Yun, "A review on deep learning - based structural health monitoring of civil infrastructures," *Smart Struct. Syst.* 24 (2019) 567–586. https://doi.org/10.12989/sss.2019.24.5.567.

[63] G. Bradski, "The OpenCV Library," Dr. Dobb's J. Softw. Tools. (2000). https://doi.org/2236121.

[64] A. Buades, B. Coll, & J.-M. Morel, "Non-Local Means Denoising," *Image Process. Line.* 1 (2011) 208–212. https://doi.org/10.5201/ipol.2011.bcm_nlm.

[65] J. Canny, "A Computational Approach To Edge Detection," *IEEE Trans. Pattern Anal. Mach. Intell.* 8 (1986) 679–714. https://ieeexplore-ieee-org.ezproxy2.library.colostate.edu/stamp/stamp.jsp?tp=&arnumber=4767851.

[66] ASTM International, "ASTM C496 Standard Test Method for Splitting Tensile Strength of Cylindrical Concrete Specimens," West Conshohocken, PA, 2017. https://doi.org/10.1520/C0496_C0496M-17.

[67] R. Hartley, & A. Zisserman, "Multiple View Geometry in Computer Vision," in: *Mult. View Geom. Comput. Vis., Second*, Cambridge University Press, 2004. http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf.

[68] Bently, LEAP Bridge, Bentley. (2017).

[69] Trimble, Tekla Structures, Trimble. (2018).

[70] Autodesk, Revit, Autodesk. (2018).

[71] B. McGuire, R. Atadero, C. Clevenger, & M. Ozbek, "Bridge Information Modeling for Inspection and Evaluation," *J. Bridg. Eng.* 21 (2016) 04015076. https://doi.org/10.1061/(ASCE)BE.1943-5592.0000850.

[72] R.R. Armendariz, & M.D. Bowman, "Bridge Load Rating (Joint Transportation Research Program)," West Lafayette, IN, 2018. https://doi.org/10.5703/1288284316650.