Bridge Inspection Using Augmented Reality and Artificial Intelligence

Joe Gabbard, Ph.D.
COGnitive Engineering for Novel Technologies
(COGENT) Lab

&

Rodrigo Sarlo, Ph.D.
Vibrations, Informatics, and Built Environments
(VIBEs) Lab

VIRGINIA TECH™

Final Report

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle	5. Report Date
InspectAR: Bridge Inspection Using Augmented Reality and Artificial Intelligence	6. Performing Organization Code.

7. Author(s)	8. Performing Organization Report No.
Joseph L. Gabbard, Rodrigo Sarlo	

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Virginia Tech, Department of ISE 1145 Perry St. Durham 253 Blacksburg, VA 24061	11. Contract or Grant No. CO4616

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered Final
NMDOT Research Bureau 7500B Pan American Freeway Albuquerque, NM 87109	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Bridge inspections are critical for maintaining infrastructure safety and reliability, requiring inspectors to conduct thorough evaluations of structural components and document their conditions. However, traditional inspection workflows rely heavily on manual measurement and data entry, which introduces inefficiencies, inconsistencies, and transcription errors. This research explores the application of Augmented Reality (AR) and Computer Vision (CV) in bridge inspections through the development of an AR-based head-mounted display system. By overlaying digital annotations, enabling hands-free interaction, and integrating automated measurement tools, the proposed system supports enhanced data collection, visualization, and more efficient data conversion in the office. Through field testing with bridge inspectors, we evaluated the usability and practicality of various AR-based inspection techniques. Key findings from real-world testing indicate that AR improves spatial awareness by anchoring annotations and data directly to defects of interest on the bridge structure. The integration of CV for crack quantification has value for facilitating measurements at a distance, but its usability as a fully automated tool is low at this time. The team determined that the best way to ensure reliability in assessments was a hybrid approach where CV is guided by the inspector, with flexibility to bypass it altogether if preferred

17. Key Words: Augmented Reality, Bridge Inspection, User-Centered Design, Computer Vision	18. Distribution Statement Available from NMDOT Research Bureau

19. Security Classi. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
None	None	51 ??	

Form DOT F 1700.7 (8-72)

INSPECTAR: BRIDGE INSPECTION USING AUGMENTED REALITY AND ARTIFICIAL INTELLIGENCE

by

Joseph L. Gabbard
Professor of Human Factors and HCI
Virginia Tech

Rodrigo Sarlo
Assistant Professor of Civil and Environmental Engineering
Virginia Tech

Report NMXXXXXXX

A Report on Research Sponsored by

New Mexico Department of Transportation
Research Bureau

in Cooperation with
The U.S. Department of Transportation
Federal Highway Administration

May 2025

NMDOT Research Bureau
7500B Pan American Freeway NE
Albuquerque, NM 87109

PREFACE

The research reported herein describes the design, development and testing of a head-worn Augmented Reality and Computer Vision solution to assist in measuring, documenting, and reporting defects as part of federally-mandated bridge inspection process.

NOTICE

The United States Government and the State of New Mexico do not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of this report. This information is available in alternative accessible formats. To obtain an alternative format, contact the NMDOT Research Bureau, 7500B Pan American Freeway NE, Albuquerque, NM 87109) or by telephone (505) 412-9553.

DISCLAIMER

This report presents the results of research conducted by the author(s) and does not necessarily reflect the views of the New Mexico Department of Transportation. This report does not constitute a standard or specification.

ABSTRACT

Bridge inspections are critical for maintaining infrastructure safety and reliability, requiring inspectors to conduct thorough evaluations of structural components and document their conditions. However, traditional inspection workflows rely heavily on manual measurement and data entry, which introduces inefficiencies, inconsistencies, and transcription errors. The adoption of the Specification for the National Bridge Inventory (SNBI) has increased the complexity of these assessments, further highlighting the need for modernized inspection methods.

This research explores the application of Augmented Reality (AR) and Computer Vision (CV) in bridge inspections through the development of an AR-based head-mounted display system. By overlaying digital annotations, enabling hands-free interaction, and integrating automated measurement tools, the proposed system supports enhanced data collection, visualization, and more efficient data conversion in the office. Through iterative design reviews, development and field testing with professional bridge inspectors, we built InpsectAR (an AR/CV App for bridge inspection) and evaluated the usability and practicality of various inspection techniques.

Key findings from real-world testing indicate that AR improves spatial awareness by anchoring annotations and data directly to defects of interest on the bridge structure. The integration of CV for crack quantification has value for facilitating measurements at a distance, but its usability as a *fully automated* tool is low at this time. The team determined that the best way to ensure reliability in assessments was a hybrid approach where CV is guided by the inspector, with flexibility to bypass it altogether if preferred.

ACKNOWLEGEMENTS

We would like to greatly acknowledge Nathaniel Pittman for his dedicated support of this research, as well as his foresight and vision for ways in which AR+CV can improve New Mexico DOT procedures and inspector safety/well-being. We further like to thank Angelo Armijo, Abigail Moya & Tesha Henderson for the contracting support. This project would not be possible without leadership support from Edward Halbig and Randy Trujillo, which we greatly appreciate as well. Lastly, we want the thank the following bridge inspectors for their time and insights gleaned through our user-centered design approach: Favio Casillas (District 1), Raymond Munoz (District 1), Erick Avila (District 1), Ben Newman (District 2), and Justin Roybal (District 2).

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Bridges play a crucial role in transportation networks, requiring regular inspections to ensure their structural integrity and assess maintenance needs. To meet regulatory standards, trained inspectors perform on-site evaluations every two years, systematically evaluating the condition of various bridge components (*1*). Since the adoption of the Specification for the National Bridge Inventory (SNBI) in the 1990s, inspectors have been required to document the condition of hundreds of bridge elements, such as steel beams and prestressed concrete decks, identify and quantify relevant defects, and assign condition ratings accordingly (*2*). The collected data must then be formatted and submitted to the Federal Highway Administration (FHWA) in a structured digital report, requiring inspectors to synthesize field observations into predefined categorical classifications (*3*).

The introduction of element-level condition data under the SNBI has facilitated the development of digital databases, providing asset managers with more detailed and structured records for bridge maintenance. However, this shift has placed increasing pressure on traditional paper-based inspection workflows. Inspectors in the field must now measure, annotate, and document an expanding list of structural elements, only to later convert their notes into digital reports at the office. This process not only adds to their workload but also increases the likelihood of transcription errors. Furthermore, studies have shown significant inconsistencies in recorded measurements among inspectors, sometimes exceeding 100 % variation (*4*). These challenges have driven growing interest in modernizing field inspection practices through digital solutions to enhance measurement accuracy and eliminate errors associated with manual data entry.

While these inspections are essential for maintaining infrastructure reliability, the conventional process remains labor-intensive, requiring inspectors to manually record data, transcribe field notes, and later input information into digital systems. This reliance on manual data entry not only consumes valuable time but also increases the risk of errors, inconsistencies, and variations between different inspectors' assessments. Emerging technologies, particularly Augmented Reality, offer transformative potential in addressing these challenges by providing intuitive, real-time visualization of structural information and streamlining data collection. With AR-based systems, inspectors can overlay digital annotations, retrieve historical condition data directly in their field of view, and interact with inspection tools hands-free—reducing cognitive load and improving efficiency. Furthermore, AR allows user-centric integration of emerging inspection technologies, such as computer vision (*5*) and data mining (*6*) in a physical context. By integrating AR head-mounted displays (HMDs) into bridge inspections, inspectors can focus more on structural assessments rather than administrative tasks, leading to faster decision-making, improved data consistency, and a more user-friendly inspection experience.

The overarching objective of this research was to explore AR head-mounted displays (HMD) coupled with computer vision (CV) as a core tool for digitizing bridge inspections conducted in outdoor environments. Hereafter, we refer to this tool as *InpsectAR*. This work aimed to evaluate how well InspectAR supports inspectors in real-world scenarios, considering both its advantages and limitations when deployed on actual bridge sites. A key aspect of this investigation was assessing the usability and effectiveness of the InpsectAR—ensuring that the system is intuitive, minimizes cognitive load, and aligns with the practical needs of inspectors working in complex field conditions. Furthermore, this research analyzed how inspectors interact with AR-based tools

over multiple iterations, gathering feedback to refine the interface and improve its functionality. By focusing on user-centered design principles, this project sought to identify potential barriers to adoption and develop practical solutions that enhance the overall inspection process.

In summary, this research makes the following key contributions:

1) Application of AR in Bridge Inspection: We explore how AR can be integrated into bridge inspection workflows, assessing its potential to enhance data collection, visualization, and overall efficiency in real-world scenarios;
2) Design and Development of CV-based Measurement and Visualization Technique for AR Bridge Inspections; and;
3) Insights from Real-World Evaluations: Through outdoor testing with professional bridge inspectors, we identify key lessons learned regarding usability, adoption challenges, and the practicality of AR technology in field conditions.

RELATED WORK

Augmented Reality is increasingly being explored as a tool to enhance bridge inspection workflows, improve inspector efficiency, and integrate machine learning (ML) for automated defect detection and measurement. However, while AR offers potential advantages in localization, data digitization, and automation, most current studies have focused on technology development without considering inspector usability and workflow integration.

The integration of user feedback in AR design is critical for adoption in real-world settings. Studies that simulated AR-based inspections in Virtual Reality (VR) identified several usability challenges. Chettupuzha et al. (2006) (*7*) developed a VR-based bridge inspection system and found that inspectors preferred text-based reports over audio-based documentation, and that sequential damage review was less effective than an X-ray visualization method. Similarly, Smith et al. (2022) (*8*) tested an AR prototype and found that inspectors preferred manual defect annotation over automated logging, highlighting the need for interactive, rather than fully automated, AR interfaces. Ramakrishna et al. (2016) (*9*) compared AR display devices and concluded that tablet-based AR was favored over head-mounted displays due to comfort and ease of interaction. Kouch et al. (2020) (*10*) evaluated the HoloLens 1 and identified glare, long loading times, and virtual keyboard limitations as usability concerns.

Samuel et al. (2022) (*11*) developed an AR-based field data collection tool that enabled inspectors to directly annotate BIM models in real-time, reducing task completion time by 50%. Wickens et al. (2021) (*12*) and Lazaro et al. (2021) (*13*) explored transparent automation in AR applications, where inspectors could adjust AI settings and override incorrect detections, resulting in better inspection accuracy. Al-Sabbag et al. (2022) (*14*) implemented a human-machine collaborative workflow, allowing inspectors to validate AI-detected defects in AR and significantly reduce false positives and negatives. Billings (2018) (*15*) emphasized that human-AI collaboration improves inspection reliability compared to fully autonomous systems.

One of the key applications of AR in bridge inspection is localization. Hansen et al. (2021) (*16*) developed an AR tool using Differential GPS (DGPS) to achieve 2-4 cm localization accuracy,

enhancing inspectors' ability to view subsurface utilities. Zhou et al. (2017) (*17*) employed QR codes to align BIM models with real-world structures, identifying tunnel misalignments with a precision of 5 mm. Moreu et al. (2019) (*18*) demonstrated an AR-enhanced bridge inspection method that improved tracking of inspection points and data collection. Maharjan et al. (2021) (*19*) integrated structural health monitoring sensors with AR, enabling real-time visualization of strain data in the field.

AR-based measurement tools have also been developed to improve defect quantification. Mascarenas et al. (2021) (*20*) created an AR-based 3D spatial measurement tool that achieved < 2% error for area measurements and 6% error for linear measurements. AlSabbag et al. (2022) (2*1*) combined robotic defect detection with AR visualization, allowing inspectors to interactively verify AIgenerated defect reports. Karaaslan et al. (2019) (*22*) introduced an AI-assisted spalling detection tool where inspectors could fine-tune the detection threshold, improving defect recognition accuracy.

ML models for defect detection and segmentation are increasingly being integrated into AR-based inspection workflows. Wang et al. (2021) (*23*) implemented an SSD-MobileNet-based crack detection model on smart glasses, achieving 91.67% accuracy, though on-device computational limitations posed challenges. Malek et al. (2022) (*24*) developed a HoloLens 2-compatible crack detection model using a Canny edge detector, but image input had to be downscaled by 90% to maintain real-time processing speeds. Al-Sabbag et al. (2020) (*21*) proposed an interactive segmentation model (f-BRS) where inspectors provided foreground/background selections, allowing iterative AI refinement of defect detection. Mohammadkhorasani et al. (2023) (*25*) implemented a serverclient pipeline in which a HoloLens streamed video to an SQL database for fatigue crack detection, achieving 100% precision for cracks at close range but dropping to 25% at 2 meters.

Comparative studies of different ML architectures for defect detection indicate that DeepLabV3 is among the best-performing models. Bianchi et al. (2021) (*26*) found that DeepLabV3 achieved superior corrosion segmentation accuracy compared to conventional CNNs. Deng et al. (2020) (*27*) enhanced Faster R-CNN architectures with deformable convolutions, improving segmentation accuracy by 4-5%. Yang et al. (2018) (*28*) developed a fully convolutional network-based crack detection model with 82% precision. Li et al. (2020) (*29*) integrated U-Net and CliqueNet architectures for high-resolution crack segmentation, significantly improving defect detection in challenging lighting conditions. Zong et al. (2022) (*30*) introduced DETRS, a transformer-based object detection model, outperforming traditional CNNs in defect classification. Munawar et al. (2022) (*31*) leveraged a GAN-enhanced CNN model for infrastructure segmentation, achieving 85% precision. Jin et al. (2021) (*32*) proposed a thermal imaging and MLbased approach for subsurface corrosion detection, demonstrating 88.8% accuracy.

The effectiveness of these ML-based approaches in AR-assisted bridge inspection depends on balancing computational efficiency with usability. Mojidra et al. (2022) (*33*) developed a motion tracking algorithm for detecting fatigue cracks, presenting an alternative to ML-based segmentation for inspectors who require real-time feedback without reliance on deep learning models. Studies have highlighted the necessity of ensuring that AI-driven AR inspection systems

do not operate in isolation but are instead designed as human-AI collaborative tools, where inspectors retain control over defect identification and verification.

Despite advancements, real-world usability testing remains limited in AR-based bridge inspection. The integration of DeepLabV3based segmentation models and server-client architectures aligns with best practices in ML-assisted defect quantification, but future research must prioritize user feedback, field testing, and adaptive AI integration to ensure that AR-based inspection tools are both effective and practical. By incorporating interactive AI, intuitive AR interfaces, and inspector-driven design, AR can transition from an experimental technology to a standard tool for improving bridge safety and efficiency.

WORK PERFORMED & RESULTS

The main work reported herein is organized into the following sections:

- Industry and Literature Review;
- Research into Localization;
- Research into Defect Automation; and;
- Design, Development and Evaluation of an AR+CV Application we call "InspectAR".

INDUSTRY & LITERATURE REVIEW

Task 1 from the Statement of Work required that VT "Provide a literature review". During the first quarter of Year 1 we conducted a literature review, along with an industry review to understand who is conducting research related to the project, what that research is, and how it may inform/refine our work. The literature review was delivered 16 November 2023. For details, see the full literature report provided.

Summary of findings: It is evident that there is a wealth of technologies for digitizing the bridge inspection workflow, ranging from highly practical data entry systems to sophisticated machine learning models. In this landscape, AR stands out as a platform for centralizing this wide array of digital tools while also capturing spatial context and integrating easily into existing inspection workflow. The last detail is crucial, since it avoids complications associated with introducing new inspection guidelines. Our review has shown, however, that AR implementation in bridge inspection is still in infancy, with the most recent examples related to either visualization or virtual measurements. These are far from realizing the full potential associated with creating a complete AR workflow. Aca-demic research has come closer to this goal by incorporating computer vision (CV) into AR as a means to streamline the documentation process, yet to our knowledge all research has been performed in structured lab settings. There is virtually no research on real bridges with real inspectors. There are significant technical challenges to be overcome when it comes to implementing such systems in the field, primarily in relation to 1) outdoor localization and 2) reduced CV performance in unstructured environments. The main challenge, however, is human-centric: So far, no work has considered the role of the inspector in the workflow. This applies not only to the interaction with automated processes like CV but also to techniques for leveraging inspectors' domain knowledge to mitigate some of the technical challenges. In other words, there is a need to shift from a focus on high-performance autonomous processes to human-computer interaction (HCI) methods that leverage imperfect autonomy. Fortunately, HCI, also known as

Human Automation Interaction (HAI), is a well-established field with a wealth of knowledge on how to interface human and computer processes. Future research in AR-based bridge inspection should focus on incorporating HCI techniques to design interfaces that inspectors will actually use, thus paving the way for implementation.

LOCALIZATION RESEARCH

Localization refers to the process of estimating an inspectors head pose along six important dimensions – x,y,z (*where inspectors are located* in space defined in cartesian coordinate space), and h,p,r (rotations around x,y,z axis that defines w*here inspectors are looking*). The more accurate the localization, the more compelling and powerful the AR annotations can be. For example, in order to overlay a previously recorded crack profile atop the same crack at a present date, we must be able to accurately localize the inspector relative to that crack (and relative to the previous visit).

For localization we performed a technical examination of Microsoft's Mixed Reality Toolkit version 3, hoping that we could leverage an existing set of World Locking Tools (WLT) in parallel with the MRTK 3 toolset. World locking tools take the burden of world-locking AR graphics off of the developer, as well as enabling scenarios that were previously unreachable. It currently supports the HoloLens family of devices via Unity's built-in VR support. World Locking Tools locks the entire holograph space of your application to the physical world – i.e., assists in localization. A hologram put in position relative to physical world features will stay fixed relative to those features, as well as remaining fixed relative to other holograms.

World Locking Tools scale naturally with both the complexity and size of the scene. Large models, large collections of models, and multi-room environments are all handled gracefully. In conclusion, it appeared WLT was not yet supported in MRTK 3, so for the remainder of the project we focused on WLT integration with MRTK 2.

Early in Year 1, we worked to understand and define the boundaries and localization capabilities of WLT. We examined both a large outdoor space on campus (Figure 1) as well as a bridge location in Blacksburg, Virginia (Figure 2).

FIGURE 1 We've tested WLT at the "Ag Quad" at Virginia Tech, the total trackable distance is 250m, includes both indoor and outdoor areas, with a total localization area of ~1.5 acres.

FIGURE 2 We also tested WLT to confirm offline tracking persistence. Virtual objects (red boxes and blue/green markings along paved path) maintained their position between

October 4, 2023 and November 8, 2023 visits. Further, the October testing was done mid-day before the tree leaves fell to the ground. The November test occurred closer to dusk and after leaf fall. It took ~15-20 seconds for WLT to recognize the bridge environment (which we can likely reduce with manual specification of bridge). The AR device + WLT localized and tracking our user through both light and dark transitions.

Outdoor tracking is more complex than indoor due to environmental factors, lighting, and sensor limitations. It is an active area of research, where our group (and others) are making progress in identifying new ways to localize in large outdoor locations (such as bridges). During this work, we determined that WLT can be tuned to accommodate varying bridge sizes. We were unable to test bridges of varying sizes, however, we are confident that our experiments localizing will support structures the size of the Socorro Bridge (approx. 440 ft long) – which we later confirmed during UX assessments. However, future work with NMDOT to examine larger bridge sizes may be needed to better understand how this constraint may (or may not) actually come into play.

DEFECT AUTOMATION RESEARCH

Year 1 Work in Defect Automation

In Year 1 we examined the relationship between inspector-guided defect identification and measurement and computer-based defect identification and measurement. This relationship is critical to future civil infrastructure inspection efforts, since a completely automated system may not be reliable, and a completely human/manual system provides no value-added over today's approaches.

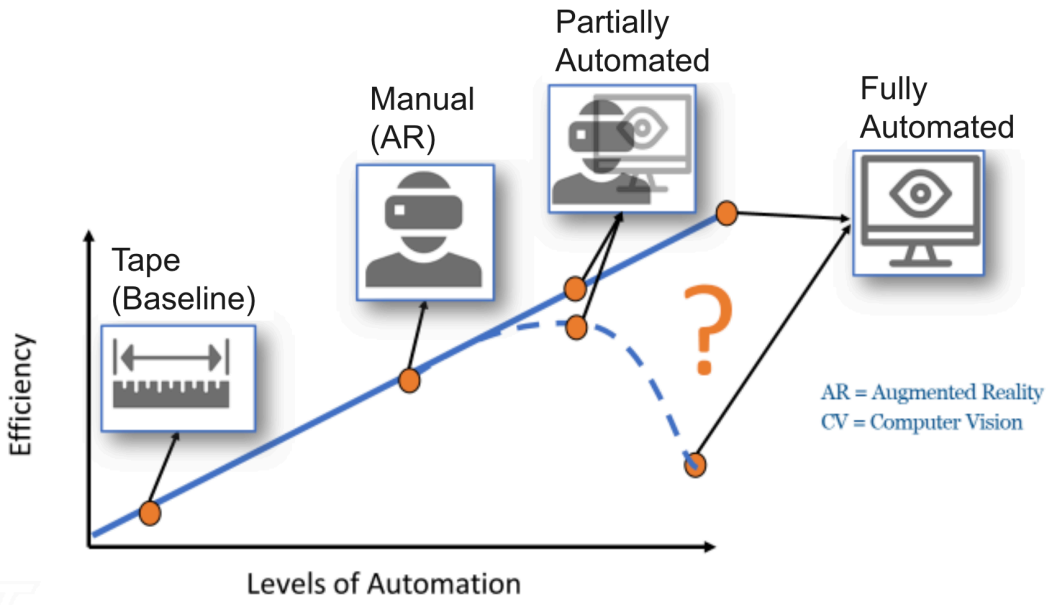

FIGURE 3 When considering how AR & CV can increase efficiency of bridge inspection, we can first think about what level of automation is optimal. Our hypothesis is that fully automated systems will not increase efficiency, and further decrease inspectors' trust and willingness to adopt new technologies in the field.

To examine the space depicted in Figure 3, we conducted a user study asking the following research question: *How does (imperfect) automation affect the perceived usability and difficulty of a crack documentation task?*

The study employed 20 student participants and 2 inspectors and examined AR UIs in each of the four conditions shown in Figure 3. An example of the participant's view is shown in Figure 4.

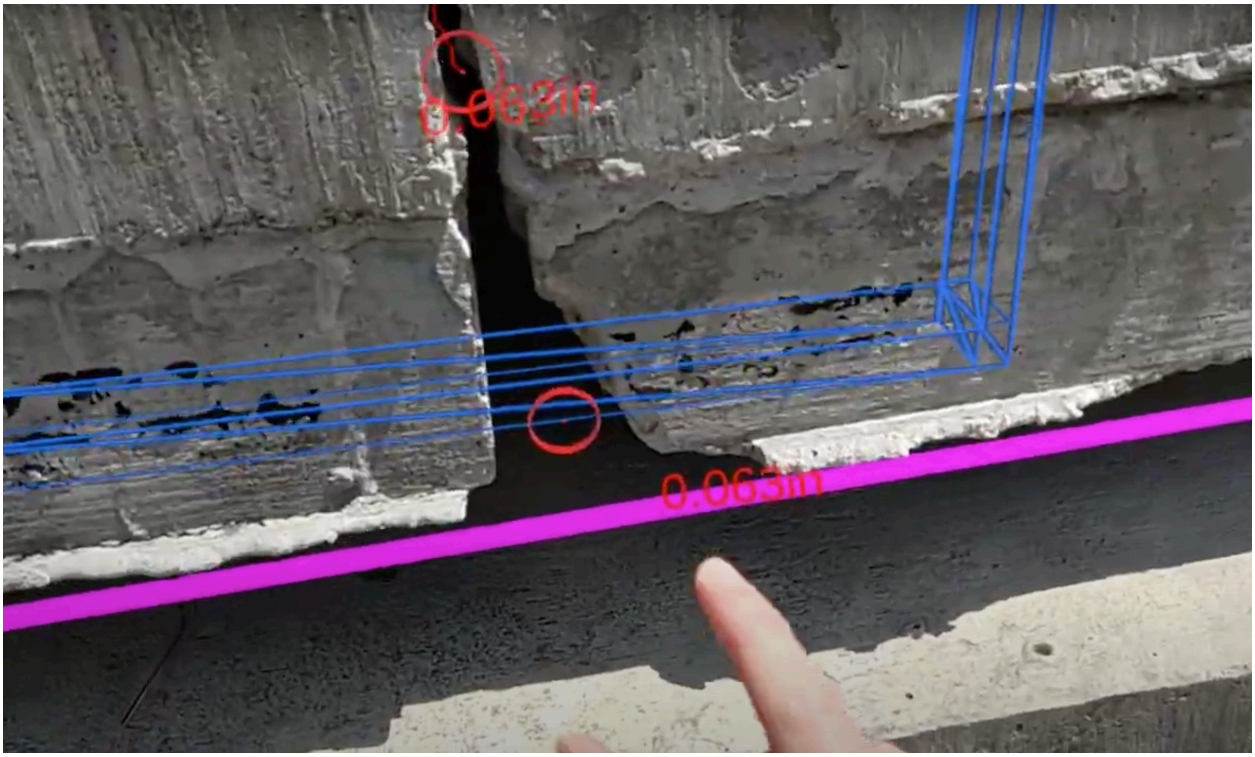

FIGURE 4 In the user study, participants measured overall crack area (shown as blue bounding box/volume), as well as maximum crack thickness (shown in red circle). In conditions which allowed user interaction (manual AR and partial automation) participants interacted with the crack and AR graphics using bare hands.

The study showed that participants; (1) performed at least as well with AR techniques as compared to manual tape measurements; (2) preferred some manual interaction, especially when able to correct/tweak suggestions from AI; and (3) did not like automated methods that did not afford human correction. Moving forward, the study also suggested that we needed to improve our crack measurement interactions and underlying algorithms.

The following subsections describe our extensive efforts to do just that -- improve the underlying algorithms to detect and measure concrete cracks using computer vision.

CV/AI for Crack Measurement

For this thrust of work, we sought to understand how best to apply computer vision (CV) and artificial intelligence (AI) to not only automatically detect a concrete crack, but also quantify, or

measure the crack width. The goal is to achieve sub-millimeter accuracy at an inspectors' arm length. The challenge is that the majority of state-of-the-art crack detection models are not trained to measure width, only to detect the presence and extent of a crack. As shown in Figure 5.

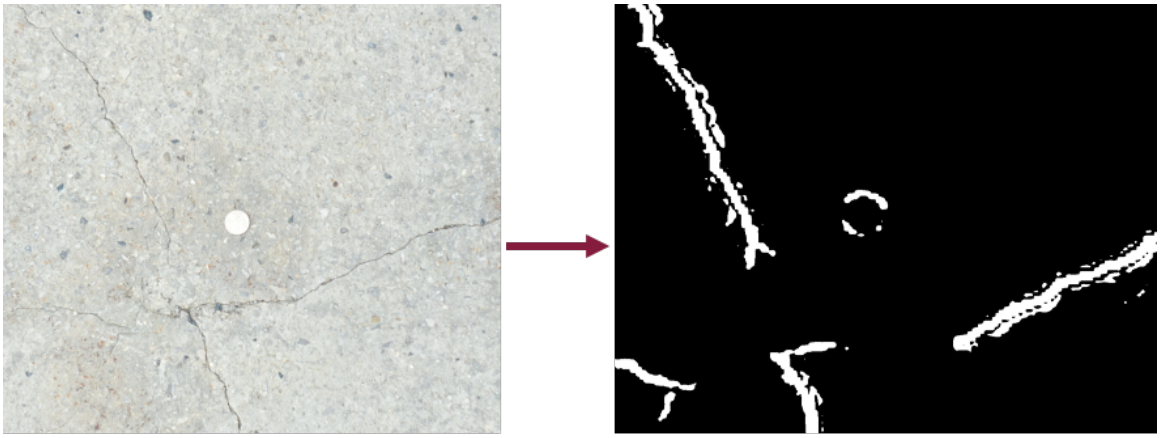

FIGURE 5 When applying traditional machine learning that leverages convolutional neural networks (CNNs) -- a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. These approaches can *detect* some (portions) of cracks, and importantly are not designed to *articulate* the actual crack width (which is why the right most image shows uniform "crack widths" in white.

As a next step, we then examined how traditional edge detection algorithms might help – since these techniques could (theoretically) identify edge boundaries on both sides of a crack since the techniques focus on transitions from high to low contrast (Figure 6).

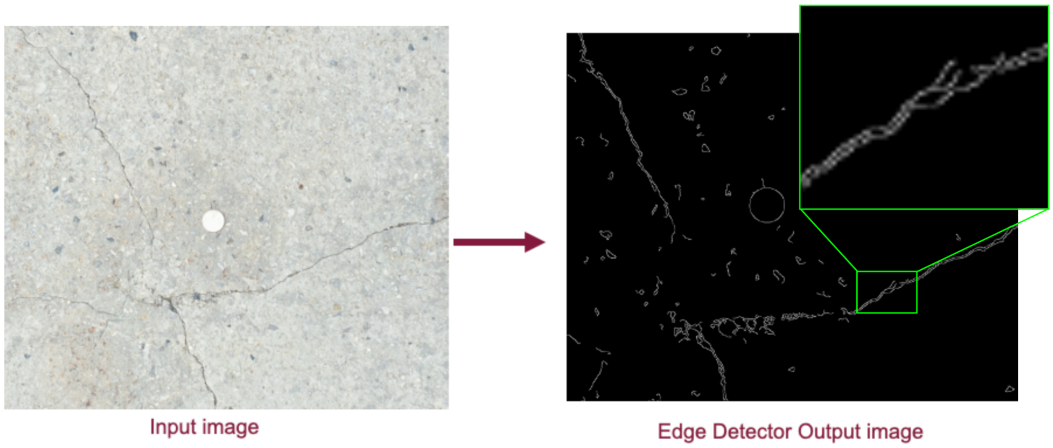

FIGURE 6 When applying traditional edge detection approaches, we find that detection of all cracks is not perfect, but separate edges of cracks are identified, which could allow for more sensitive estimations of crack width along all cracks (and determination of maximum crack width).

Initially we combined the two approaches mentioned above to explore whether this novel combination could result in good detection of cracks and good estimate of crack widths. The results were promising, as shown is Figure 7 below.

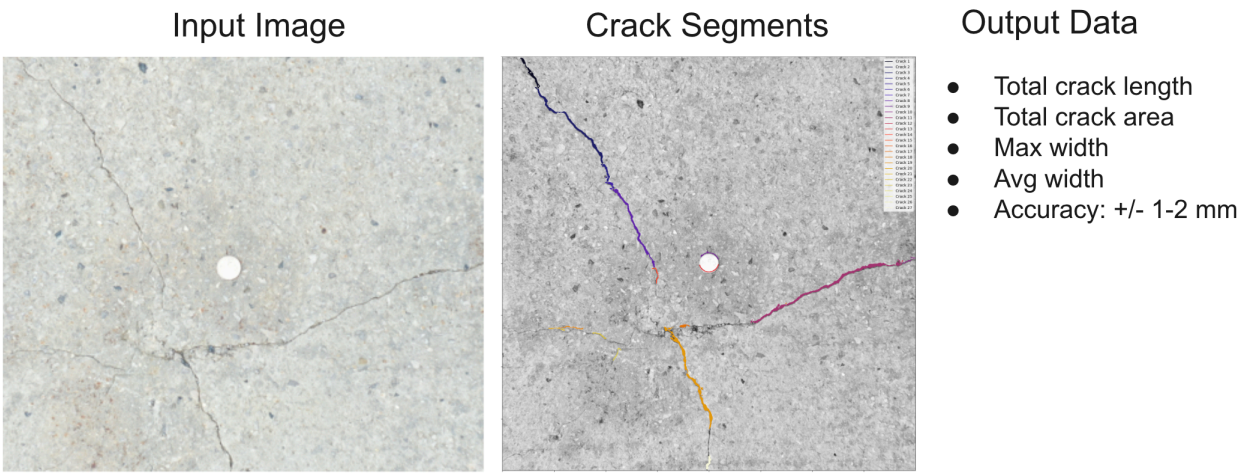

FIGURE 7 When combining CNNs and edge detection techniques we are able to more accurately quantify the total crack length, crack area, maximum crack width and average crack width within +/- a few millimeters (assumes a high-definition photo at close range).

However, after further testing, we decided that we needed more accuracy in detection. Thus, we have our approach to applying a combination of (1) machine learning approaches that leverage convolutional neural networks (CNNs) -- a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization, and (2) traditional computer vision approaches and algorithms to assist in identifying the edge boundaries on both sides of a crack since.

With respect to better classifying crack data needed for reporting, we formalized the definition of the crack documentation task as the collection of three main measurements, consistent with standard inspection practices and illustrated in Figure 8 -- 1) The height of the cracked region, measured vertically in the direction of gravity; 2) The length of the cracked region, measured horizontally in alignment with the length of an element; 3) The width of the largest crack within this region. These measurements were not intrinsically available from the image mask returned by the ML algorithm and required additional post-processing steps to obtain.

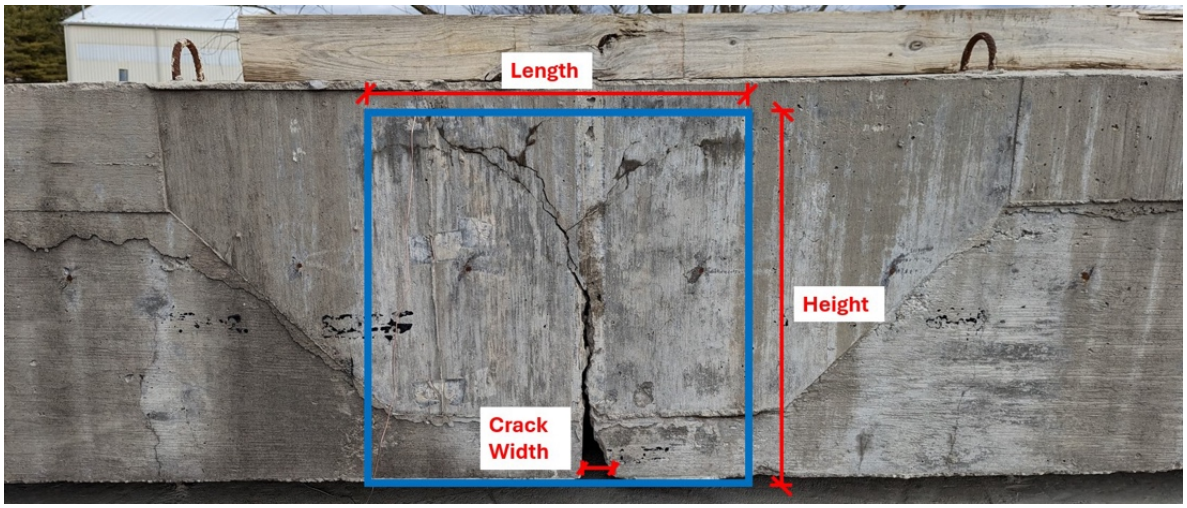

FIGURE 8 Crack measurements for standard inspections.

To derive these three measurements from the "image mask" returned by the machine learning algorithm, additional image processing was required, as detailed in Figure 9. We developed a custom crack evaluation code that expanded upon previous computer vision (non-ML) focused research in crack detection and evaluation. The innovation of this research lay in the integration of machine learning results with traditional edge detection techniques. Traditional edge detection tended to produce a very noisy image. Although this noise could be calibrated for different distances and textured surfaces, it remained a major challenge of such approaches. In contrast, the image mask generated by machine learning was significantly less noisy but lacked the detail provided by edge detection. By performing a "bitwise AND" operation on the two results, we effectively filtered the noise from the edge detection results while preserving the higher level of detail, thereby enhancing the overall accuracy of the crack evaluation.

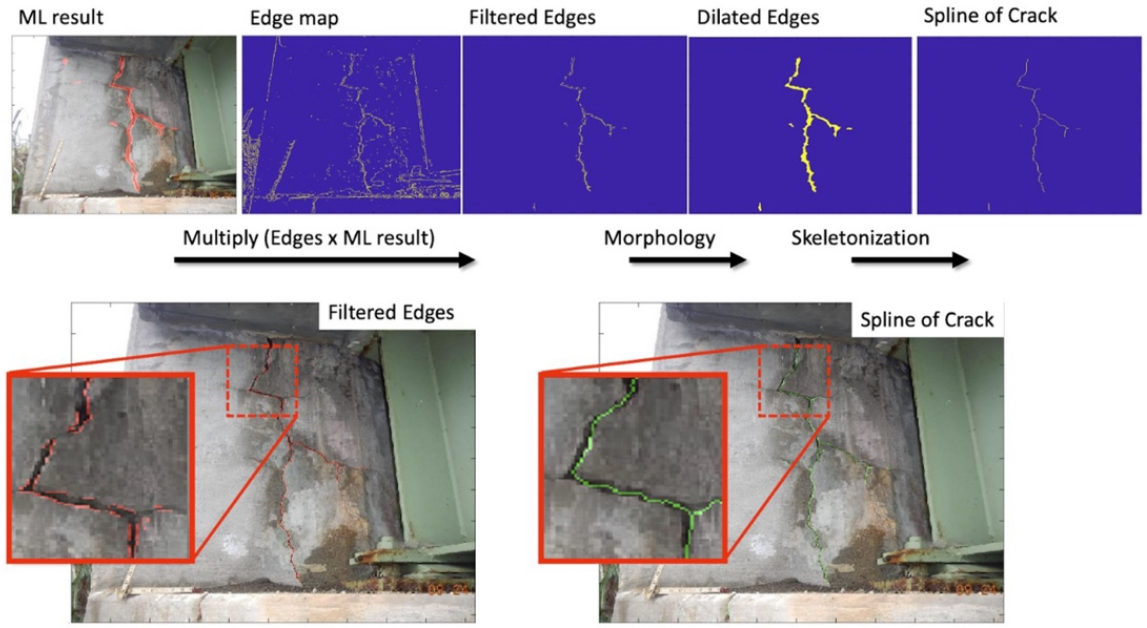

FIGURE 9 Post processing machine learning mask into crack spline.

From this set of filtered edges, we utilized Gaussian morphological operators such as "opening" and "closing" to eliminate any remaining noise and to close gaps in the results. Following this, we extracted the "skeleton" of the result, producing a single-pixel spline that traced the centerline of the crack. From this spline, we extracted the endpoints for use in our crack documentation workflow. For the "crack width" measurements, we adopted an approach similar to wherein we selected a given pixel on the centerline and extracted a nearby region of pixels. A polynomial curve of order 3 was fitted to this centerline. Then, utilizing the "filtered edges" data, we divided the same region into edges on one side of the centerline and edges on the other side. The polynomial curve, fitted to the centerline, was then offset perpendicular to the centerline in each of the two directions, using a least squares optimization to determine the optimal offset that minimized the distance from the edges on that side. The two offsets were combined to yield a "crack width" measurement in pixels.

At this point, the crack endpoints were in pixel coordinates (x,y), and the crack thickness was quantified in pixels. This data, along with the pixel coordinates of every pixel on the crack centerline, was converted to JSON format and transmitted from the server to the HoloLens 2 via the previously mentioned HTTP pipeline and offline router. The HoloLens 2 then unpacked the JSON data, projecting the centerline of the crack as a "line" object with the specified coordinates and creating "crack point" objects at each endpoint.

For the crack width measurements, given a point on the crack centerline and the thickness computed by the server, we calculated the "edge points" in pixel space above and below the crack as simply the centerline point ± thickness/2. For each of these "edge points", we projected them onto the physical surface and then computed the distance between them in 3D space. This method circumvented potential errors associated with attempting to derive a "pixels to meters" conversion factor, which would not only require a planar surface but also be susceptible to minor errors in the points used to define that planar surface.

Year 2 Work in Defect Automation

In Year 2, we closely examined our CV model described above which produces an image mask, where black pixels represented the background and white pixels indicated detected regions, such as cracks. This output format was found to be non-optimal with the data traditionally recorded by inspectors, which included the area of the bridge element affected by the defect and a condition state rating. To resolve this, the model outputs were post-processed with a custom image processing routine, converting the results into data that could be directly utilized by inspectors.

Thus, we improved our previously reported process into a more robust pipeline in order to get more accurate crack detections and measurements. Specifically, we designed a *crack segmentation and quantification pipeline* to detect and measure cracks from high-resolution images using a series of processing steps. An example of an image being processed through this pipeline as well as a flowchart of the pipeline itself is shown in Figure 10.

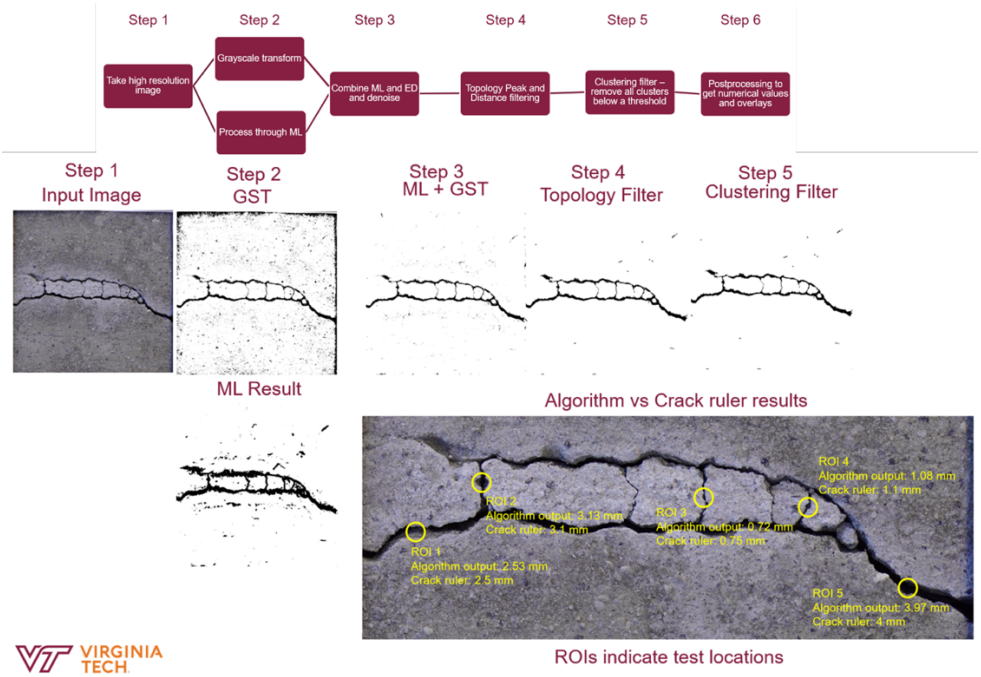

FIGURE 10 Pipeline image process example and example ROIs.

The sequence of the pipeline is as follows:

1. High-Resolution Image: The process starts with a high-resolution image, ensuring enough detail to detect cracks with precision.
2. Grayscale Transformation (GST) and Machine Learning (ML): In this step, the high-resolution image from step 1 is processed through a grayscale transformation thresholding operation and a deep learning model are performed in parallel.
   - **A.** The grayscale transformation converts the image to grayscale, highlighting contrasts to detect cracks. This works by assuming that the darkest parts of the image corelate to a crack. This process is effective in preserving the actual thickness of cracks, but it often results in a noisy image with false positives.
   - **B.** The machine learning model can reliably map the outline of the crack but tends to struggle with maintaining its thickness.
3. Combining GST and ML Outputs: To address the limitations of the individual outputs, the grayscale transformation and ML results are combined. The grayscale transformation provides the thickness, while the ML result offers a cleaner outline. This combination helps maintain the crack's thickness while removing most noise, leading to a more precise outline.
4. Topology Peak and Distance Filtering: The combined output undergoes topology-based filtering, removing noise and refining the outline. This step attempts to remove disconnected sections and false positives. The topology filter creates a density map of the pixels in the combined output, then plots it in 3D resulting in an image that looks a lot like a mountain range. The tops of the peaks are then sliced off and kept, resulting in a clean outline of the crack with minimal data loss.
5. Clustering Filter: To achieve a cleaner result, clusters below a certain threshold are filtered out. This step eliminates small, insignificant clusters that could cause errors in measurements.

6. Post-Processing for Numerical Values: The final step involves extracting numerical values from the processed results to quantify the cracks. This output is then compared against manual crack ruler measurements for validation.

The key reason for combining grayscale transformation and machine learning results is that each approach has distinct strengths and weaknesses, specifically:

- Grayscale Transformation (GST): GST is effective in detecting the outline of cracks while preserving their thickness. However, this process creates a noisy image, making it challenging to extract clear crack boundaries.
- Machine Learning (ML): ML excels at outlining cracks but struggles with maintaining their actual thickness. It can lead to inaccurate measurements if used alone.

By combining both outputs, the system aims to take advantage of the GST's ability to maintain thickness and the ML's capacity to create a cleaner outline. This integration provides a more accurate depiction of the crack's characteristics, leading to better results.

The results of our improved pipeline are illustrated through a set of box plots (Figure 11) showing the absolute error that compares our algorithm's output against manual crack ruler measurements. The box plots in Figure 11 show the absolute error across increasing distance at which the photo is taken (which corresponds to decreasing image resolutions). Each plot corresponds to a different resolution—26.66 PX, 18 PX, and 13.33PX—demonstrating how the resolution does not impact the measurement accuracy. The x-axis in each graph shows the actual width of the crack as measured by a crack ruler, while the y-axis indicates the corresponding absolute error of the algorithm's measurement. These plots help verify the system's accuracy across different resolutions and capture distances, showing that the average absolute error remains close to zero, affirming the algorithm's robustness even when image resolution varies. The data suggests that the absolute error is centered around zero, indicating high accuracy. The standard deviation is inversely proportional to the input image resolution, but the overall results remain consistent, demonstrating the system's reliability and scale-invariant capabilities.

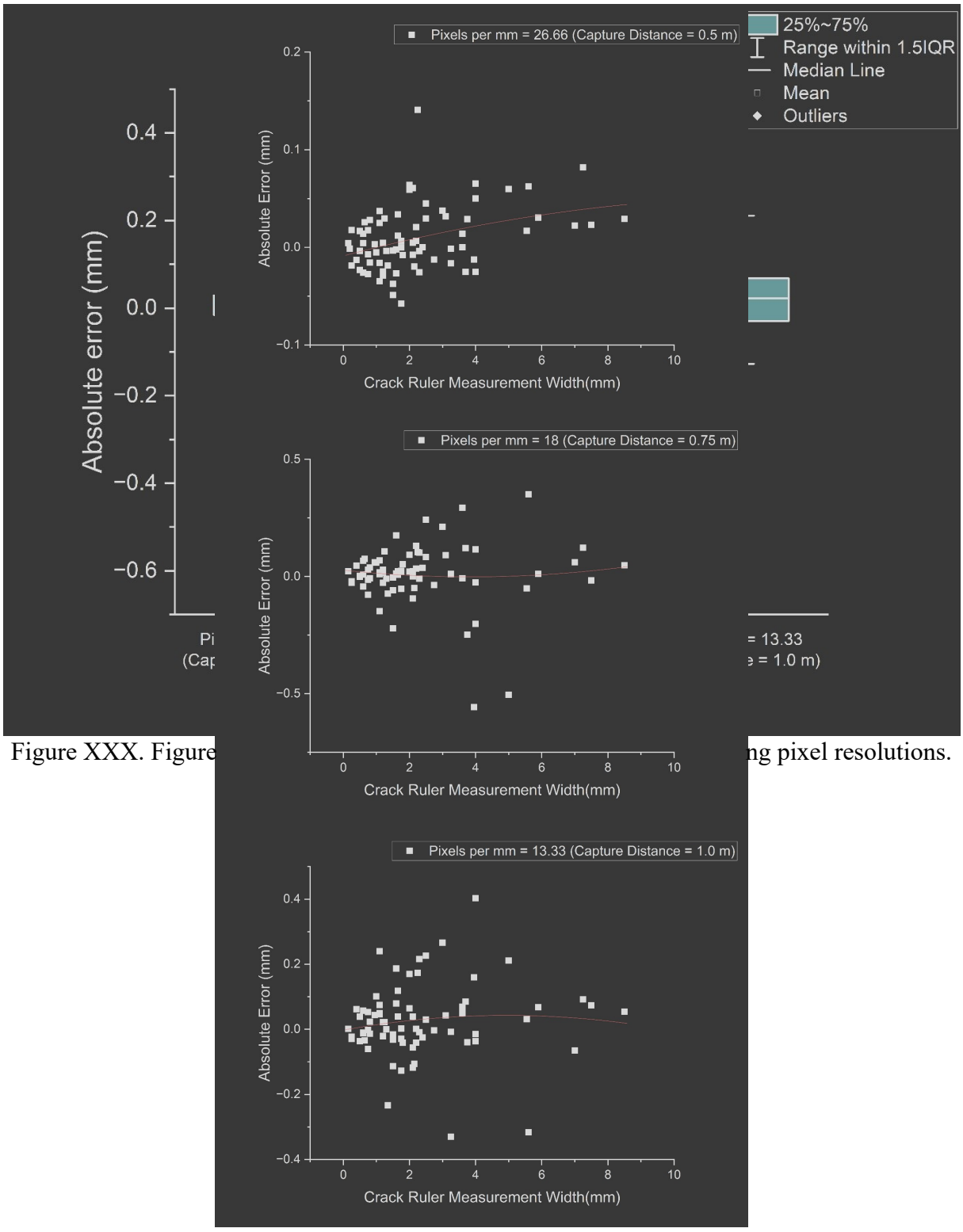

Figure XXX. Figure ng pixel resolutions.

FIGURE 11 AI/CV Pipelines measurements as compared to crack ruler using photos taken at three different distances (with three different image resolutions).

Another plot (Figure 12) combines absolute error values from three different resolutions on a single graph, plotting them against an index number representing specific regions of interest (ROI) where measurements were taken. Each point on the graph corresponds to the absolute error measured for a particular ROI at these varying resolutions.

The graph effectively demonstrates that despite changes in capture distance, and hence resolution, the absolute error for most points remains close to zero, illustrating the system's robust scale invariance. This consistency across different resolutions ensures that the system can reliably measure crack widths accurately, regardless of changes in the sampling resolution of the image. This property is crucial for practical applications where distance from the target can vary but precision is required consistently.

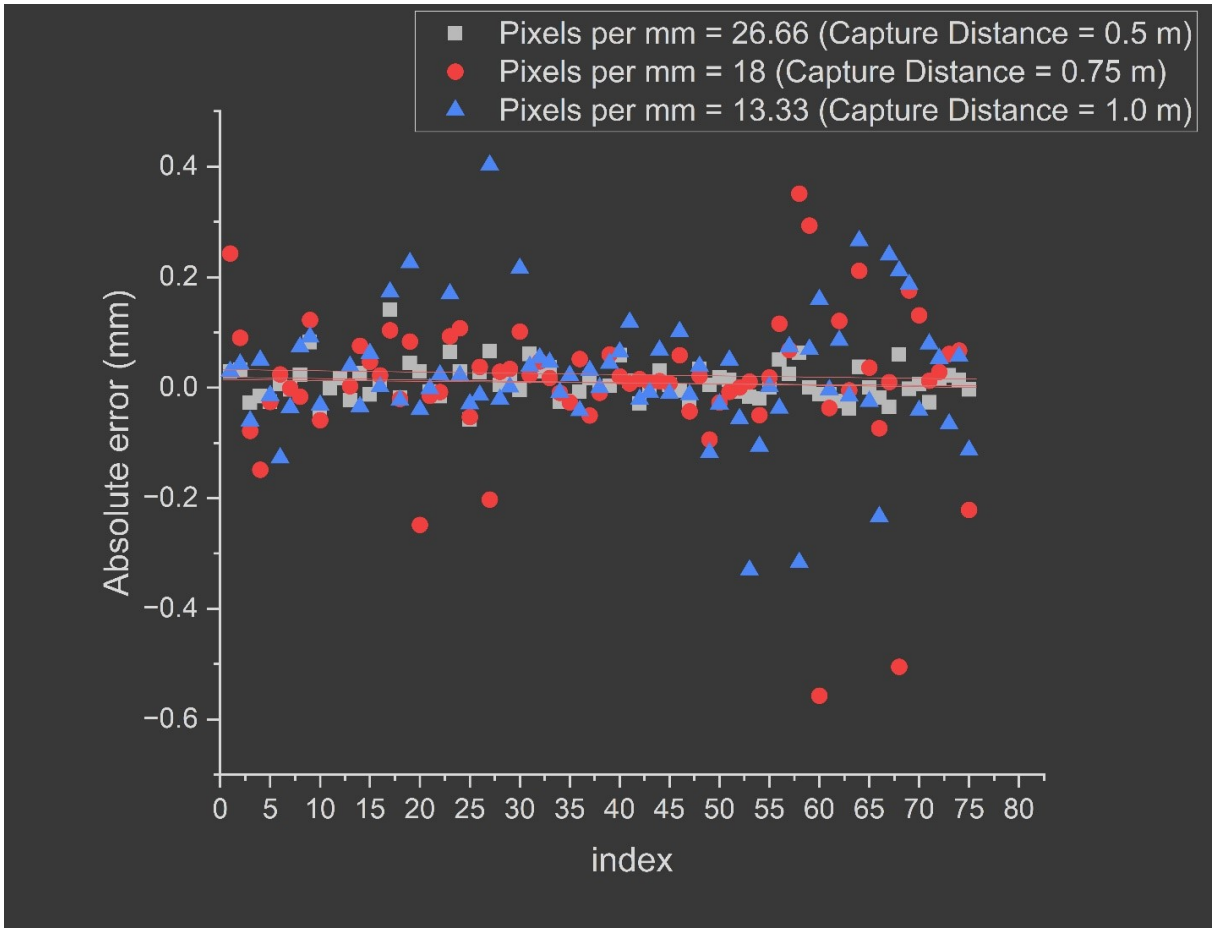

FIGURE 12 Data from figure 6 superimposed on top of on one another, demonstrating our pipeline's robust scale invariance.

Benchmarking our CV Approach

To improve the accuracy and reliability of our crack segmentation and quantification pipeline, we designed a multi-step process that operates on high-resolution images, presented in the Y1Q3 report and as described above. The process is based on a crack detection machine learning model, but leverages traditional image processing to improve the precision of the crack boundary, leading to more accurate thickness measurements. As a reminder, this process includes:

1. **High-Resolution Image Acquisition**: Capturing detailed images to ensure sufficient resolution for accurate crack detection and measurement.
2. **Grayscale Transformation (GST) and ML Processing**:
    - **Grayscale Transformation:** Converts the image to grayscale to highlight contrasts, assuming the darkest areas correspond to cracks. This step preserves the crack's thickness but introduces noise.
    - **Machine Learning:** Applies a deep learning model to delineate the crack's outline. While the ML approach provides cleaner results, it struggles to maintain accurate thickness representation.
3. **Combining GST and ML Outputs:** Merges the strengths of GST and ML to maintain the crack's thickness while reducing noise, resulting in a more precise crack outline.
4. **Topology Peak and Distance Filtering:** Further refines the crack outline by retaining only the most prominent features, reducing false positives and noise.
5. **Clustering Filter:** Removes insignificant clusters below a threshold, ensuring only meaningful crack data is retained.
6. **Post-Processing for Numerical Values:** Analyzes the processed output to extract numerical values that quantify the crack. These values are validated against manual measurements using a crack ruler, ensuring accuracy.

We benchmarked our hybrid pipeline (referred to as “hybrid”) against several state-of-the-art models which relied on machine learning only:

1. **RCUNet Model** - Based on "RUC-Net: A Residual-Unet-Based Convolutional Neural Network for Pixel-Level Pavement Crack Segmentation" by Yu, Gui, et al. (2022).
2. **DeepCrack** - Based on "DeepCrack: A Deep Hierarchical Feature Learning Architecture for Crack Segmentation" by Liu et al. (2019).
3. **HrSegNet** - Based on "Real-time High-Resolution Neural Network with Semantic Guidance for Crack Segmentation" by Yongshang Li et al. (2023).
4. **UNet_Focal** - Based on "Computer vision-based concrete crack detection using U-net fully convolutional networks" (2019).
5. **U2CrackNet** - Based on "U2CrackNet: a deeper architecture with two-level nested U-structure for pavement crack detection" by Shi et al. (2023).

These models were either reverse-engineered and trained on the original datasets or on our dataset when the original was unavailable. When pretrained models were available, they were utilized directly. The models were tested on two datasets to evaluate their performance in comparison to our hybrid model.

IOU and Dice Coefficient testing (Figures 13 and 14)

Figures 13 and 14 display the Intersection over Union (IOU) and Dice coefficient results for each model, including our hybrid model. These metrics are standard in the industry to assess the accuracy of segmentation models. The results were obtained by testing on an independent dataset of 800 images that none of the algorithms had encountered during training or testing. Figure 6 and 7 present the distribution of the most common scores for each model, with a score of 1 representing a perfect match between the detected and labeled cracks. The data are displayed in density and box-and-whisker formats, respectively. The figures shows that all models performed similarly with the exception of the DeepCrack model (orange), which performed very poorly. The hybrid model generally achieves IOU and Dice coefficients that are comparable or better than the state-of-the-art models. Notably, the hybrid model exhibits high-density peaks around 0.75 to 0.9 for both IOU and Dice coefficients, indicating a strong tendency to achieve higher scores. It does, however, have a greater number of zero scores than the other models, although they are still a minority of cases. The reason for this will be investigated moving forward. Overall, the results show that our hybrid algorithm performs well in crack segmentation as typically defined by literature, generalizing to unseen images. This gives us confidence that the algorithm has not been subject to overfitting with our data set.

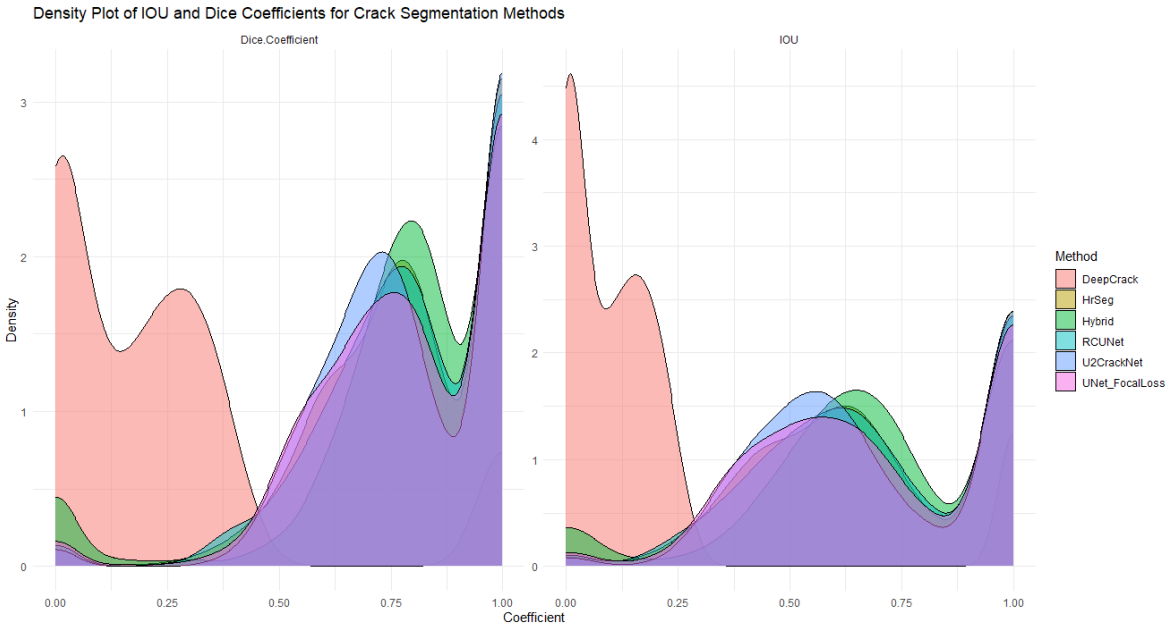

FIGURE 13 Density Plot of IOU and Dice Coefficient Results of Tested Models.

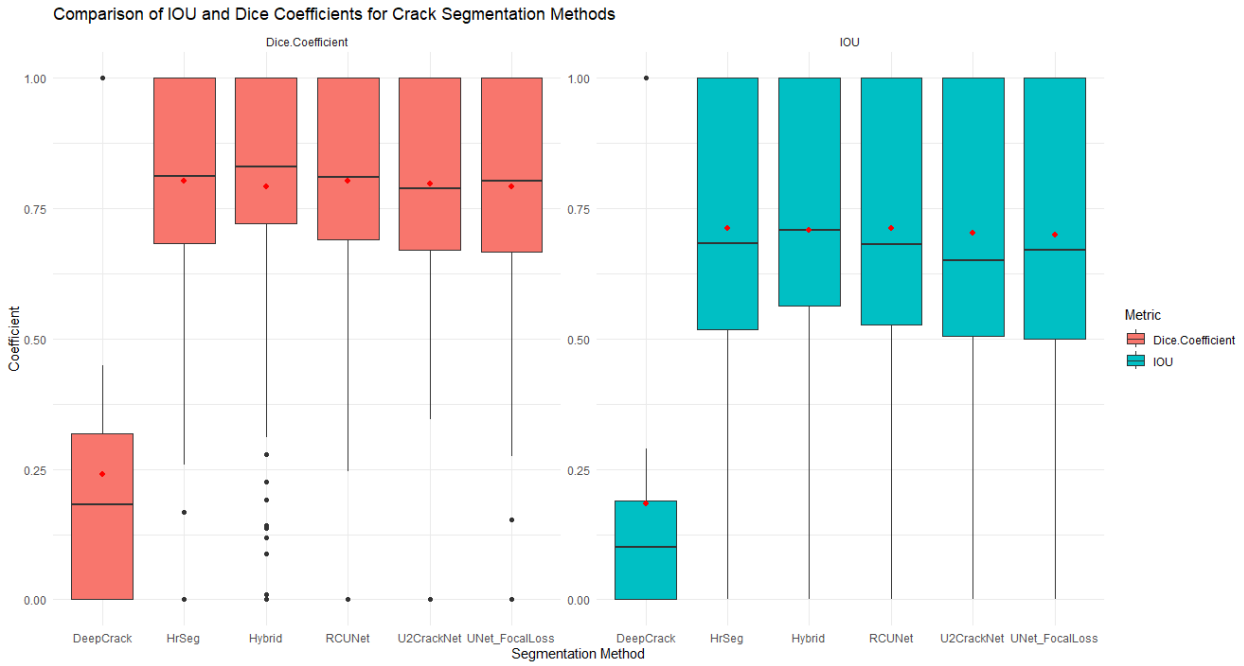

FIGURE 14 Box Plots of IOU and Dice Coefficient Results of Tested Models.

Absolute Error Across Resolutions (Figures 15 and 16)

The IoU and DICE evaluations, however, are not indicative of inspection performance, because this depends on the accuracy of crack width measurements. Figures 15 and 16 provide an analysis of absolute error for crack width measurement across different resolutions for each algorithm, comparing the performance of the hybrid model to the state-of-the-art models. The testing involved a dataset of 15 sample cracks, each captured at three resolutions—26.66, 18, and 13.33 pixels per mm—resulting in a total of 45 images. Each crack had five preselected probe points, yielding a total of 225 probe points (75 points per resolution). After processing each image through each algorithm, an absolute error was calculated at each probe point by comparing the algorithm's output to the "true" measurement obtained manually via a crack width ruler.

Figure 15 displays the absolute errors for each algorithm across the three tested resolutions, while Figure 16 presents the same data on a fixed y-axis scale to more clearly illustrate the differences in performance of the algorithms tested. In both figures, the hybrid model consistently shows lower absolute error across all resolutions, confirming its robustness and accuracy in real-world applications. A notable trend is that as the resolution decreases (from 26.66 to 13.33 pixels per mm), the absolute error generally increases for all models. However, the hybrid model's error remains lower than others across all resolutions, particularly at higher resolutions, where the errors cluster closer to zero. This trend suggests that the hybrid model is less sensitive to resolution changes compared to other models, making it more reliable for practical applications.

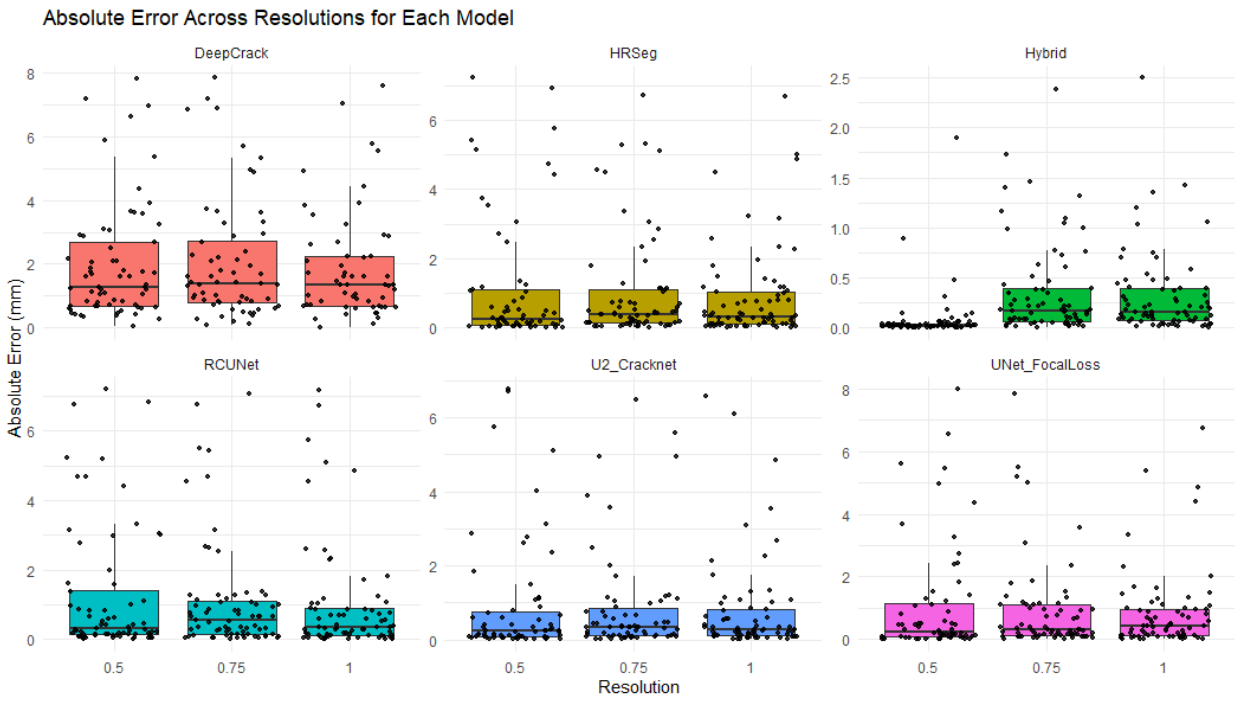

FIGURE 15 Failure Rate of Each Model Across Resolutions.

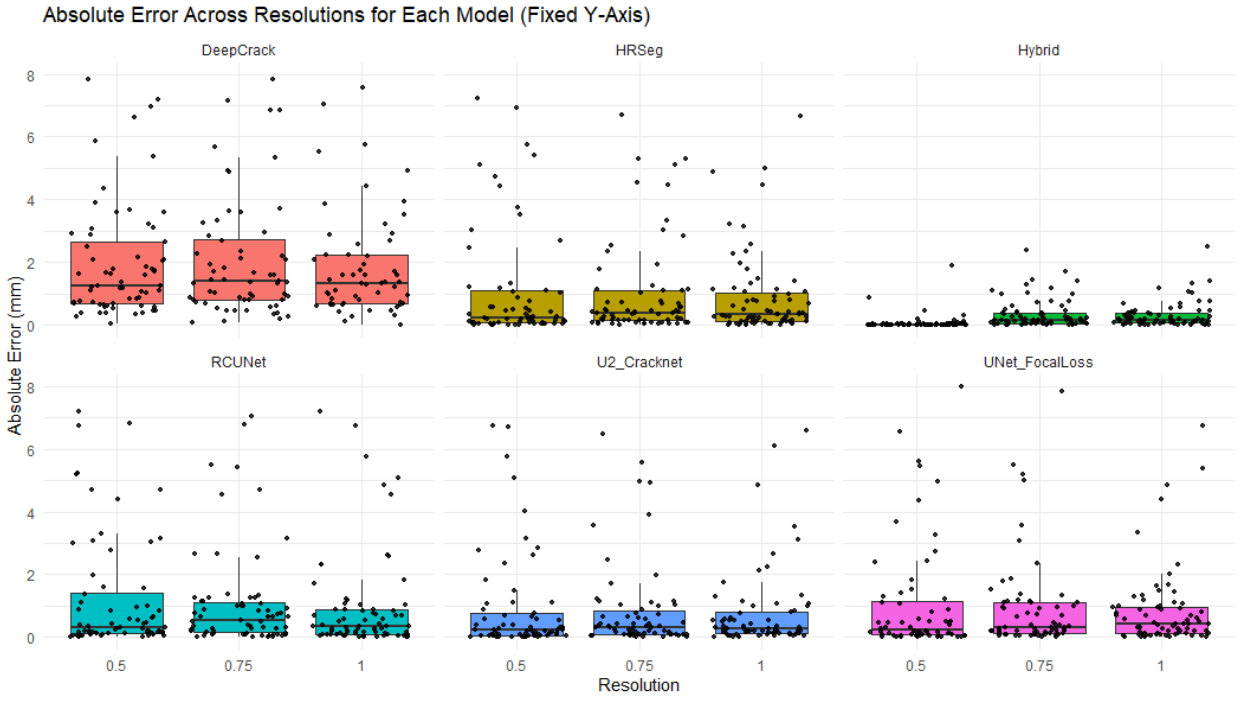

FIGURE 16 Absolute Error of Tested Models on Probed Points on Fixed Y Axis.

Failure Rate in Crack Detection (Figure 17)

Figure 17 shows the count of instances where a probe point was empty, indicating cases where the algorithm failed to detect and measure crack thickness at the preset points. This metric highlights the reliability of each model under varying conditions. The hybrid model consistently has the fewest empty probe points across all resolutions, underscoring its reliability in detecting cracks at the probe points. Models like DeepCrack and HrSegNet exhibit significantly higher failure counts, particularly at the lowest resolution (13.33 pixels per mm), indicating greater sensitivity to resolution challenges. As the resolution decreases, the failure count generally increases for all models, though the hybrid model maintains a relatively low and consistent failure rate, demonstrating its robustness in low-resolution scenarios.

Together, these results underscore the hybrid model's effectiveness and reliability. The hybrid model not only meets or exceeds state-of-the-art performance in segmentation metrics like IOU and Dice coefficients but also demonstrates lower absolute error and failure rates across resolutions, positioning it as a robust and scale-invariant alternative to purely machine learning-based methods in crack detection and quantification.

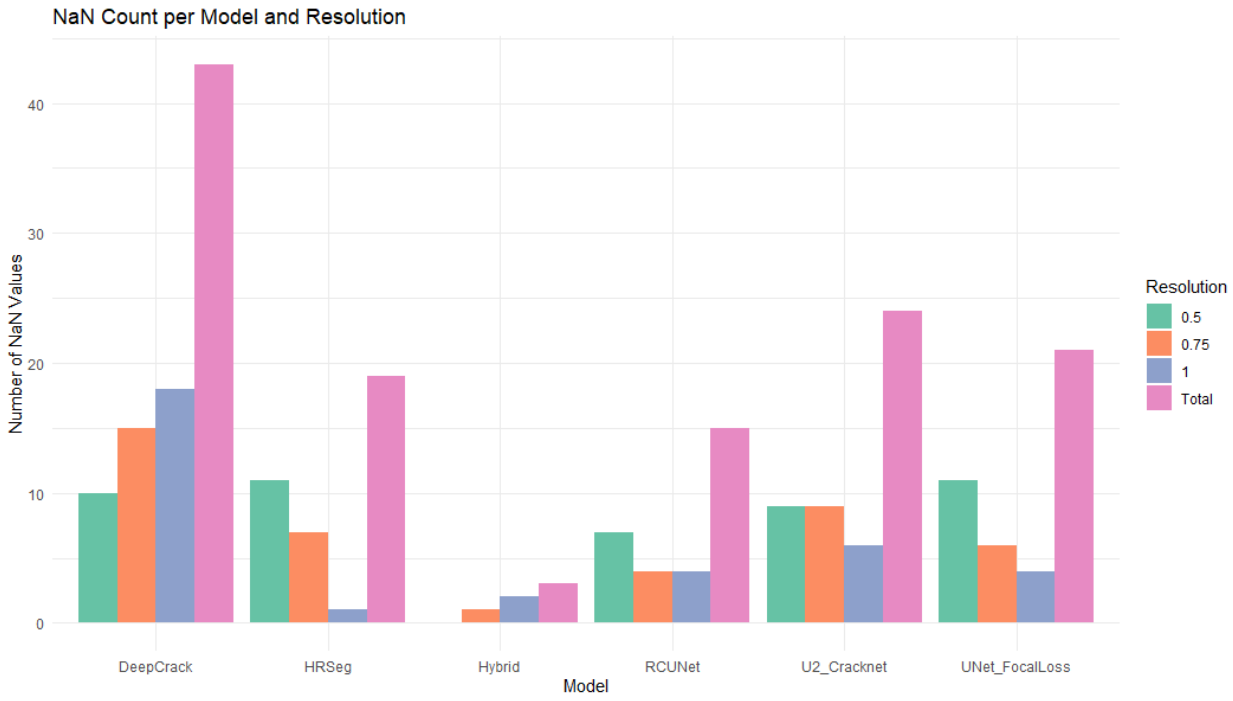

FIGURE 17 Count of instances where a Probe Point was Empty (indicating cases where the algorithm failed to detect and measure crack thickness at the preset points).

Projection of CV Results in 3D Space

A major challenge we encountered in our research was the accuracy of projecting CV-generated annotations onto the environment through the HoloLens 2. Machine learning and computer vision

operate in the 2D image space (x,y), whereas humans interact with physical objects in 3D (X,Y,Z). To enable meaningful interaction between humans and computers in a crack annotation task, the transformation between the image space (x,y) and the physical space (X,Y,Z) needs to be accurate to the millimeter scale. This is a significantly stricter requirement than for generic AR applications, where errors of a few inches might be considered acceptable. Previous research into projection in other fields (such as computer graphics) often utilized the "camera projection formula," as depicted in Figure 18, where **C** is the camera coordinates in pixels, **I** is the intrinsics matrix which corrects for lens distortion and scales the image to pixel dimensions, **E** is the extrinsic matrix which aligns world coordinates to the camera's local coordinate system, and **W** is the world coordinates of the projected point. This formula allows for the mapping of world coordinates (X,Y,Z) to pixel coordinates directly (x,y), while the inverse operation is more complex but follows similarly. However, due to issues like engineering tolerances on the camera components, lens distortions, and the eye calibration of the HoloLens 2, this method resulted in a projection error of 2-3 in, which was unacceptable for our objectives.

Camera Coordinates **Intrinsics** **Extrinsics** **World Coordinates**

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{11} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

FIGURE 18 Traditional camera projection formula

Other groups have developed more accurate projection approaches; however, these approaches still necessitated additional, user-specific eye calibration steps. Our method utilized Holographic Image Blend shader code that takes into account the existing eye calibration within the HoloLens 2 to project an image onto a floating canvas, as shown in Figure 19. This canvas provided a direct mapping from image coordinates (x,y) to canvas coordinates (Cx, Cy, Cz).

By applying basic linear algebra, we were able to draw a line from the known focal point of the camera (Fx, Fy, Fz) through the canvas coordinate (Cx, Cy, Cz), and then raycast this vector onto the environmental mesh. The intersection of this vector with the environmental mesh indicated the physical point (X, Y, Z) corresponding to the specific pixel coordinate (x,y). Through this method, we observed translational offsets on the order of 0.02 in for crack centerline projections, demonstrating the precision of our approach.

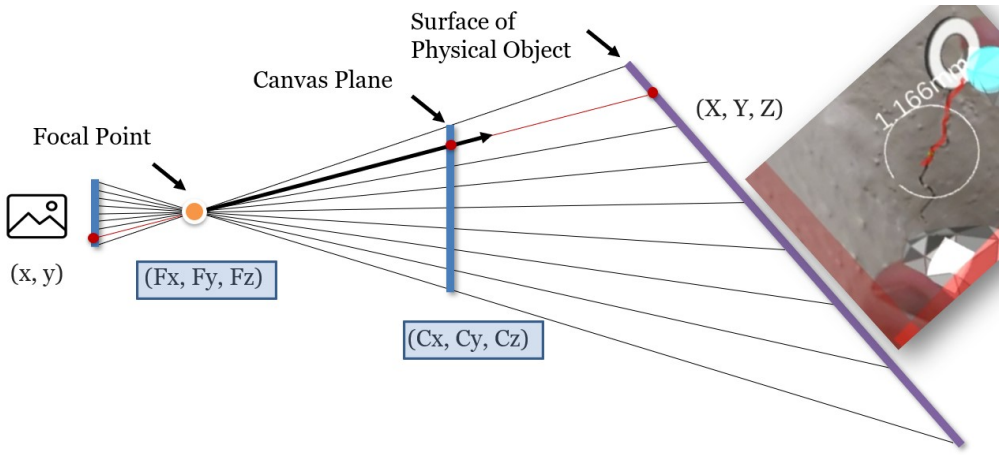

FIGURE 19 Illustration of our projection elements.

INSPECTAR – AN AR APP WITH CV FOR DOCUMENTING BRIDGE DEFECTS

Using an iterative user-centered design process, we transitioned from conceptual models and lab-based prototypes to a functional AR application tailored for real-world bridge inspections. Guided by inspector feedback and AASHTO reporting standards, we designed InspectAR's UI and interaction flows (Figure 20) to align with existing workflows. Structured menus follow the hierarchical AASHTO element format, and input options range from manual annotations to computer vision-assisted measurements, allowing flexibility across varied field conditions. The final CV features were highly specialized to provide the best possible advantage over traditional measurement measurements, rather than being intended for general use.

In this section, we describe InspectAR's core functionality, field architecture and underlaying data structures, highlighting how it supports bridge inspectors through intuitive defect documentation, real-time visualization, and integrated data management.

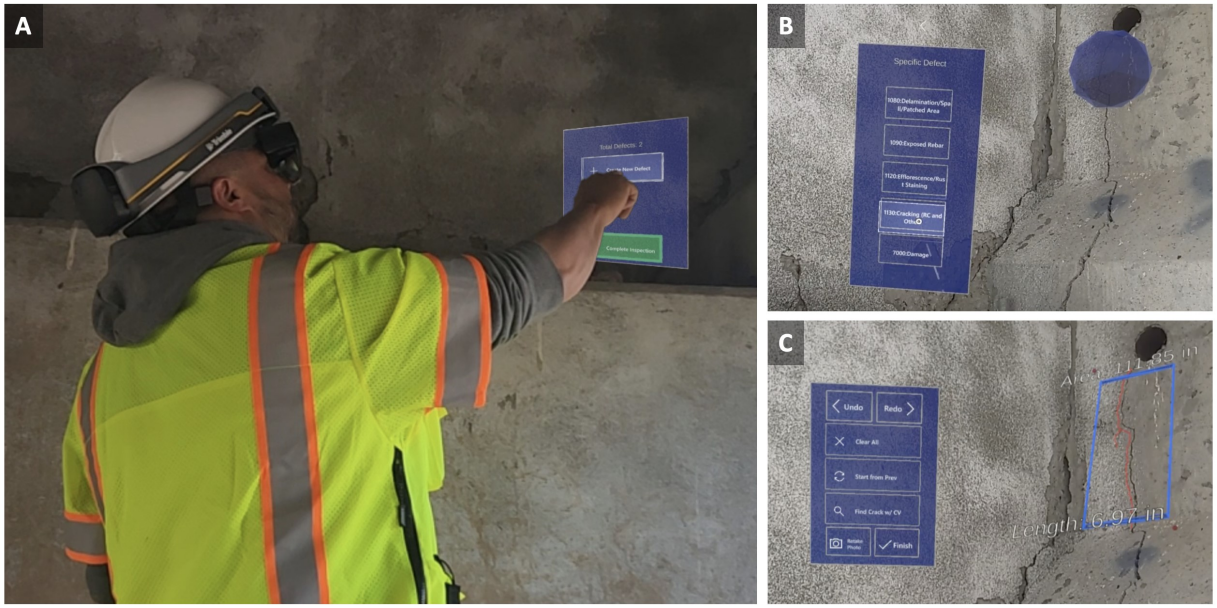

FIGURE 20 InpsectAR allows bridge inspectors to measure and document bridge defects using Augmented Reality and Computer Vision. (A): Inspectors can create new spatially-anchored defects to physical bridge surfaces, (B): assign meta data about each defect, (C): and use CV to make measurements and overlay virtual cracks that serve as a reference for future inspections.

InspectAR Core Functionality

Start/Continue Inspection Menu

When the InspectAR application loads, users are prompted to either start a new inspection or continue a previous one. This option is determined dynamically based on 1) the bridge site and 2) whether a prior inspection was completed. If the HoloLens 2 recognizes the location as previously visited, it retrieves bridge data using a unique identifier created during the initial inspection. If that inspection is incomplete, the "Continue Inspection" option appears; otherwise, only "Start Inspection" is shown. For unrecognized sites, "Start Inspection" also appears, and a new identifier is generated. The 3D geometry used for localization is stored locally on the HoloLens 2 and can be transferred between devices, while inspection data e.g., images, measurements, and notes, is maintained on the laptop server.

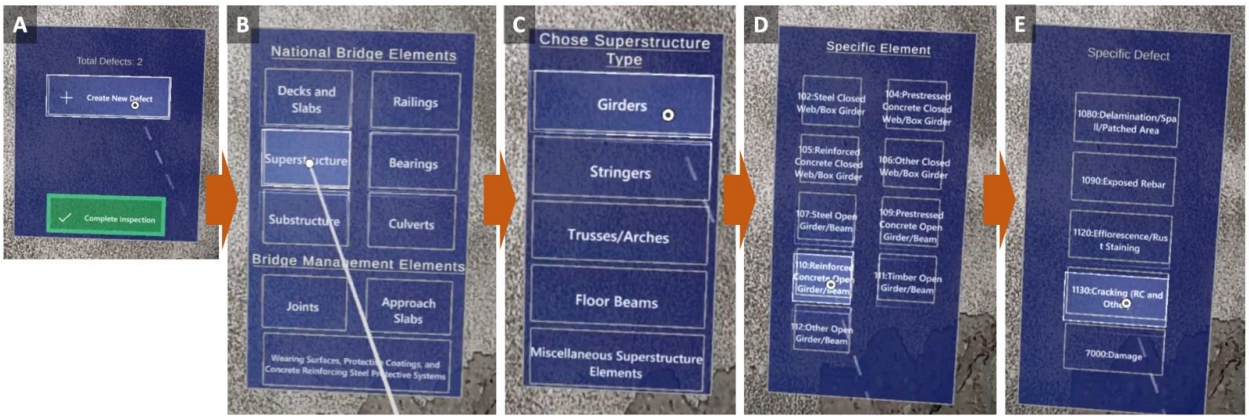

FIGURE 21 (A): Create new defect menu, (B): Selection Menu for Bridge Type, (C): Natural Bridge Element Menu, (D): Selection Menu for Specific Bridge Element Type, (E): Selection Menu for Specific Defect Types.

Defect Creation Menu

InspectAR streamlines defect documentation through a structured and intuitive workflow. After starting or resuming an inspection, users are presented with a menu offering two options: "Create New Defect" and "Complete Inspection" (see Figure 20 (A)) and Figure 21 (A)). The number of currently created defects is also displayed.

Selecting "Create New Defect" launches a guided annotation process. To mark a defect location, inspectors use a point-and-pinch gesture to ray cast onto the bridge surface, placing a visual marker (a blue orb; see Figure 20 (B)). These orbs persist across inspection sessions, serving as spatial

anchors to help inspectors track previously documented defects. Instructional text appears alongside the interface to guide inspectors through the next steps, improving usability and reducing uncertainty. This instructional content is designed to evolve over time based on inspector feedback.

The “Complete Inspection” button, also part of this menu, is used only after all defects have been added and fully documented. Pressing it marks the inspection session as complete, updates the bridge record, and returns the user to the main menu, preparing the system for a future inspection at that site.

Structural Metadata Input

For new defect entries, the system prompts inspectors to provide structural metadata about the defect’s location following SNBI and AASHTO conventions. The first menu (Figure 21 (B)) presents a categorized list of general bridge elements, such as Decks and Slabs, Railings, and Superstructure. Based on their selection, inspectors are guided to a secondary menu (see Figure 21 (C)) with more specific options. In this example, the inspector selects “Superstructure” as the bridge element and “Girders” as a Superstructure type, ensuring accurate classification of the defect within the system.

Once an inspector selects a general bridge element, the system presents a submenu of specific element options (Figure 21 (D)), numbered according to the AASHTO bridge inspection manual definitions. These options include elements such as “110: Reinforced Concrete Open Girder/Beam”, which is selected in the example.

After selecting the specific bridge element, the system displays a list of possible defects associated with that element, also as defined by AASHTO (see Figure 21 (E)). Each defect is labeled with its corresponding AASHTO numerical code, ensuring standardized and compliant documentation. In this case, the inspector selects “1130: Cracking (RC and Other).” The same back button remains available at this stage for correction if needed.

This structured, hierarchical selection process minimizes the need for manual code lookup, reduces errors, and ensures that defect codes align with the selected bridge element. As a result, inspectors can accurately classify defects within seconds, improving efficiency while maintaining compliance with established inspection protocols.

Data Panel

After specifying the defect metadata, the system displays a detailed data panel with multiple options tailored to general and specific information about the defect, including defect specific measurements such as crack width in our example (see Figure 22 (A)).

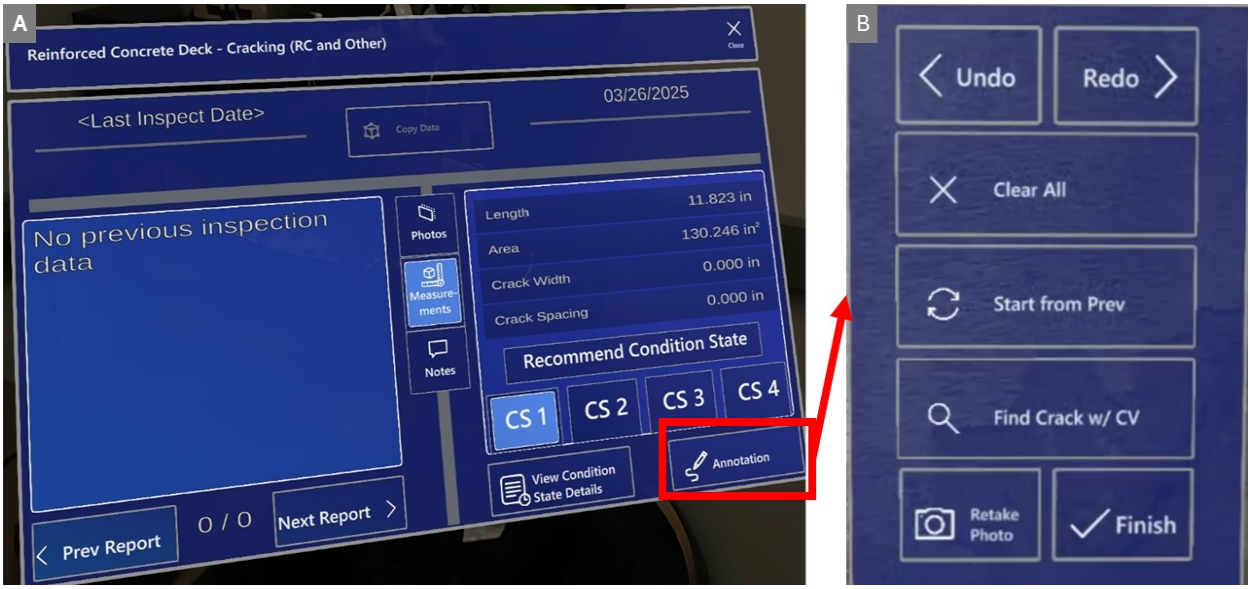

FIGURE 22 Data and Annotation Panels

The left side of the panel is dedicated to previous inspection data, providing inspectors with a reference for historical defect records. The right side is used for current inspection data, where inspectors document newly identified defects. When logging a new defect, only the right-hand panel is populated, while for previously recorded defects, both sides are populated for comparison.

At the top center of the panel, an optional copy button allows inspectors to transfer data from a previous inspection, serving as a baseline for new entries. In the middle section, inspectors can adjust the panel's mode using designated buttons, which modify the displayed information and available actions. These modes enable inspectors to efficiently view and manage (1) measurements, (2) photos, and (3) notes, ensuring comprehensive and structured defect documentation.

One of the most important decisions an inspector makes related to defect annotation is to specify a condition state (CS), which denotes the severity of the defect (CS1 = little to no damage, CS2 = minor, CS3 = moderate, CS4 = severe). The CS level can be selected directly in the data panel, however, in order to assist junior inspectors some additional tools have been added. The first is the "Recommend Condition State" button, which compares the measurement fields against AASHTO hard criteria to highlight a recommended CS level. The second is the "View Condition State Details" button, which provides both example images of the selected defect at various condition states as well as the written condition state criteria from the AASHTO manual.

While the inspector is free to make manual measurements and input them directly into the data panel via a virtual keyboard, they also have the option to leverage virtual annotation tools by

selecting the “Annotation” button on the bottom left. This brings up a new menu explained in the following section.

Annotation Menu

The Annotation Menu (see Figure 20 (C), 22 (B)) allows inspectors to annotate defects directly on the bridge surface using a ray-cast and pinch interaction. This hands-free workflow enables precise measurement of defect features such as crack length and area without relying on physical touch or manual data entry.

When annotation is initiated for a defect, the system first requires the inspector to capture a new photo, even if historical annotations exist. This image is used to generate a projection-aligned canvas plane, critical for spatial registration. After capture, the photo is previewed to allow for retakes in cases of blur or poor framing.

Once the image is confirmed, the system displays annotation instructions and prompts the inspector to draw a bounding box using point-and-pinch gestures. Points are ray cast onto the bridge surface and locked at the time of placement. The system fits a bounding box around the selected region, dynamically updating its dimensions and calculated area. Inspectors can add as many points as needed, with Undo and Redo options available to refine the selection. Once placed, points cannot be repositioned individually, simplifying the interaction and reducing input errors.

Inspectors may also choose to reuse annotations from previous inspections if they remain valid, avoiding the need to redraw. However, bounding box creation is required at least once per session to support measurement and projection alignment.

Once the bounding box is finalized, the inspector can trigger the “Find Crack” operation. This initiates CV processing for the selected region, which updates the defect’s crack width measurement and prepares the virtual crack visualization described in the following section.

Automated Inspector-Guided Crack Width Measurement and Virtual Display

Once triggered by the “Find Crack” button in the annotation menu, InspectAR’s automated crack width measurement system enables sub-millimeter accuracy from a distance—helping inspectors quantify defects even in hard-to-reach locations.

The process begins once the inspector defines a bounding box around the crack. Using custom-built coordinate transformation functions, the HoloLens 2 converts this bounding box from 3D world coordinates into pixel coordinates on the associated 2D image. The image, bounding box coordinates, and a scaling factor derived from the image metadata are then sent to a nearby server. On the server, the image is cropped to the region of interest and passed through a CV algorithm that segments the crack and calculates the maximum crack width. This result is returned to the HoloLens 2, which automatically populates the Crack Width field in the data panel with the calculated value.

Alongside the numeric result, InspectAR also renders a virtual crack visualization a spline that is projected back onto the bridge surface (see Figure 23) by reversing the original coordinate

transformation. This spline provides visual feedback on the crack extent detected by the CV algorithm and helps inspectors verify whether the correct region was captured, especially near the point of maximum width. However, the spline does not reflect crack thickness; it serves purely as a spatial indicator. Width is calculated separately and only shown as a numeric value in the interface.

FIGURE 23 Virtual Crack Visualization Overlaid on Bridge Surface Defect Image Based on CV Server Calculations

This combined visual and quantitative output supports both immediate inspection needs and longer-term defect monitoring in future visits, by serving as a reference to evaluate whether the crack has grown.

Export of Data from AR App to Support Creating Inspection Reports

As part of the UX process, the team spent approximately 2 hours with the inspection team discussing the transfer of field notes to the AASHTOWare Bridge Management (BrM) software. The discussion focused on how defect quantities were documented in the field notes and how these were then aggregated and added to BrM.

The team found that the District 1 team was very methodical in their note-taking. For example, for girders, the team separated the notes by span number, then beam number, then listed the individual defects and quantities on that beam. This was a very promising finding, because this organizational

style aligns well with the AR app's workflow and organizational principles. It is important to note that not all inspection teams follow this style, as gleaned from the team's previous conversations with other inspectors).

After taking notes in the field, the inspectors preferred to aggregate all the defect information into a spreadsheet first before entry into BrM. Based on this, the team decided that the best way to export the AR-collected bridge inspection would also be as a spreadsheet. While it is possible to export the AR-collected data directly to BrM format (for direct import), it would bypass the inspectors' internal quality control process and make modifications and corrections more difficult. In addition, inspectors stated that a majority of time in the office was spent aggregating field notes into the spreadsheet rather than inputting spreadsheet data into BrM.

When inspectors complete an inspection, the InspectAR app creates a final set of JSON files that contain all inspection data and which is housed on the HoloLens 2/Trimble XR10. Once the JSON files are migrated from the HoloLens 2/Trimble XR10 to a laptop server, a custom script can be run to convert JSON files to .csv files. See "A Standard Procedure for Performing Bridge Inspections Using AR Technology" Supplemental Document for details on how to setup and run the script.

Since the output format is a .csv file, it is directly importable into Excel, following the inspector-provided format shown in Table 1 below. Using the current version of the app, the collected data could be used to populate every column EXCEPT for the "Total Quantity column". This column, however, could easily be copied from BrM or a previous inspection as it does not change from inspection to inspection. Because data collected via the AR app is already organized by element and defect codes, the export program will simply add all the quantities associated with the corresponding element-defect-condition state combinations to generate the quantities shown in Figure 24.

To demonstrate this functionality, we generated a series of randomized JSON files designed to replicate the outputs produced by the HoloLens 2 application during a real bridge inspection. Each JSON file contained simulated defect data, including element identifiers, defect types, quantities, and condition states, carefully structured to match the expected HoloLens 2 output format. We created a diverse dataset spanning deck, superstructure, and substructure components to simulate an extended bridge inspection scenario.

These simulated files were then processed through the CSV generation script to verify that the system could correctly parse multiple entries, aggregate them, and output a complete table in the format inspectors are already familiar with, based on the model table provided on page 7 of the handout. By testing with randomized, field-like data, we confirmed that the process is robust against variability in the input and does not require manual correction after processing.

We anticipate that automating this step will significantly reduce the time inspectors currently spend aggregating results from field notes, freeing up more time for quality control activities or reducing

post-inspection processing time. As such, we believe this functionality represents the most direct and tangible "value add" we can demonstrate at this stage.

TABLE 1 AR app output format converted into .csv file. Note that the "Total" column data must be manually added, all other data are provided by InspectAR.

Element	Description	Category	Total	CS1	CS2	CS3	CS4
109	Pre-stressed Aashto girders	2	1500	1457	15	14	14
1110					15	14	14
12	Reinforced Concrete Deck	2	12080	11860	76	85	59
1080					21	52	26
1130					55	33	33
205	Reinforced concrete columns	2	12	5	4	0	3
1080					3	0	3
1130					1	0	0
215	Reinforced concrete abutments	2	80	60	8	8	4
1080					8	8	4
234	Reinforced concrete pier caps	2	120	98	9	1	12
1080					9	1	12
300	Strip seal expansion joints	2	84	60	7	2	15
7000					7	2	15
310	Elastomeric bearings	2	25	0	24	1	0
1000					8	1	0
1040					16	0	0
321	Reinforced concrete approach slabs	2	960	919	29	4	8
1080					29	4	8
330	Metal bridge railings	2	652	620	7	15	10
7000					7	15	10
515	Steel protective coating	2	4023	3948	57	12	6
1000					57	12	6
520	Concrete steel protective coating	2	12080	12050	25	2	3
1120					25	2	3
7369	Reinforced concrete wingwalls	2	48	20	12	10	6
1080					12	10	6
7370	Wire enclosed riprap	2	15674	15517	46	49	62
2350					46	49	62
7371	Standard guardrail	2	652	636	6	9	1
7000					6	9	1
7377	Concrete diaphragms	2	36	21	13	1	1
2230					13	1	1

InspectAR Field Architecture

We designed and tested a field architecture which supports computer vision-based crack documentation within the AR space while also addressing two key limitations of the bridge inspection setting: 1) inability of the HoloLens 2 AR platform to run large computer vision models and 2) lack of internet and access to cloud computing. The architecture is based on a server-client structure as shown in Figure 25. The Microsoft HoloLens 2 AR headset (the client) captures images of the physical bridge then sends these over a wireless *offline* network to a laptop (the server). The laptop, which has much greater storage and computational resources, both stores the images and processes them using a machine learning (ML) algorithm. Then, using the same network, it sends the ML results back to the HoloLens 2, which projects them onto the physical space. The results consist of a crack outline (crack spline) and the crack end points. Three physical hardware components are necessary to implement this architecture: 1) the Microsoft HoloLens 2, which is worn by the inspector, 2) a wireless router, for creating the offline network, and 3) a laptop.

During the project, we fine tuned the architecture to work as fluidly as possible in a field setting. The key main goals of the architecture are:

1. Ensuring that the Microsoft HoloLens 2 AR headset posed no significant safety concerns on-site
2. Selecting a sufficiently strong router such that both the router and laptop can be kept within a work vehicle, thus allowing the inspector to carry only the headset. This requires that the network is sufficiently powerful to cover the inspection site with enough bandwidth to send and receive images.
3. Ensuring that the projection of crack detection results is as stable as possible over long distances and multiple visits.

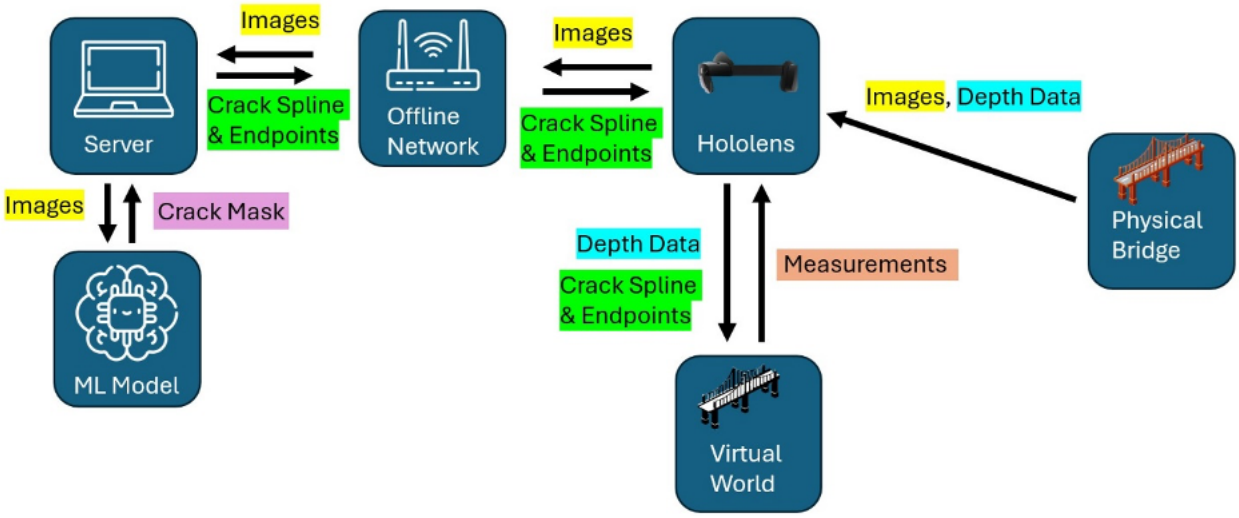

FIGURE 24 Final InspectAR Field Architecture for this Project Phase

InspectAR Data-Storage Schemas

Internal AASHTO-based Dictionaries

During the project, we also designed and developed a storage approach for an internal AASHTO-based dictionaries. These dictionaries represent bridge elements, defects and condition state information as defined by the AASHTO and can be altered/expanded to meet specific NMDOT needs. The dictionary tables (see Tables 2 – 7) are integrated with the AR app so that relevant data can be "pulled" from these tables in real-time. Thus, any updates/extensions to the tables results in real-time updates to InspectAR content and inspector options. Below we show the current state of each table in the dictionary:

TABLE 2 Table mapping bridge category to possible bridge elements (shown in AASHTO numbering system).

Bridge (Sub)Cat to Bridge Element Lookup
{"Decks and Slabs",new int[] { 12, 13, 38, 15, 16, 28, 29, 30, 31, 54, 60, 65 } },
{"Railings", new int[] { 330, 331, 332, 333, 334 } },
{"Girders", new int[] { 102, 104, 105, 106, 107, 109, 110, 111, 112 } },
{"Stringers", new int[]{ 113, 115, 116, 117, 118 } },
{"Trusses/Arches", new int[] { 120, 135, 136, 141, 142, 143, 144, 145, 146 } },
{"Floor Beams", new int[]{ 152, 154, 155, 156, 157 } },
{"Miscellaneous Superstructure Elements", new int[] { 147, 148, 149, 161, 162 } },
{"Bearings", new int[] { 310, 311, 312, 313, 314, 315, 316 } },
{"Columns/Pier Walls", new int[] { 202, 203, 204, 205, 206, 207, 208, 210, 211, 212, 213 } },
{"Abutments", new int[] { 215, 216, 217, 218, 219 } },
{"Piles/Pier Caps/Footings", new int[] { 220, 225, 226, 227, 228, 229, 231, 233, 234, 235, 236 } },
{"Culverts", new int[]{ 240, 241, 242, 243, 244, 245 } },
{"Joints", new int[]{ 300, 301, 302, 303, 304, 305, 306 } },
{"Wearing Surfaces, Protective Coatings, and Concrete Reinforcing Steel Protective Systems", new int[]{ 510, 515, 521, 520 } },
{"Approach Slabs", new int[]{320,321} }

TABLE 3 Table mapping bridge elements (shown in AASHTO numbering system) to a common name.

Bridge Element Number to Name
{203, "Other Column"},
{204, "Prestressed Concrete Column"},
{205, "Reinforced Concrete Column"},
{206, "Timber Column"},
{207, "Steel Tower"},
{208, "Timber Trestle"},
{210, "Reinforced Concrete Pier Wall"},
{211, "Other Pier Wall"},
{212, "Timber Pier Wall"},
{213, "Masonry Pier Wall"},
{215, "Reinforced Concrete Abutment"},
{216, "Timber Abutment"},
{217, "Masonry Abutment"},
{218, "Other Abutments"},
{219, "Steel Abutment"},
{220, "Reinforced Concrete Pile Cap/Footing"},
{225, "Steel Pile"},
{226, "Prestressed Concrete Pile"},
{227, "Reinforced Concrete Pile"},
{228, "Timber Pile"},
{229, "Other Pile"},
{231, "Steel Pier Cap"},
{233, "Prestressed Concrete Pier Cap"},
{234, "Reinforced Concrete Pier Cap"},
{235, "Timber Pier Cap"},
{236, "Other Pier Cap"},

TABLE 4 Table mapping bridge elements (shown in AASHTO numbering system) to relevant defects (also shown in AASHTO numbering system).

Bridge Element to Defect
{12, new int[] { 1080, 1090, 1120, 1130, 1190, 7000 } },
{13, new int[] { 1080, 1090, 1100, 1110, 1120, 1190, 7000 } },
{38, new int[]{ 1080, 1090, 1120, 1130, 1190, 7000 } },
{15, new int[]{ 1080, 1090, 1100, 1110, 1120, 1190, 7000 } },
{16, new int[]{ 1080, 1090, 1120, 1130, 1190, 7000 } },
{28, new int[] { 1000,1010,1020,7000 } },
{29, new int[]{ 1000,1010,1020,7000 } },
{30, new int[] { 1000,1010,1020,7000 } },
{31, new int[]{ 1020,1140,1150,1160,1170,1180,7000 } },
{54, new int[]{1020,1140,1150,1160,1170,1180,7000}},
{60, new int[]{ 1000,1010,1020,1080,1130,1220,7000 } },
{65, new int[] { 1000,1010,1020,1080,1120,1130,1220,7000 } }

TABLE 5 Table mapping all defects (shown in AASHTO numbering system) to common defect name.

Bridge Element Number to Name
{0, "" },
{12, "Reinforced Concrete Deck"},
{13, "Prestressed Concrete Deck"},
{38, "Reinforced Concrete Slab"},
{15, "Prestressed Concrete Top Flange"},
{16, "Reinforced Concrete Top Flange"},
{28, "Steel Deck with Open Grid"},
{29, "Steel Deck with Concrete Filled Grid"},
{30, "Steel Deck Corrugated/Orthotropic/Ect"},
{31, "Timber Deck"},
{54, "Timber Slab"},
{60, "Other Deck"},
{65, "Other Slab"},
{110, "Reinforced Concrete Open Girder/Beam"},
{111, "Timber Open Girder/Beam"},
{112, "Other Open Girder/Beam"},
{113, "Steel Stringer"},
{115, "Prestressed Concrete Stringer"},
{116, "Reinforced Concrete Stringer"},
{117, "Timber Stringer"},
{118, "Other Stringer"},
{120, "Steel Truss"},
{135, "Timber Truss"},
{136, "Other Truss"},
{141, "Steel Arch"},
{142, "Other Arch"},
{143, "Prestressed Concrete Arch"},
{144, "Reinforced Concrete Arch"},
{145, "Masonry Arch"},
{146, "Timber Arch"},
{152, "Steel Floor Beam"},
{154, "Prestressed Concrete Floor Beam"},
{155, "Reinforced Concrete Floor Beam"},

TABLE 6 Table mapping defect numbers as prescribed by AASHTO numbering system to common defect names. Note that the table has more rows than shown below.

Defect Number to Name
{1090,"Exposed Rebar"},
{1080,"Delamination/Spall/Patched Area"},
{6504,"Wheel Track Rutting (Asphaltic Plug Joint)"},
{1120,"Efflorescence/Rust Staining"},
{1130,"Cracking (RC and Other)"},
{7000,"Damage"},
{1190,"Abrasion/Wear (RC/PSC)"},
{1900,"Distortion"},
{6000,"Scour"},
{4000,"Settlement"},
{1100,"Exposed Prestressing"},
{1110,"Cracking (PSC)"},
{1140,"Decay/Section Loss"},
{1180,"Abrasion/Wear (Timber)"},
{1170,"Split/Delamination (Timber)"},
{1020,"Connections"},
{1150,"Check/Shake"},
{2310,"Leakage"},
{2320,"Seal Adhesion"},
{2370,"Metal Deterioration or Damage"},
{2330,"Seal Damage"},
{2340,"Seal Cracking"},
{2360,"Adjacent Deck or Header"},
{2350,"Debris Impact"},
{1010,"Cracking (Steel"},
{1020,"Connection"},
{1000,"Corrosion"},
{3440,"Effectiveness (Steel Protective Coatings)"},
{3420,"Peeling/Bubbling/Cracking"},
{3410,"Chalking"},

TABLE 7 Table showing mapping between a specific defect number as prescribed by AASHTO numbering system and the specific criteria used to determine condition state. The AR app provides this content on demand to inspectors as needed and is also used to recommend a condition state based on AR app measurements. Rows in yellow employ a numerical entry to recommend condition state, white rows are binary inputs (present/absent). As with other tables shown above, this table only shows some of the overall data in the full table.

Defect Number	Type of Entry ("num entry" or "toggle entry")		Condition State 1	Condition State 2	Condition State 3	Condition State 4
2220	toggle entry		Lateral and vertical alignment is as expected for the temperature conditions.	Tolerable lateral or vertical alignment that is inconsistent with the temperature conditions.	Approaching the limits of lateral or vertical alignment for the bearing but does not warrant a structural review.	The condition is beyond the limits established in condition state three (3) and/or warrants a structural review to determine the strength or serviceability of the element or bridge.
2210	toggle entry		Free to move.	Minor restriction.	Restricted but not warrant structural review.	Restricted but not warrant structural review.
2240	num entry	Enter % loss of bearing area	None.	Less than 10%.	10% or more but does not warrant structural review.	Restricted but not warrant structural review.
2230	num entry	Enter thickness & bulging amount	None.	Bulging less than 15% of the thickness.	Bulging 15% or more of the thickness. Splitting or tearing. Bearing's surfaces are not parallel. Does not warrant structural review.	Restricted but not warrant structural review.
3210	num entry	Enter spall depth and diameter	None.	Delaminated. Spall less than 1 in. deep or less than 6 in. diameter. Patched area that is sound. Partial depth pothole.	Spall 1 in. deep or greater or 6 in. diameter or greater. Patched area that is unsound or showing distress. Full depth pothole.	The wearing surface is no longer effective.
3220	num entry	Enter width and spacing	Width less than 0.012 in. or spacing greater than 3.0 ft.	Width 0.012-0.05 in. or spacing of 1.0- 3.0 ft.	Width of more than 0.05 in. or spacing of less than 1.0 ft.	The wearing surface is no longer effective.
3230	toggle entry		Full effective. No longer evidence of leakage or further deterioration of the protected element.	Substantially effective. Deterioration of the protected element has slowed.	Limited effectiveness. Deterioration of the protected element has progressed.	The wearing surface is no longer effective.
3230	toggle entry		Full effective. No evidence of leaking or further deterioration of the protected element.	Substantially effective. Deterioration of the protected element has slowed.	Limited effectiveness. Deterioration of the protected element has progressed.	The wearing surface is no longer effective.
3210	num entry	Enter spall depth and diameter	None.	Delaminated. Spall less than 1 in. deep or less than 6 in. diameter. Patched area that is sound. Partial depth pothole.	Spall 1 in. deep or greater or 6 in. diameter or greater. Patched area that is unsound or showing distress. Full depth pothole.	The wearing surface is no longer effective.
3220	num entry	Enter width and spacing	Width less than 0.012 in. or spacing greater than 3.0 ft.	Width 0.012- 0.05 in. or spacing of 1.0- 3.0 ft.	Width of more than 0.05 in. or spacing of less than 1.0 ft.	The wearing surface is no longer effective.

Organization of Data on HoloLens 2 AR Headset

As mentioned above, InspectAR's core UI features include an explicit "finish inspection" button that closes out the on-site inspection and allows for a set of quality checks (e.g., check to make sure all previous identified defects are attended to during this inspection). To finalize the robustness of this approach, we developed a file and folder structure to better support inspections that span AR system shutdowns, random hardware/software crashes, inspector lunch breaks, and inspections that may last more than one day.

Specifically, when the AR App is first launched, it looks into the HoloLens 2 local storage to see if there is a folder whose name ends in "InProgress". If such a folder exists, this data is loaded, and the inspector is first presented with a "Continue Inspection" button. If no such folder exists, the inspector is presented with a "Start New Inspection" button. Upon pressing this button, a new folder is created and named using the date, start time, and the "InProgress" suffix. Once an InProgress inspection is complete (i.e., inspector presses "Finish Inspection"), the folder name is updated to remove "InProgress" suffix, add a "Completed" suffix, and also add the end date and

end time (Figure 25). In cases where an inspection is completed in one pass, this information can be used to determine how long an inspection took to complete since the folder contains both the start and end times.

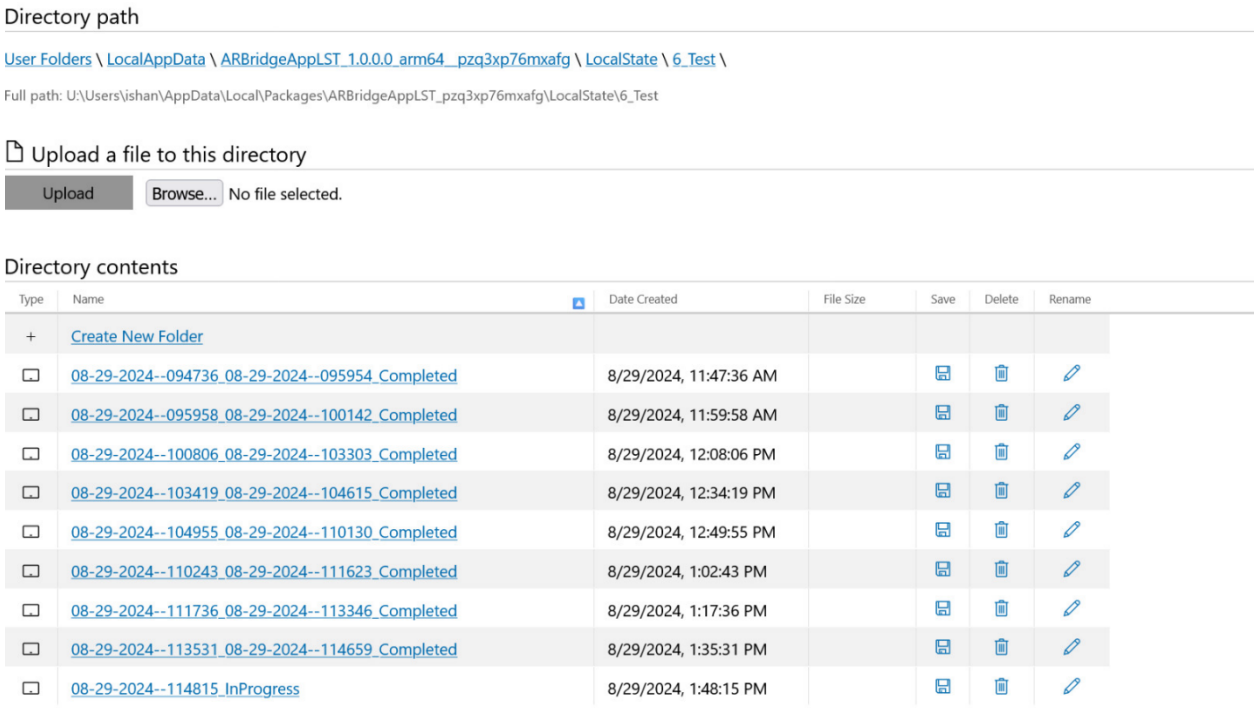

FIGURE 25 Folders on the HoloLens 2 showing Completed and InProgress inspection data.

Within each folder, there is a single JavaScript Object Notation (JSON) file that contains all the data from the inspection (Figure 26) -- see next section for details on the JSON file. Photos are currently sent to the laptop server for off-AR-display storage. Future versions of InspectAR would benefit from a functionality that would copy the JSON file over to the laptop server once an inspector chooses “Finish Inspection”.

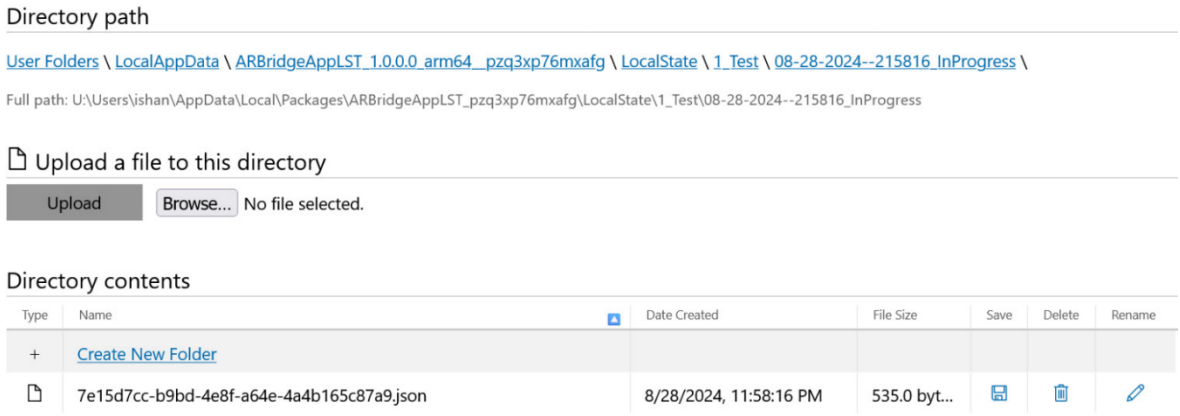

FIGURE 26 A JSON file within an InProgress inspection folder.
All inspection data for the current inspection is stored in this file.

Internal Organization of Data within App: Early in Year 2, we addressed how to best store data collected in the field by an Inspector using the AR App. Our goal was to create a robust data organization and storage solution that could provide fast real-time access to historical data, and also easy data interchange and usage outside of the AR App (e.g., to assist in reporting).

We organized the data using JavaScript Object Notation, or JSON; a lightweight data-interchange format that is easy for humans to read and write and is also easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures: (1) A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
(2) An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence. These are universal data structures. Virtually all modern programming languages support them in one form or another.

Figure 27 depicts is an example content of the JSON file from our August 29, 2024 demonstration:


```
{"sUID":"d5960988-69b8-432d-b3a4-4abfd75430a2","defectPoints":[
    {"x":-0.5667747855186462,"y":1.4484851360321046,"z":6.907707214355469},
    {"x":-0.5117009878158569,"y":1.4688291549682618,"z":7.0628767013549809},
    {"x":-0.34324225783348086,"y":0.8988879919052124,"z":6.887741565704346},
    {"x":-0.4068468511104584,"y":1.0325007438659669,"z":7.027364253997803}],
"defectNormal":{"x":0.5332107543945313,"y":0.14895489811897279,"z":-0.45711350440979006},
"elementCategory":2,
"bridgeElement":110,
"defectType":1130,
"defectPosition":{"x":-0.5585168600082398,"y":1.2951436042785645,"z":6.945657253265381},
"defectCategory":0,
"categoryInfo":[],
"storageDateTime":1724947044,
"length":2.1,
"height":1.6,
"conditionState":2,
"defectMetricsNumericList":[
    {"key":0,"value":7.675000190734863},
    {"key":11,"value":183.6230010986328},
    {"key":4,"value":0.009999999776482582},
    {"key":5,"value":1.0}],"
defectMetricsToggleSingleList":[]}
```


FIGURE 27 A sample of data from an inspection JSON file.

The JSON file contains the following information: The 3D position of a set of *defectPoints* that are placed by inspector (e.g., to denote end points of cracks, or crack branching points). The *defectNormal* which estimates the surface on which the defect is noted. The *elementCategory*, *bridgeElement* and *defectType* are based on AASHTO classification. The *defectPosition* specifies the 3D center of the colored Orb that provides a visual reference at a distance. *defectCategory* and *categoryInfo* are placeholders to further classify defects based on custom needs. The *storageDateTime* is self-explanatory. The *length* and *height* denote the size of the box that bounds the defect. The *conditionState* documents the condition state designated by the inspector. The *defectMetricsNumericList* captures the values associated with a specific defect (in this case 1130, Cracking (RC and other)), providing a reference to a key for each data field captured (e.g., crack width, crack spacing, etc.).

USER-CENTERED ASSESSMENT OF INSPECTAR

During the two-year process of designing and developing InspectAR, we undertook in parallel, a user experience (UX) process to ensure that the needs of bridge inspectors were front and center in all decision making. Our goal was to generate an AR-based system that would solve problems currently encountered in the field, while also adding new benefits unique to AR and head-worn displays' typical set of embedded sensors. Specifically, our UX work aimed to: (1) design and deliver an intuitive interface that mimics the traditional inspection workflow as closely as possible, (2) support structured data entry to match AASHTO component [1] and element level reporting requirements, and (3) facilitate the incorporation of automation tools (e.g. computer vision) while easily allowing manual fallback options in case of malfunction.

Our UX work was inspired by processes espoused in [2], which begins by understanding the core work and the associated work roles while also identifying barriers to efficient workflow and opportunities for improvement to inform iterative cycles of user interface design and AR/CV prototyping. We also employed user-based assessments in the field. In this section, we describe the UX work we undertook with actual bridge inspectors and bridge engineers.

UX Assessment Location

We collaborated with NMDOT personnel to identify target bridges in New Mexico to field our work, as well as a targeted bridge inspection use case. Nathaniel Pittman shared photos and bridge inspection reports from a few candidate bridges and shared these data with our team.

During a phone call, and subsequent emails, two options were discussed: (1) Bridge 8827 - located over an arroyo in Albuquerque, NM with low clearance, and (2) Bridge 9270 - unlimited access under the bridge without disturbance of traffic, owned by the city of Socorro, NM. Both bridges were considered viable by the team, with a preference for 9270. We ultimately decided on Bridge 9270.

We also defined a "critical use case" to center our research, design and app development efforts around. Specifically, we decided that our critical use case will afford:

- Overlay of previous inspection data

- AI-assisted updated crack measurement, and
- High priority crack(s) that require constant monitoring

We further confirmed with the team that Bridge 9270 has features consistent with the critical use case described above.

First UX Assessments at Socorro Bridge 9270

On August 29, 2024 we conducted a field demonstration that allowed bridge inspectors, district leads, and other personnel to use the AR App firsthand to annotate cracks in the aforementioned Socorro Bridge. This demonstration allowed stakeholders to better understand the promise of AR in actual bridge inspection settings, as well as allow the research team to get critical feedback on both current and future InspectAR functionality (Figure 28).

FIGURE 28 Photos of Stakeholders working with the AR App at Socorro bridge (top), along with first person AR-view of noted defects (middle), and crack measurement panel (bottom)

The following personnel participated in the field demonstration

- Edward Halbig, Bureau Chief, NMDOT
- Abigail Moya, Research and Climate Bureau, NMDOT
- Angelo Armijo, Research and Climate Bureau, NMDOT
- Nathaniel Pittman, Bridge Design Bureau, NMDOT

- Favio Casillas, District 1, NMDOT
- Raymond Munoz, District 1, NMDOT
- Erick Avila, District 1, NMDOT
- Ben Newman, District 2, NMDOT
- Justin Roybal, District 2, NMDOT

Results: The general feedback from those in attendance was very positive. Stakeholders reported that the interactions (e.g., pressing virtual buttons) were intuitive to use, and the visual interfaces made sense and could be interpreted. Indeed, one stakeholder in particular noted that he was not "tech savvy" but still found the AR App easy to use. Stakeholders reported seeing the value in using an AR App (as opposed to a tablet), and appreciated being able to take photos, document measurements and access historical data. Others mentioned the value of the condition state definitions and examples. Some stakeholders commented on the weight of the Trimble XR10 (integrated hardhat and HoloLens 2), which we expect will become lighter as the technology advances.

By observing bridge inspectors use the AR App, as well as through follow-on interviews with each bridge inspector, we identified a number of areas for future consideration, including:

- Provide interface mechanism to denote locations on bridge (e.g. Pier 1, Span 1, etc.).
- Support more body-relative user interfaces that "follow" inspectors around as they move.
- Identify a commercial quality router for field architecture (ongoing).
- Assess how using AR head-worn display effects safety (e.g., climbing ladder).
- Examine integration opportunities with BRM.
- Integration of small drones to afford inspection at a distance (e.g., AR through "eyes" of the drone).

Reflections on Field Demonstration Experiences and Discussions**:** Upon returning to Virginia Tech, the research team had a few lengthy conversations that aimed to articulate the notable insights from the trip. We identified the following three value-added goals to focus on since we believe an AR/CV based system can provide, along with how we might demonstrate these value-adds.

1. Provide spatial localization of inspection data (current and historical): Demonstrate via overlaying a virtual crack onto real crack, simultaneous presentation of current and historical data, and ease of access to historical defect data by simply selecting defects directly on bridge. We project that this capability would enable inspector more easily interpret previous inspections as well as track defect deterioration over time.
2. Generate time savings in the inspection-to-report production phase: Demonstrate via data export from the system such that the data can be viewed in, for example, Microsoft Excel. Once in Excel, data can be copied/pasted over into the reporting system. Photos taken onsite could also be exported for inclusion in the report.
3. Support remote measurement via CV: Demonstrate via a feature that allows inspectors to select the cracked area of interest from a distance of 3 to 6 feet, runs CV to detect/measure crack, then overlays crack length and thickness information onto the bridge.

Second UX Assessment at Socorro Bridge 9270

On February 20, 2025, we conducted a field evaluation that allowed bridge inspectors and other personnel to use again the AR App firsthand to annotate cracks in the aforementioned Socorro Bridge. This demonstration allowed the research team to get critical feedback on both current and future App functionality. In this visit, we showcased the ability to measure cracks using computer vision "at a distance". Inspectors also worked to manually measure cracks prior to using the AR App (see Figure 29s). On-site interactions with inspectors are critical to improving the UX of our AR App, and as such we followed each hands-on demonstration with one-on-one semi-structured interviews.

The following personnel participated in the field demonstration

1. Favio Casillas, District 1, NMDOT
2. Raymond Munoz, District 1, NMDOT
3. Erick Avila, District 1, NMDOT
4. Randy Trujillo, Bridge Design Bureau, NMDOT
5. Nathaniel Pittman, Bridge Design Bureau, NMDOT
6. Angelo Armijo, Research and Climate Bureau, NMDOT
7. Abigail Moya, Research and Climate Bureau, NMDOT
8. Tesha Henderson, Research and Climate Bureau, NMDOT

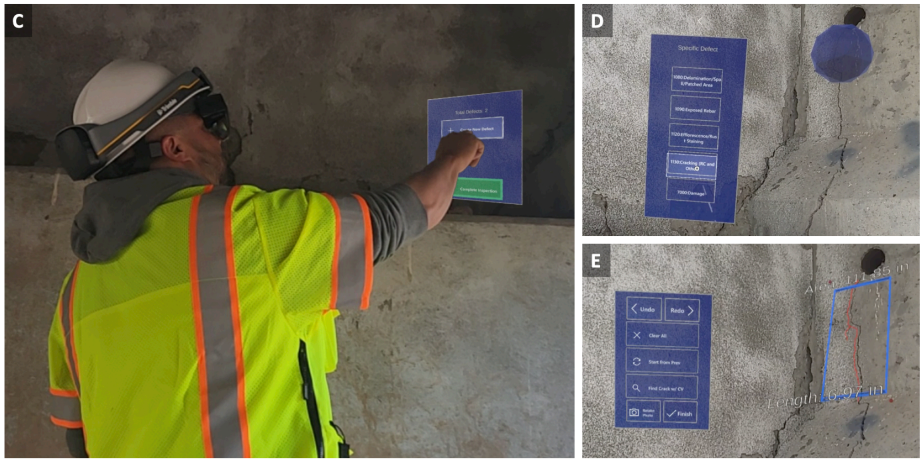

FIGURE 29 Photos of bridge inspectors working with the AR App at Socorro bridge ***(note that bottom-left image has been "PhotoShopped" to convey a third person AR view).***

Results: The general feedback from those in attendance was once again very positive. Stakeholders reported that the interactions (e.g., pressing virtual buttons) were intuitive to use, and the visual interfaces made sense and could be interpreted. Stakeholders reported seeing the value in using the CV-based approach, especially in hard-to-reach areas. As before, some stakeholders commented on the weight of the Trimble XR10 (integrated hardhat and HoloLens 2), which we expect will become lighter as the technology advances. We also noticed some hand-tracking problems with female stakeholders, that in hindsight could be have been due to rings on the stakeholders fingers. Generally speaking, the AR technical community is working diligently to update sensor- and image-based models to accommodate a wider range of persons – including height, hand-size, gender, skin-tone and more.

Results from both UX assessments, reinforced a core principle guiding our design of InspectAR: rather than imposing a rigid technological workflow onto inspectors, we aimed to build a flexible, assistive system that enhances their existing expertise. Our approach prioritized usability and adaptability, offering inspectors intuitive, context-driven interactions and tools designed explicitly for the practical challenges of fieldwork. InspectAR was intentionally developed as a hybrid tool, combining automated and manual capabilities, allowing inspectors to retain control over critical judgments and adapt their workflow to varying site conditions or personal preferences.

This philosophy informed key aspects of our application, such as hands-free interaction, spatially anchored annotations, and access to historical data. By closely aligning the AR application's functionality with how inspectors naturally perform tasks, we sought not to replace traditional inspection methods but to meaningfully augment and streamline them. The following section describes in detail how these design considerations were implemented in the final InspectAR application.

DISCUSSION & LESSONS LEARNED

The development of the InspectAR application represents a significant advancement in modernizing traditional infrastructure assessment processes. Our research and field evaluations demonstrated that AR, combined with CV, has the potential to enhance inspection efficiency, granularity, accuracy, and data management. However, the on-site demonstrations revealed several key challenges and insights related to the design and implementation of AR systems in the wild as well as the application of UX in occupational AR settings. While these findings apply to our bridge inspection work, we believe them to scale to other outdoor AR settings.

LAB DESIGN VERSUS FIELD DESIGN

Through this work, we learned that designing AR user interfaces (UI) for outdoor occupation is an endeavor that cannot be done primarily in a lab setting or with traditional user interface design applications (e.g., Figma). Outdoor environments are highly unstructured and it is simply impossible to capture large amounts of nuance in any structured setting – the devil is in the details. Instead, the design of fieldable and, more importantly, usable AR UIs is best done *in situ* with an applied and specific task as early as possible in the design process. In our case, it was not sufficient to define the AR task as a "bridge inspection" or even "defect annotation." Instead, we focused on the documentation and quantification of a very specific type of defect: reinforced concrete cracking. By doing so, we were able to build a much more precise mental model of how inspectors view cracks, which was very different from the mental model that the team had at the beginning of the project. For example, cracks appearing close together but on different structural elements could be classified and even measured differently, depending on the context. These nuanced observations eventually led to the design of the inspector-guided crack quantification algorithm as a way to leverage CV capabilities while strictly enforcing what areas of the structure should or should not be analyzed based on the inspector's judgement.

Along with providing nuanced observations, real-world testing exposes many issues than a lab testing setting simply misses. While lab tests can show promising results, the real-world outdoor environment revealed major differences in, for example, how well the CV pipeline performed. Factors like lighting changes, background textures, and the way in which inspectors took photos all affected the CV algorithm's accuracy in ways that weren't seen in the lab. Additionally, hardware issues—such as unreliable hand tracking and network connection drops—became much more noticeable when inspectors were actively trying to use the system in the field. These examples showcase that field testing is essential, as lab testing alone can give an overly optimistic view of performance. Thus, we argue that lab testing should be used to address specific problems that InspectAR in the field, not as a full development environment, or worse yet a proxy setting for meaningful evaluations.

WORKFLOW VERSUS TOOLS

To date, majority of research in the design of AR-based bridge inspection solutions has focused on what would generally be considered "tools," standalone applications designed achieve a specific function. These include things like real-time annotations, virtual measurements, information visualization, and others mentioned in the related works section. While promising, these tools

cannot be considered *useful* until they are integrated in an inspection "workflow," a continuous task sequence from arriving at the bridge to transferring notes into a digital database like AASHTOWare Bridge™. Note that this does not mean a complete inspection, per se, but rather any subset of tasks that would flow from one to the other in standard practice.

Our team was very deliberate in choosing the narrowest possible set of tasks that could still be considered a full workflow, focusing on the inspection of cracks on reinforced concrete elements. This led to some important considerations for the field experiments. First, the bridge site had to be carefully chosen to ensure that the workflow was possible (e.g., there were multiple instances of the target defect present and accessible). Secondly, the inspectors had to be guided to which defects to focus on, that is, it was not a "free" inspection environment. Thirdly, InspectAR still had to be designed with the appearance of being a complete inspection application to give inspectors an unbiased experience. In other words, the application menus presented options to document the full range of possible AASHTO elements and defects, despite the fact that inspectors were expected to select only specific options. In this way, it was easier to explain how the experimental workflow might be expanded to a broader set of defects in the future.

One of the advantages of the workflow approach was that it forced the team to maximize utility and minimize "traps" from an HCI perspective. For example, transitioning between different interaction contexts (e.g., world-relative vs. body-relative information) proved to be a significant design challenge. Inspectors frequently got trapped by losing cognitive association between virtual information and real-world referents, especially when switching between surfaces and floating menus. In some instances, users simply did not understand that there was a menu outside of their FOV. One proposed solution is the use of animated transitions to guide user attention, such as briefly highlighting relevant surfaces before fading away. Another issue arose with spatial positioning – placing UI elements in fixed locations did not always work well, as users are required to cover large distances (tens to hundreds of meters) during inspections. A hybrid approach, where elements remain body-relative but can be repositioned dynamically based on user preference, was suggested.

Also, thanks to the workflow approach, the team emphasized the importance of "graceful degradation" early on in the project. In our context, we took graceful degradation to mean providing the inspectors with fallback options in case some of the features of InspectAR did not work properly. This applied mainly to the use of CV as an automated means of crack quantification. Although convenient when it works, CV will invariably fail in certain scenarios; thus, the application allowed inspectors to easily retake photos or simply enter information manually. In fact, if inspectors did not like the CV option, they could avoid using it altogether.

AUTOMATION VERSUS ASSISTANCE

In concert with a successful workflow design, it is important to consider the degree of automation which optimizes the performance of the HCI system. In the broad world of technology-assisted bridge inspections, there is a strong interest in highly automated approaches, which by and large seek to *replace* certain functions of the inspector. However, there exist many challenges with these approaches, including their poor performance in unstructured and unsupervised conditions. We chose an assistive approach, which seeks to facilitate or augment certain inspector functions, but

within which the inspector retains some control. Even within this approach, however, there remains the question of *how much* control to give an automated algorithm, particularly with respect to defect detection and quantification.

We have explored this question in detail in previous work with support for hybrid approaches where the inspector can easily validate and modify the results of an automated algorithm. However, as a result of further field user evaluations and feedback, we have obtained further insights into this question. In particular, some users' optimistic perceptions of automation can make it difficult to comprehend the rationale behind certain design choices that limit automation. In our case, the design of a hybridized form of automation, where the inspector had to interactively specify the cracked region for automated processing, was initially interpreted as overly limited by some SMEs. In their minds, perhaps influenced by the salesmanship of certain AI companies, CV should be capable of scanning the entire bridge environment and highlighting all defects as the inspector walked an area. The low viability of such an approach did not become clear until the team demonstrated how an un-guided crack segmentation algorithm could be confused by graffiti or delamination into producing many false positives.

VIRTUALIZATION VERSUS ADDING VALUE

In an attempt to translate a manual workflow into AR, it can be tempting to virtualize every interaction in order to create a more uniform, digital experience. However, one of the greatest advantages of AR is that this is not necessary. It is important to be judicious about which tasks (and how) to virtualize in order to maximize the value added. We learned early in our contextual inquiry that inspectors often use spray paint to paint "dots" on the bridge to denote defect features e.g., maximum crack width location or the ends of cracks). They also generally inspect at an "arm's length" from the defect of interest. From this understanding, we designed an arm's length interaction technique for measuring cracks that required inspectors to touch the bridge surface to place and scale virtual dots as part of the measurement process. However, when demonstrating and then testing with SMEs, it became clear that this approach simply required more steps than the manual analogue (ruler) without improving accuracy, therefore not adding tangible value to the process. In response, the team pivoted away from the arm's length idea in favor of measurements at a distance, which are difficult for humans to perform accurately but can be accomplished via CV given adequate camera resolution. Being able to take measurements at a distance also alleviates many access requirements (e.g. deploying and climbing up a ladder) which take up a substantial amount of time in most inspections. This required a transition from touching to ray casting as well as from processing the entire FOV to only an area marked by the inspector. For arm's length (close-up) crack width measurements the application allows inspectors to take measurements with a ruler, then enter them into the interface. In summary, it was important to clearly distinguish between functions that the inspectors did well versus those that the AR did well and virtualize only the latter, even if it meant 1) transitioning from manual to digital data capture and 2) deviating from standard work practice. A great advantage of AR via HMDs is that not every interaction needs to be virtual.

TRAINING VERSUS INTUITION

Familiarity with and intuition for the AR environment is a major obstacle when evaluating fieldable AR applications. Even after an initial 30 minute training session on how to interact with the AR environment, inspectors still struggled with intuitive concepts that the research team largely took for granted. We observed that the expectation from inspectors (all first time AR users) was that they will see screen (not unlike a table or webpage) either fixed or floating in front of them. However, when AR UI elements are large or distributed across a large space (like a whole bridge!), the notion that one has to move their head around to "visually scan" the space in order to piece together the whole visual puzzle is not obvious. We are researchers understand the limited FOV gives us a small window into a larger view – so we know to pan our heads left and right to spatially view and integrate geometry, colors and textures into a cohesive understanding of virtual objects and/or scenes. Many inspectors would say "I see what looks like part of a thing, but I don't know what it is". In a similar vein, despite having been trained on AR gestures (e.g. point-and-pinch, open palm, button press), the HoloLens 2 struggled to capture their gestures as these were generally done very subtly, as if interacting with true physical objects. An experienced AR user intuitively understands that gestures should be exaggerated for best performance (e.g., poking by moving the hand rather than the finger). Many of the interventions during the field studies were aimed at correcting gestures.

In spite of these initial struggles, however, the team was surprised by the proficiency of inspectors who returned on the second bridge study (Phase 4), even though more that six months had passed since their last use of AR. This suggested to the team that there may be a relatively shallow learning curve for AR, and that many of the skills are intuitive and not memorized. Even so, the inspectors were not nearly as proficient with AR as the research team, and so it was not possible to evaluate the efficiency of their use of InspectAR relative to a manual inspection. In order to do so, it would be necessary for them to practice with the application for 510 hours over various days, until a solid level of conform is reached. This would be aimed at building an intuitive understanding of how to interact with the AR environment, rather than learning gestures or other skills, which can be picked up in a very short time (30 minutes or less). In general, our lessons suggest that teams building applications for novice users should strictly limit interactions to only the most basic gestures, excluding interactions like scrolling, grabbing, or even typing. This would help to minimize the time to proficiency and enable more meaningful comparisons to manual benchmarks.

LIMITATIONS AND FUTURE WORK

We developed and deployed InspectAR on a Microsoft HoloLens 2, approximately six years old at the time of this work. While it served our research needs, newer hardware could offer significant improvements for outdoor usage, such as enhanced cameras, more robust localization, better spatial registration, and longer battery life. Additionally, the combined setup of the HoloLens 2 with the Trimble XR10 proved bulky and heavy.

Throughout our UX research, we engaged bridge inspectors and engineers to gather representative insights. However, future work should include more diverse stakeholders, particularly inspectors from different transportation departments, to identify regional variations in inspection practices.

Our field evaluations were conducted a concrete bridge. Future studies should examine bridges of varying materials and scales to verify and extend insights gained during our contextual inquiry.

Similarly, usability studies were carried out under mild weather conditions with moderate cloud cover. Different environmental conditions (e.g., colder weather requiring gloves, strong sunlight affecting image quality for CV) could influence certain interface design aspects, such as larger touch targets and automatic image quality checks.

Currently, our computer vision implementation addresses only crack defects in concrete bridges. Although InspectAR's backend supports multiple defect types, additional CV capabilities should be developed and tested for critical defects such as spalls and delaminations. Finally, we plan to enhance InspectAR by integrating it directly with Bridge Management Software (*36*) via APIs or standardized XML imports.

CONCLUSION

The InspectAR application showcases the potential of augmented reality and computer vision in revolutionizing traditional bridge inspection methods. By digitizing the workflow while maintaining inspector autonomy, the system bridges the gap between manual assessments and automated defect detection. Our iterative development process, informed by direct user feedback, resulted in key enhancements that improve usability, accuracy, and efficiency.

REFERENCES

(1) U.S. Congress. United states code:Federal-aid highways, 23 u.s.c. §§ 101-144 https://www.loc.gov/item/uscode1970-006023001/, 1970. [Periodical] Retrieved from the Library of Congress.

(2) Federal Highway Administration. Specifications for the national bridge inventory. https://www.fhwa.dot.gov/bridge/snbi/snbi_march_2022_publication.pdf, March 2022. U.S. Department of Transportation.

(3) Federal Highway Administration. Fhwa bridge programs and data, 2025. Accessed: 2025-04-11

(4) G. A. Washer, M. M. Hammed, P. Jensen, and R. J. Connor. Quality of element-level bridge inspection data. Transportation Research Record, 2674(2):252–261, 2020.

(5) D. W. Yoo, H. Tarashiyoun, and M. Moghaddam. Modeling gaze behavior for real-time estimation of visual attention and expertise level in augmented reality. In 2023 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), pp. 487–492. IEEE, 2023

(6) Y. Okazaki, S. Okazaki, S. Asamoto, and P.-j. Chun. Applicability of machine learning to a crack model in concrete bridges. Computer-Aided Civil and Infrastructure Engineering, 35(8):775–792, 2020.

(7) A. A. Chettupuzha, B. Farooq, C. Y. Shih, J. L. Wright, S. Lee, S. Kiziltas, J. H. Garrett Jr, H. Karimi, and A. Smailagic. Design and evaluation of augmented reality interfaces for bridge inspection. In Proceedings of Joint International Conference on Computing and Decision Making in Civil and Building Engineering, 11th ICCCBE, June 14-16, 2006, Montreal, Canada, pp. 1477–1486, 2006

(8) A. Smith, C. Duff, R. Sarlo, and J. L. Gabbard. Wearable augmented reality interface design for bridge inspection. In 2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), pp. 497–501. IEEE, 2022.

(9) P. Ramakrishna, E. Hassan, R. Hebbalaguppe, M. Sharma, G. Gupta, L. Vig, G. Sharma, and G. Shroff. An ar inspection framework: Feasibility study with multiple ar devices. In 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct), pp. 221–226. IEEE, 2016.

(10) K. A. Kouch, K. Panuwatwanich, P. Sancharoen, and S. Sahachaisaree. Early view. 2020.

(11) I. John Samuel, O. Salem, and S. He. Defect-oriented supportive bridge inspection system featuring building information modeling and augmented reality. Innovative Infrastructure Solutions, 7(4):247,2022.

(12) C. D. Wickens, W. S. Helton, J. G. Hollands, and S. Banbury. Engineering psychology and human performance. Routledge, 2021

(13) M. J. Lazaro, Y. Kang, M. H. Yun, and S. Kim. The effects of visual complexity and decluttering methods on visual search and target detection in cockpit displays. International Journal of Human–Computer Interaction, 37(7):588–600, 2021.

(14) Z. A. Al-Sabbag, C. M. Yeum, and S. Narasimhan. Enabling human–machine collaboration in infrastructure inspections through mixed reality. Advanced Engineering Informatics, 53:101709, 2022

(15) C. E. Billings. Aviation automation: The search for a human-centered approach. CRC Press, 2018.

(16) L. H. Hansen, P. Fleck, M. Stranner, D. Schmalstieg, and C. Arth. Augmented reality for subsurface utility engineering, revisited. IEEE Transactions on Visualization and Computer Graphics, 27(11):4119–4128, 2021.
(17) Y. Zhou, H. Luo, and Y. Yang. Implementation of augmented reality for segment displacement inspection during tunneling construction. Automation in Construction, 82:112–121, 2017.
(18) F. Moreu, C. Lippitt, D. Maharjan, M. Aguero, and X. Yuan. Augmented reality enhancing the inspections of transportation infrastructure: Research, education, and industry implementation. 2019
(19) D. Maharjan, M. Agüero, D. Mascarenas, R. Fierro, and F. Moreu. Enabling human–infrastructure interfaces for inspection using augmented reality. Structural Health Monitoring, 20(4):1980–1996, 2021.
(20) D. D. Mascarenas, J. P. Ballor, O. L. McClain, M. A. Mellor, C.-Y. Shen, B. Bleck, J. Morales, L.-M. R. Yeong, B. Narushof, P. Shelton, et al. Augmented reality for next generation infrastructure inspections. Structural Health Monitoring, 20(4):1957–1979, 2021.
(21) Z. A. Al-Sabbag, C. M. Yeum, and S. Narasimhan. Interactive defect quantification through extended reality. Advanced Engineering Informatics, 51:101473, 2022.
(22) E. Karaaslan, U. Bagci, and F. N. Catbas. Artificial intelligence assisted infrastructure assessment using mixed reality systems. Transportation Research Record, 2673(12):413–424, 2019.
(23) S. Wang, S. A. Zargar, and F.-G. Yuan. Augmented reality for enhanced visual inspection through knowledge-based deep learning. Structural Health Monitoring, 20(1):426–442, 2021
(24) K. Malek, A. Mohammadkhorasani, and F. Moreu. Methodology to integrate augmented reality and pattern recognition for crack detection. Computer-Aided Civil and Infrastructure Engineering, 38(8):1000–1019, 2023
(25) A. Mohammadkhorasani, K. Malek, R. Mojidra, J. Li, C. Bennett, W. Collins, and F. Moreu. Augmented reality-computer vision combination for automatic fatigue crack detection and localization. Computers in Industry, 149:103936, 2023
(26) E. Bianchi and M. Hebdon. Development of extendable open-source structural inspection datasets. Journal of Computing in Civil Engineering, 36(6):04022039, 2022
(27) L. Deng, H.-H. Chu, P. Shi, W. Wang, and X. Kong. Region-based cnn method with deformable modules for visually classifying concrete cracks. Applied sciences, 10(7):2528, 2020
(28) X. Yang, H. Li, Y. Yu, X. Luo, T. Huang, and X. Yang. Automatic pixel-level crack detection and measurement using fully convolutional network. Computer-Aided Civil and Infrastructure Engineering, 33(12):1090–1109, 2018.
(29) G. Li, B. Ma, S. He, X. Ren, and Q. Liu. Automatic tunnel crack detection based on u-net and a convolutional neural network with alternately updated clique. Sensors, 20(3):717, 2020.
(30) Z. Zong, G. Song, and Y. Liu. Detrs with collaborative hybrid assignments training. arXiv preprint arXiv:2211.12860, 2022
(31) H. S. Munawar, F. Ullah, D. Shahzad, A. Heravi, S. Qayyum, and J. Akram. Civil infrastructure damage and corrosion detection: An application of machine learning. Buildings, 12(2):156, 2022.

(32) H. Jin Lim, S. Hwang, H. Kim, and H. Sohn. Steel bridge corrosion inspection with combined vision and thermographic images. Structural Health Monitoring, 20(6):3424–3435, 2021.
(33) R. Mojidra, J. Li, A. Mohammadkhorasani, F. Moreu, C. Bennett, and W. Collins. Vision-based fatigue crack detection using global motion compensation and video feature tracking. Earthquake Engineering and Engineering Vibration, pp. 1–21, 2023.
(34) American Association of State Highway and Transportation Officials. A Policy on Geometric Design of Highways and Streets. American Association of State Highway and Transportation Officials, Washington, D.C., 6th ed., 2011
(35) R. Hartson and P. S. Pyla. The UX book: Agile UX design for a quality user experience. Morgan Kaufmann, 2018.
(36) AASHTOWare. Aashtoware bridge, 2025. Accessed: 2025-04-11.