Interactive Defect Quantification Through Extended Reality

Zaid Abbas Al-Sabbag[a], Chul Min Yeum[a,*], Sriram Narasimhan[b]

[a]*Department of Civil and Environmental Engineering, University of Waterloo, Canada*
[b]*Department of Civil and Environmental Engineering, University of California, Los Angeles, USA*

Abstract

In this study, a new visual inspection method that can interactively detect and quantify structural defects using an *Extended Reality* (XR) device (headset) is proposed. The XR device, which is at the core of this method, supports an interactive environment using a holographic overlay of graphical information on the spatial environment and physical objects being inspected. By leveraging this capability, a novel XR-supported inspection pipeline, called *eXtended Reality-based Inspection and Visualization* (XRIV), is developed. Key tasks supported by this method include detecting visual damage from sensory data acquired by the XR device, estimating its size, and visualizing (overlaying) information on the spatial environment. The crucial step of real-time interactive segmentation—detection and pixel-wise damage boundary refinement—is achieved using a feature Back-propagating Refinement Scheme (f-BRS) algorithm. Then, a ray-casting algorithm is applied to back-project the 2D image pixel coordinates of the damage region to their 3D world coordinates for damage area quantification in real-world (physical) units. Finally, the area information is overlaid and anchored to the scene containing damage for visualization and documentation. The performance of XRIV is experimentally demonstrated by measuring surface structural damage of an in-service concrete bridge with less than 10% errors for two different test cases, and image processing latency of 2-3s (or 0.5s per seed point) from f-BRS. The proposed XRIV pipeline underscores the advantages of real-time interaction between expert users and the XR device through immersive visualization so that a human-machine collaborative workflow can be established to obtain better inspection outcomes in terms of accuracy and robustness.

Keywords: visual inspection, extended reality, augmented reality, damage detection

1. Introduction

1.1. Motivation

Large inventories of lifeline infrastructures supporting the water, energy, and transportation needs of our society are past or nearing the end of their intended service life and are deteriorating at an alarming rate. The potential failure of engineered structural components which comprise nearly all such infrastructures poses significantly increased risks to public safety [1], while simultaneously exacerbated by the effects of climate change. New structural inspection guidelines in many jurisdictions have begun to adopt quantitative assessment criteria to evaluate their condition, replacing or augmenting qualitative indicators (e.g., estimating condition based on partial or faint visibility, or tapping areas of concrete with a hammer to measure delamination) [2]. For example, the *AASHTO Manual for Bridge Element Inspection* now requires inspectors to identify the types, locations, and sizes of damage on bridge elements to support more objective and quantitative estimates for deterioration [3]. Such a shift necessitates technology that can support an inspector in the field to achieve an accurate and objective assessment and satisfy new inspection requirements.

*Corresponding author at 200 University Ave. West, Waterloo, Ontario N2L 3G1, Canada

Email addresses: zaalsabb@uwaterloo.ca (Zaid Abbas Al-Sabbag), cmyeum@uwaterloo.ca (Chul Min Yeum), snarasim@ucla.edu (Sriram Narasimhan)

Visual Inspection: Fully manual visual assessment remains the most widely used inspection method today. This typically involves an inspector visually observing the object and documenting various defects observed (e.g., surface cracks, spalling, corrosion, etc.). Measurements of defects, in this case, will typically involve the inspector physically accessing the area with a handheld measurement device. While this process can be simple and can produce the desired outcomes in many situations, especially when performed by an experienced inspector, often such methods are found to be inconsistent (across multiple inspectors and inspections), cumbersome, costly, and unreliable (because of incorrect measurements) [4]. New technology-driven standalone inspection methods supported by advances in computer vision, aerial and ground robots have tried to address such challenges [5, 6, 7, 8]. While these technologies provide the inspector with new means to gather and process condition data from sensors (cameras, lidars, etc.), they continue to rely on workflows that are mostly asynchronous and are often not trustworthy. For example, under challenging situations (e.g., poor lighting, shadows, etc.), we cannot expect to collect enough high-quality data to ensure reliable outcomes, and inference from the automated system alone may be unsatisfactory. Thus, the expertise and intelligence of the inspector should be incorporated into the process by adopting interactive processes (e.g., checking data quality, refining processed data). This paper aims to address this crucial aspect by implementing superior data collection, spatial mapping, and advanced computer vision algorithms so that a human-machine collaborative inspection workflow can be established.

XR technology: XR broadly describes a range of technologies that enable users to experience virtual content in either a simulated or real environment. XR encompasses *Augmented Reality* (AR), *Virtual Reality* (VR), and *Mixed Reality* (MR) [9]. AR enables visualization of virtual objects on video streams of real scenes and is widely used in handheld digital devices such as smartphones or tablets. However, interactivity between virtual objects, users, and the physical scene is limited because, in most cases, these devices do not have a depth perspective, and thus they cannot create a detailed spatial map of the real scene. Users also have no way to directly interact with 3D virtual objects, since most interactions in AR are often mediated through 2D touch screens. On the other hand, VR fully supports interactive environments between virtual objects and users through controllers or hand gestures, but this can only run in the virtual world and there is no interaction with the real-world scene. MR combines key elements of AR and VR to let users see and interact with virtual objects that are overlaid on the real environment and can enable immersive visualization of information as well as human-machine interaction [10, 11]. A key component of MR is the ability to digitize the user's environment by creating a spatial map of the scene to facilitate interaction between real and virtual objects and to replicate immersive visual effects such as occlusion or collision between virtual objects and the real scene. Examples of MR devices that are currently available include Microsoft's HoloLens 1 (HL1), HoloLens 2 (HL2), and Magic Leap One [12, 13, 14]. In this study, we used HL2 due to its intuitive user interface combined with advanced 3D spatial mapping capabilities of the physical environment which are crucial for the proposed activities.

1.2. Literature Review

Significant literature exists on the topic of alleviating access through the use of aerial drones and terrestrial mobile platforms equipped with vision cameras [15, 16, 17, 18, 19]. Similarly, automated detection of damaged areas from large volumes of images using Deep Convolutional Neural Networks (DCNNs) have also been studied extensively in the context of computer vision applications to this problem [20, 21, 22, 23, 24, 25]. Simultaneous Localization and Mapping (SLAM) algorithms [26, 27, 28, 29] enable inspectors to reconstruct digital 3D point cloud models of the environment to localize and quantify damage in structural elements [30, 31, 32]. However, a majority of these works do not address the human-machine collaborative aspect.

In this context, XR technologies have seen applications in medical, construction, manufacturing, education, entertainment, aerospace, and other fields [33, 34, 35, 36]. However, their use in vision-

based infrastructure inspection and structural health monitoring is relatively limited. In one of the early works on the subject, AR was proposed for the rapid assessment of earthquake-induced building damage [37]. AR was also studied as a means to annotate, organize and visualize images and metadata during inspections here [38]. Another research work [39] demonstrated the application of HL1 for bridge inspections by allowing users to measure physical distances between any two points on the structure. HL1 device has also been used to develop a human-infrastructure interface for inspectors to visualize and interact with real-time data received from low-cost smart sensors mounted on structures [40]. In a related work, a framework for MR enabled inspection system using AR/MR devices was recently presented in [41]. The authors demonstrate that their system can detect and segment damage using a semi-supervised SegNet-based DCNN, where users can modify the prediction threshold in real-time to improve segmentation results. However, the system was implemented on an AR device without spatial mapping or 3D sensing capabilities, limiting quantitative evaluation. All of these studies have addressed the collaborative aspects of XR technology. However, they did not harness the spatial mapping capabilities provided by XR devices—through SLAM—to localize damage in the real scene and quantitatively evaluate inspection regions with high accuracy, which can limit the potential usage of XR devices in the actual inspection process.

1.3. Objectives

In this study, a novel real-time visual interactive inspection system is proposed, which can detect and measure the area of surface damage enabled through XR (HL2), called *e**X**tended **R**eality-based **I**nspection and **V**isualization* (XRIV). XRIV allows inspectors to capture images of a structure, detect visual damage on a surface, estimate its area, and then visualize them as holographic objects overlaid on the scene. Data collection and analysis are integrated into a single synchronous on-site process. A DCNN-based interactive segmentation algorithm called feature Back-propagating Refinement Scheme (f-BRS) is utilized to segment the damage region on an image and refine the results interactively through user feedback. Once the damage region is segmented interactively, the 3D world coordinates of this region's boundary are estimated through back-projection using the pose of the camera and the spatial map created by the device, followed by area estimation.

The main novelty of this work is to show how discrete hardware and software components within XR, and advanced computer vision algorithms such as interactive segmentation, are customized and integrated into a single, end-to-end workflow to create an interactive visual inspection system by involving the expert user to obtain and refine real-time quantitative damage measurements. Integration toward enhancing interactivity between users and the system results in more accurate and efficient visual inspection in the field. Also, a novel procedure was devised and deployed to spatially match 2D image pixels to the 3D scene using a ray-casting algorithm, followed by estimating the physical size of the damage. We experimentally demonstrated an end-to-end visual inspection system by leveraging the immersive and interactive spatial mapping capabilities provided by XR and data-driven reasoning achieved by computer vision. We successfully established a fully collaborative environment for structural inspections and measurements of visual defects. Fig. 1 captures XRIV in action, where it creates an interactive environment for the user (through hand gestures) to measure the area of surface concrete spalling damage. Fig. 1a shows the user's point-of-view when using XRIV implemented on HL2, while Fig. 1b shows the same scene from a third-person view through a two-way interface on a connected Android smartphone.

This paper is organized as follows. First, Section 2 covers the technical background of the key XRIV components: interactive segmentation, spatial computation, and area computation. Section 3 then outlines how these components are integrated as part of the full system implementation and how computational tasks are distributed between the XR device and the server to achieve near real-time analysis. Section 4 shows the results of the several experimental tests that were conducted to validate the accuracy of the f-BRS algorithm for spalling segmentation, the spatial measurement accuracy of HL2, and to demonstrate the performance of XRIV in the field. Section 5 summarizes the key conclusions of

(a) (b)

Figure 1: The proposed XRIV system for damage area measurement enabled through the HoloLens 2. Image (a) was captured using the HoloLens 2, while image (b) was captured using Spectator View [42] from a connected Android smartphone that views the same scene as HoloLens 2 through a two-way interface. Note that the smartphone is only used to capture image (b) and is not required for XRIV.

the study.

2. Technical Background

2.1. Interactive Segmentation

Most vision-based methods for detecting and quantifying damage have utilized automated DCNN-based methods to classify images when damage may be present [20, 21, 25], or to perform automatic pixel-wise segmentation of the damage region [23, 24, 43]. Although the accuracy of these methods has dramatically improved over the years, there are still some challenges. For example, automated algorithms do not provide the user with a direct way to refine incorrect segmentation results. Any segmentation algorithm, no matter how accurate it is, will inevitably encounter ambiguous situations where its performance degrades. Even in routine settings, the boundary of damage is not always clear and varies depending on how damage is defined (e.g., concrete flaking vs. spalling). In such cases, human judgment and expertise can significantly improve the outcome and leverage input from users to remove false detection and to enhance the performance of segmentation algorithms. Therefore, XRIV leverages a state-of-the-art interactive segmentation algorithm, called f-BRS [44], to detect and segment damage in images captured using an XR headset. The proposed XRIV framework utilizes the holographic user interface and f-BRS to enable inspectors to be involved in the detection and segmentation process by being able to correct inaccurate results with minimal user interaction.

DCNN-based interactive segmentation algorithms fall under the umbrella of semi-supervised learning, which are machine learning algorithms that use a combination of labelled and unlabelled data for training or to perform predictions [45]. With prior training on unlabelled data, and with some limited labelling provided by the user, interactive segmentation can segment object classes without being trained on those classes. Users first select multiple points on the image, called seed points, inside (positive) and outside (negative) a target region of interest [46]. Seed points provide the DCNN with initial conditions to separate the foreground (damage region in this case) from the background. The seed points are used to produce positive and negative distance maps that encode information about where the points were selected and placed on the image. These maps are then concatenated to the image array and fed to the DCNN as multi-layered inputs. The output of the DCNN is a prediction map with confidence values ranging from 0 to 1 for every pixel in the input image. The confidence values represent the probability of how likely each of the pixels is included in the foreground, such as structural damage. The prediction map is then converted to a binary segmentation mask by thresholding the confidence values to become either 0 or 1.

An issue with DCNN-based interactive segmentation is that user-provided seed points are not always entirely consistent with output prediction maps. This is because a simple feed-forward DCNN model uses back-propagation to train the network to minimize overall segmentation error but does not constrain the prediction map to be consistent with seed points during the training phase [46]. For example, when the user selects a positive seed point at a certain location in the foreground of the image, the value of the prediction map at that location may be slightly less than 1, even though the user is confident that this location is part of the foreground. These small inconsistencies accumulate and increase the overall error of the segmentation, resulting in inaccurate boundary detection regardless of the number of seed points. A Back-propagating Refinement Scheme (BRS) solves this issue by running a backward pass through the network for each selected seed point to update the input distance maps and enforce an additional constraint to make the output prediction map consistent with the selected seed points' locations [47]. BRS produces prediction maps that are consistent with user-selected seed points, producing higher overall accuracy for segmentation masks with fewer seed points. However, BRS is a computationally expensive algorithm and may produce significant delay because it requires running a backward pass through the DCNN for every selected seed point.

To address this issue, f-BRS builds upon the concept of back-propagation refinement but utilizes backward passes to modify the values of feature maps of selected intermediate layers instead of modifying the inputs of the network. This means that f-BRS requires running backward passes from the output of the DCNN to the intermediate layers while skipping the initial layers for each pass. Thus, f-BRS can achieve the same level of accuracy as BRS but reduce computational cost per seed point because each backward pass is shorter. This makes f-BRS ideal for real-time interactive segmentation applications.

The segmentation model in XRIV builds on a pretrained f-BRS model that utilizes a ResNet-34 backbone pretrained on ImageNet, and a DeepLabV3+ decoder [48] containing the two intermediate convolutions layers that were trained on the benchmark Semantic Boundaries Dataset (SBD) [49]. Three f-BRS configurations are considered in this study (labeled A, B, and C), which differ based on the intermediate layers to which the back-propagation refinement is applied in the network. f-BRS-A applies the refinement scheme to the output from the ResNet backbone, f-BRS-B applies it inside the DeepLabV3+ decoder but before the first convolution layer, and f-BRS-C applies it before the second convolution layer. The performance of each configuration was experimentally compared in Section 4.1.

In addition to the variation in the network configuration, a key difference between the proposed and the original implementation [44] is in how seed points are used for segmentation. The original implementation was designed to be used on a static image to allow users to select one seed at a time by successively placing points on incorrectly segmented regions to improve the result. However, direct implementation of this process is not suitable for XR applications because processing each seed point is computationally inefficient, and the user would need to wait for the previous seed point to be processed before selecting the next seed point, which can become time-consuming if many seed points need to be selected to obtain a reasonably accurate result. Instead, the f-BRS algorithm was customized for real-time implementation in XRIV to allow users to select multiple seed points and then capture an image for conducting segmentation at once, instead of processing each seed point individually. This can speed up the process considerably without an associated penalty in the accuracy because image segmentation would only need to be run once unless the user decides to modify the segmentation result.

2.2. Spatial Computation

A key component of XRIV is spatial computation, which is the process of computing how points in the 3D world coordinate system are projected to the 2D pixel coordinate system of the built-in camera and vice versa, as shown in Fig. 2. The XR headset's built-in camera follows a pinhole camera geometry model and stores a 3×4 camera projection matrix (P) for each image or video. P contains camera intrinsics and the pose of the camera (obtained from SLAM in the device) synchronized to the time of image capture. Since the camera parameters are known (pro-

Algorithm 1 Ray-casting

Input

C: Ray origin point (Camera Optical Center)
V: Ray direction unit vector ($P^{+}x$)
M: Set of m triangles in spatial mesh

Output

X: Intersection point between ray and spatial mesh

$\lambda_{min} \leftarrow$ empty ▷ Initialize λ_{min} as a placeholder value
for triangle $i = 1, 2, \ldots, m$ in M **do** ▷ Iterate over triangle i joined by points A_i, B_i, C_i with normal N_i
  $\lambda_i \leftarrow \dfrac{(C - A_i) \cdot N_i}{V \cdot N_i}$ ▷ Find λ_i using ray-plane intersection equation
  if $\lambda_i > 0$ **then** ▷ Check if the intersection is in front of the camera
    $X_i \leftarrow C + \lambda_i V$ ▷ Assign candidate intersection point X_i
    $v_0 \leftarrow C_i - A_i$
    $v_1 \leftarrow B_i - A_i$
    $v_2 \leftarrow X_i - A_i$
    $\begin{bmatrix}\alpha \\ \beta\end{bmatrix} \leftarrow \begin{bmatrix}v_0 \cdot v_0 & v_1 \cdot v_0 \\ v_0 \cdot v_1 & v_1 \cdot v_1\end{bmatrix}^{-1} \begin{bmatrix}v_2 \cdot v_0 \\ v_2 \cdot v_1\end{bmatrix}$ ▷ Find barycentric coordinates α, β of X_i on triangle surface

    if $0 <= \alpha <= 1$ and $0 <= \beta <= 1$ and $0 <= \alpha + \beta <= 1$ **then** ▷ Check if X_i is inside triangle
      if $\lambda_i < \lambda_{min}$ **then**
        $\lambda_{min} \leftarrow \lambda_i$ ▷ Assign λ_i to λ_{min} if it is the lowest calculated value yet
      end if
    end if
  end if
end for
$X \leftarrow C + \lambda_{min} V$ ▷ Calculate intersection point closest to origin point using λ_{min}

vided by the manufacturer), the estimation is limited to the 6 DOF pose of the camera associated with an image, which includes 3D translation and rotation parameters. P can then be constructed using a known

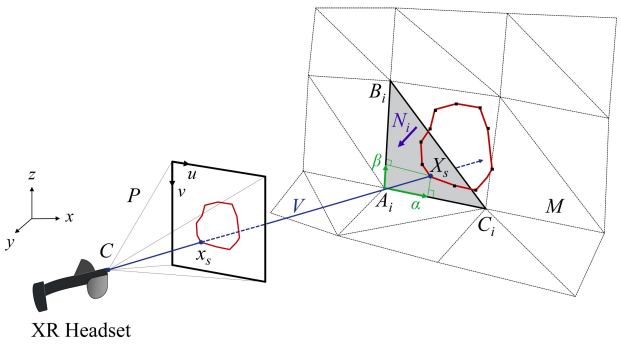


Figure 2: Pinhole camera geometry: Eq. (2) describes the projection of 3D world points to image points. A ray-casting algorithm in Algorithm 1 is used to back-project 2D image pixel coordinates of damage boundary to spatial mesh in 3D.

3×3 camera intrinsic matrix (K), a 3×3 camera rotation matrix (R), and a 3×1 camera translation vector (t), as presented in Eq. (1).

$$P = K \begin{bmatrix} R \mid t \end{bmatrix} \tag{1}$$

The 2D pixel coordinates (x_s) of points along the boundary of the segmented damage region on the image plane and the corresponding 3D points (X_s) in the world coordinates are related through the projective transformation in Eq. (2):

$$x_s = PX_s \tag{2}$$

Here, the 2D pixel coordinates and 3D world coordinates are represented in the homogeneous coordinate system as 3×1 vectors $(u, v, 1)^T$ and 4×1 vectors $(x, y, z, 1)^T$, respectively. This transformation is also used to project the world coordinates of user-selected

seed points (X_p) onto the image plane to obtain their image pixel coordinates (x_p).

The projective transformation in Eq. (2) allows X_s to be directly mapped to pixel coordinates on the image plane. However, since P is a non-square non-singular matrix, inverting this transformation yields non-unique solutions for X_s that correspond to x_s. A unique solution for X_s is obtained through back-projection, where image pixels are extended into the 3D world coordinates as vectors to determine associations, a process known as ray-casting. This process is used to anchor points on the spatial mesh of the 3D environment and to obtain the physical scale of the damage region for area computations.

The details of the ray-casting algorithm implemented in this paper are described in Algorithm 1. With reference to Fig. 2, the process of ray-casting locates the intersection point of the ray extended from the camera optical center (C) from a point on the image plane with the spatial mesh (M) constructed by the XR device. Suppose that the ray has an origin point at C along a certain ray direction vector (V) specified by an image point (x) and with an unknown distance parameter (λ), then the points on the ray (X) can be represented as a parametric line equation in Eq. (3) [50]:

$$X(\lambda) = C + \lambda V \quad \text{where} \quad V = P^{+}x \tag{3}$$

where P^{+} is the pseudo-inverse of P. M consists of a set of triangles in the world coordinates that are joined along their edges, as seen in Fig. 2.

The ray-casting algorithm takes C, V, and M as inputs and iterates over every triangle in M to determine an intersection point (X_i) between the ray and each triangle using a ray-plane intersection equation. Then, each X_i is checked to determine if it lies inside the triangle using a point-in-triangle test [51]. The point X_i is first converted to 2D barycentric coordinates (α, β) that indicate where the point is relative to the triangle's vertices, denoted as A_i, B_i and C_i. If α, β, and their sum value are between 0 and 1, then X_i is determined to be inside the triangle, which is the intersection point. However, a ray can intersect with M at multiple points since the real scene of M could be complex. In order to select a single output intersection point (X) while removing all other points that are either occluded or behind the camera, the X_i associated with the minimum λ (λ_{min}) greater than 0 is retained as the output. Therefore, every 2D image point (x) can be mapped to a unique 3D world coordinates point (X) on M using ray-casting.

Algorithm 1 is used to determine the world coordinates of each point along the segmented damage boundary by setting $x = x_s$ and $X = X_s$ for every image point of the boundary. Ray-casting is also used to determine the world coordinates of seed points (X_p) selected by the user through hand gestures using the XR interface. In this case, V in Algorithm 1 is determined through hand gesture, while C denotes the world coordinates of the wrist where the ray is assumed to originate.

2.3. Area Computation

Once the 3D world coordinates of the damage region boundary have been determined, the physical area of the bounded region is computed. To compute the areas in world coordinates, the general assumption is that the boundary lies on a planar surface with a single normal vector (N_s). However, due to the non-flatness of the mesh surface being analyzed as well as errors that are present in spatial mesh, the boundary points are not always placed on the same plane.

To overcome this issue, a best-fit reference plane is first determined by minimizing the least-squares error of the boundary points. The world coordinates of the boundary (X_s) are projected onto the reference plane with normal vector N_s to obtain a set of 2D planar coordinates (y_s) using a projective geometric transformation (T) by making one of three dimensions as zero. The points are first shifted to the centroid of the region by subtracting the average of the boundary points ($\bar{X}_s$) before pre-multiplying them by T in Eq. (4). The transformation is constructed by concatenating any two ortho-normal column vectors, denoted as n_u and n_v, that lie on the reference plane and are perpendicular to N_s.

$$y_s = T(X_s - \bar{X}_s) \quad \text{where} \quad T = \begin{bmatrix} n_u & n_v \end{bmatrix}^T \tag{4}$$

The set of 2D planar boundary coordinates, denoted as y_s, describe a closed polygon joined by k

number of points. The area of this polygon (A_s) is then calculated using the Shoelace formula [52] in Eq. 5, which is used to subdivide the polygon into k triangles and add up the areas of all of them. For every point (u_i, v_i) connected to the next point (u_{i+1}, v_{i+1}) in y_s, the area of a triangle connecting both points to the origin is calculated using a matrix determinant. A_s is then calculated by summing up the areas of all the triangles,

$$A_s = \frac{1}{2} \sum_{i=1}^{k} \begin{vmatrix} u_i & u_{i+1} \\ v_i & v_{i+1} \end{vmatrix} \quad (5)$$

3. System Implementation

The implementation schema of XRIV includes component subsystems where tasks are apportioned between the XR device and a computation server to allow for efficient use of available computing resources to enable real-time analysis. The user input/output interaction systems enabled through the XR interface allow users to interact with holographic objects such as the UI menu (shown in Fig. 1) using hand gestures and to visualize the segmented damage region and its area information through holographic overlays. Spatial computation using SLAM is executed on the XR device and the remaining heavy computational tasks including DCNN-based interactive segmentation and area computation are offloaded to a remote computational server. Here, the computational server can include, for example, cloud computing or a physical device such as a smartphone or a mobile computer. The detailed implementation pipeline of XRIV is described in Fig. 3. The interactive components of XRIV, namely locating damaged regions, the selection of the seed points, and interactively refining segmentation results to quantify damaged regions are illustrated in Fig. 4.

One of the key inputs for interaction is actualized in the component *Input User Interaction* in Fig. 3, namely selecting input seed points to detect damage on structures using interactive segmentation and to refine their results. First, the user selects positive and negative seed points from inside and outside the target region, respectively (Step 1 in Fig. 4a), to assist the DCNN with separating the foreground (damage region) from the background. Hand gesture recognition in the XR device allows the user to place the virtual seed points by interacting with the XR device's UI features. Upon receiving this input, the XR device then calculates the 3D world coordinates (X_p) of the seed points using a ray-casting algorithm, denoted *rayCast*, to anchor the points to the spatial mesh (M) (Algorithm 1). Then, the user captures an image (I) using the XR device's built-in camera that must show the full target damage region as well as all the seed points selected, initiating Step 2 in Fig. 4b. The 2D pixel coordinates (x_p) in I corresponding to X_p are then obtained by multiplying X_p by the corresponding projection matrix (P) (see Eq. 2 in Section 2.2).

Next, the XR device sends I and x_p to the computational server to segment the damage region using the f-BRS segmentation algorithm, denoted as *Segment* in Fig. 3 (Section 2.1). *Segment* takes x_p as input seed points to segment I and produces a binary mask (B) that indicates where the damage is present in I. A topological structural analysis algorithm, denoted as *findContours*, is used to extract contours from B to obtain the 2D pixel coordinates of the boundary (x_s) of the segmented region [53]. The 2D pixel coordinates, x_s, of the damage boundary extracted from I are then sent back to the XR device. Note that the computational offloading of *Segment* and *findContours* through the remote computation server in XRIV enables computationally expensive tasks to be undertaken in a remote machine rather than on a wearable device with limited computing and power resources.

The pseudo-inverse P^+ calculation and the back-projection of x_s as a set of continuous rays originating from the camera optical center (C) of I are executed on the XR device (denoted by *rayCast* in Fig. 3) to obtain a set of 3D world coordinates of the damage region boundary (X_s) corresponding to x_s. The XR interface then displays a holographic overlay of the target damage region and anchors it to M (red colored region in Fig. 4b).

In *User Output Interaction* in Fig. 3, the user evaluates whether the target damage region is properly segmented based on the holographic overlay. In some cases, damage regions might be over-or underestimated such as the one shown in Fig. 4b. This component allows the user to interactively improve

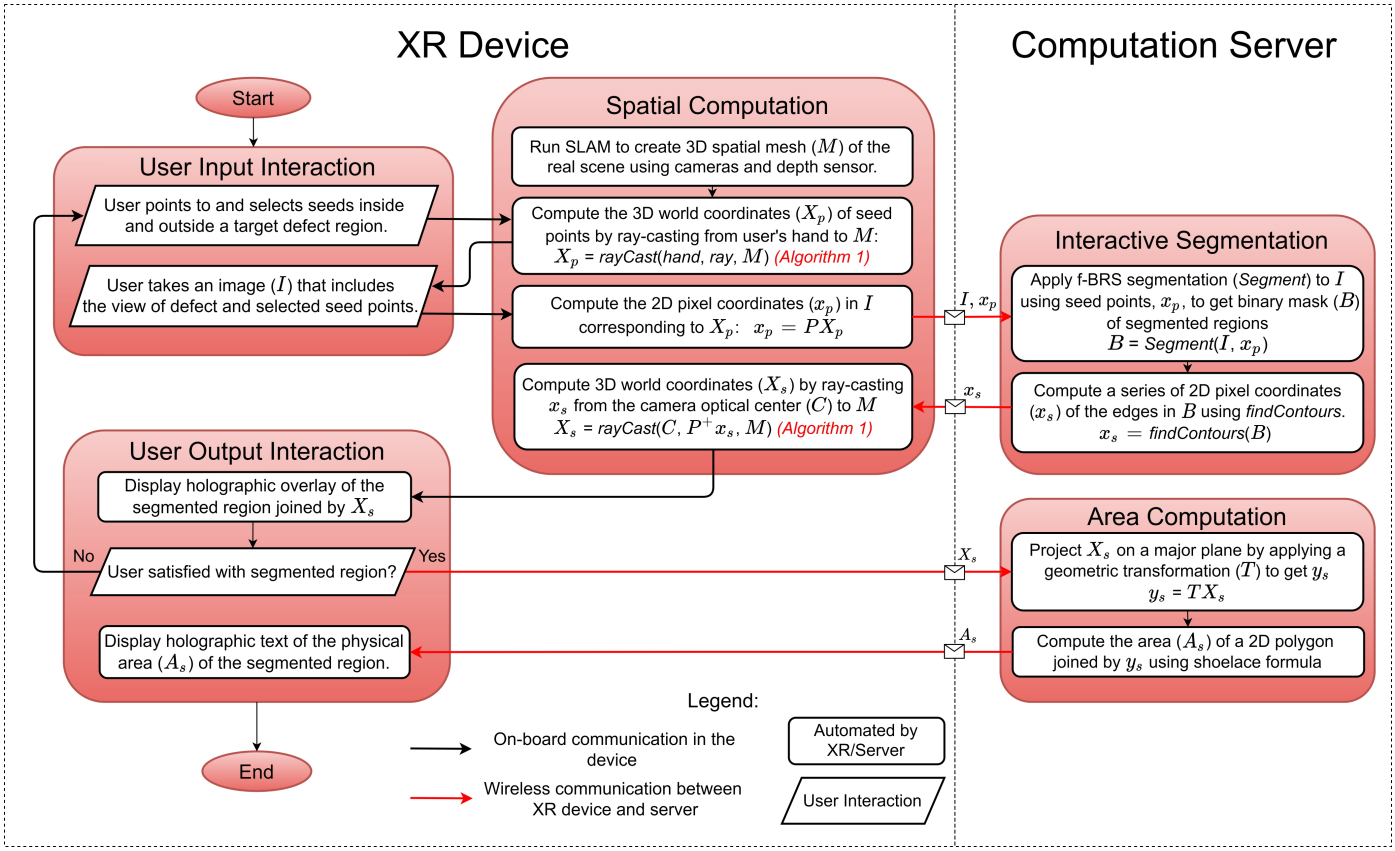
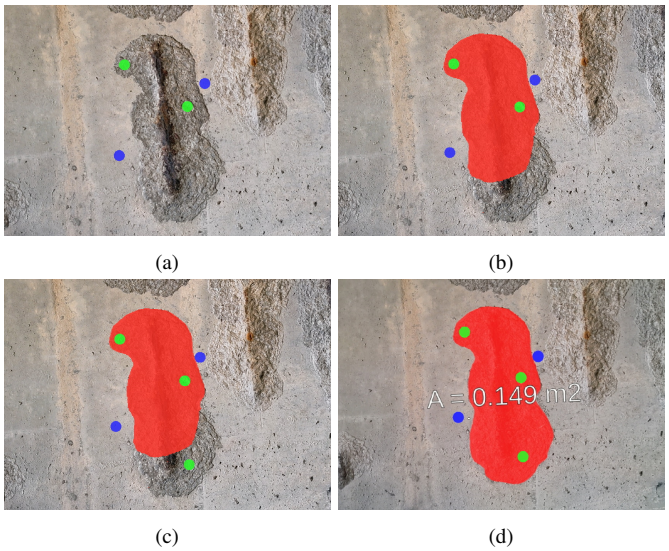


Figure 3: Real-time XRIV pipeline using XR device and computational server. Wireless communication is through HTTP requests.



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

Figure 4: XR interface for interactive damage segmentation, demonstrated on the HoloLens 2: (a) Step 1: select positive and negative seed points inside and outside damage region, (b) Step 2: apply interactive segmentation algorithm to the captured image, (c) Step 3: add seed points to refine segmentation results, and (d) Step 4: calculate the area of the refined segmented region.

the quality of segmentation, by adding more positive or negative seed points and then repeating the segmentation process (Step 3 in Fig. 4c). After obtaining a satisfactory segmented damage region, the user presses a button on the holographic menu (shown in Fig. 1) to obtain the physical area (A_s), which is mathematically represented as a closed polygon connected by X_s (see Section 2.3). Finally, a holographic text of the physical area of the target damage region is displayed in the XR interface and anchored to the target damage region (see Step 4 in Fig. 4d).

In addition to real-time analysis capability, XRIV also supports offline processing of these tasks. Such offline processing may be desirable to online processing, e.g., when several damage regions need to be inspected, or when a field inspection should be completed within a short period and the inspector prefers to analyze the data and monitor its quality at a later date. Although such an asynchronous offline mode does not facilitate in-situ real-time analysis, it allows for more flexibility in the selection of segmentation algorithms or the method of segmen-

tation. For example, the user can provide one seed point at a time while checking the quality of segmentation, rather than providing a set of seed points at once (explained in Section 2.1), or can implement other detection or segmentation algorithms.

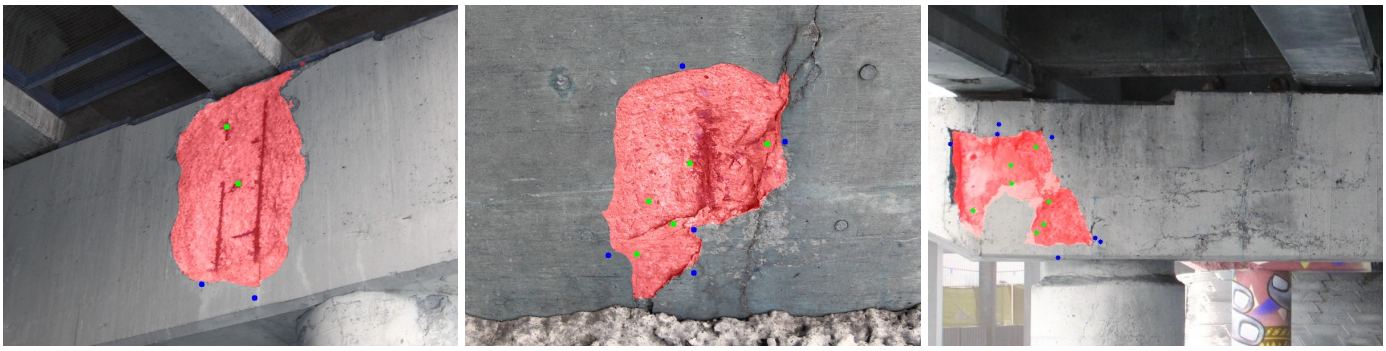
4. Experimental Results

4.1. f-BRS Segmentation Evaluation

To evaluate the performance of f-BRS for damage (spalling) segmentation, different implementations were tested by varying the seed points provided by the user. A total of 91 images of spalling damage were collected from in-service bridges in Southern Ontario and the boundary of the spalling damage was manually annotated. Three implementations of f-BRS and the original BRS were first compared. The original f-BRS study uses the evaluation metric of Intersection-over-Union (IoU) to measure segmentation error between the predicted and the ground-truth regions [44]. However, since the area measurement accuracy is a major concern rather than IoU in this paper, the evaluation metric chosen here is the area measurement accuracy, which is defined as the predicted area (A_P) as a percentage of the ground-truth area (A_G).

The first step was to randomly select a pair of positive and negative seed points which were placed inside and outside the ground-truth region, respectively. With these seed points, a predicted mask was generated and compared with the ground-truth region to determine the area measurement accuracy. In the next iteration, the set of positive and negative seed points were randomly selected in locations that were incorrectly segmented to generate a more accurate segmentation mask. This process was repeated for up to 12 pairs of seed points to incrementally refine the segmentation results. Fig. 5 presents sample segmentation results when the area measurement accuracy using f-BRS-B is over 80%. The positive (inside spalling region) and negative (outside spalling region) seed points used for the segmentation of each image are displayed over the segmentation masks.

The results of the experiment, presented in Fig. 6 show the average area measurement accuracy of all damage in images, and are computed depending on the number of seed points pairs. Overall, all implementations of f-BRS, as well as BRS, can segment spalling-type damage with 80% accuracy on average if five or more seed point pairs (10 seed points in total) are selected, which is satisfactory for vision-based inspection applications [2, 3]. Overall, f-BRS-B achieves slightly better accuracy among the f-BRS implementations and approaches a similar level of accuracy as BRS. Note that f-BRS was designed to improve the speed of the segmentation in BRS without significantly sacrificing accuracy. Therefore, the accuracy of BRS becomes an upper limit for f-BRS. For this experiment—performed on a computer equipped with a GPU: NVIDIA GeForce RTX 2060—f-BRS-B took approximately 0.73s on average per seed points pair to segment each image (1296×864 image resolution) in the dataset, while BRS took 3.32s. This means that f-BRS-B is ap-



(a) 2 pairs (b) 5 pairs (c) 7 pairs

Figure 5: Sample f-BRS-B damage image segmentation results when area measurement accuracy is above 80%, for 2, 5, or 7 pairs of positive (inside spalling region) and negative (outside spalling region) seed points pairs. Note that these images are illustrations and were not captured from an XR device.

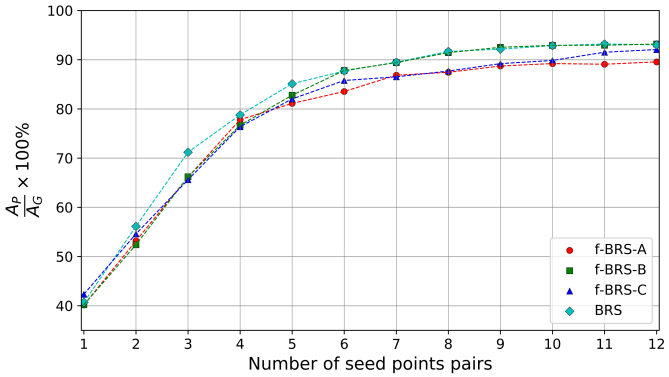


Figure 6: Evaluating area measurement accuracy as a function of the number of seed point pairs used for the segmentation models: f-BRS-A, f-BRS-B, f-BRS-C and BRS.

proximately 4.5 times faster than BRS while still maintaining a similar level of accuracy, which conformed to the findings of the original study [44]. This makes f-BRS-B the ideal implementation to choose for XRIV. Although f-BRS was developed to segment any visually distinct regions and its pretrained models were not trained on images having spalling-type damage, it can produce a reasonable performance for segmenting defects like spalling while maintaining a reasonable computation time. Note that since this experiment was performed using automated seed point selection, the accuracy of segmentation is lower than if we used human input to correct inaccurate segmentation results.

It should be noted that the number of seed points required depends on the unique situations, e.g., locations, shapes of damage regions, surface texture variation between damage and background, and ambiguous damage boundary (e.g., concrete spalling with flaking). This can be seen in Fig. 5. Thus, although we obtain over 80% accuracy using five pairs of seed points on average in this experiment, the ideal number of seed points required to obtain a satisfactory accuracy may vary depending on these conditions and the user's judgment.

4.2. HoloLens 2 Performance Evaluation

HL2, used as the XR device in this study, uses a SLAM algorithm to simultaneously build a spatial mesh of the user's environment while localizing the built-in camera's pose relative to the spatial mesh in real-time. The spatial computation component of XRIV, outlined in Section 2.2, relies on the XR device's SLAM system as an input to compute the physical size of segmented damage regions. Therefore, an experiment was designed and conducted to evaluate the performance limits of HL2, which is directly related to the performance of XRIV.

Most of the SLAM algorithms utilize either pure visual feature matching and tracking, or a fusion of visual features and 3D depth/point cloud information. For a comprehensive survey of SLAM algorithms, readers are referred to [26, 27, 28, 29] and SLAM applications for infrastructure [15, 30, 16, 18, 31]. Due to the proprietary nature of HL2, details of the SLAM algorithm are not available in the public domain or in their documentation.

In general, all SLAM algorithms suffer to some degree from localization and mapping errors which accumulate over time due to uncertainties in the sensor measurements [54]. Localization error is the error associated with the position of the sensor within the global map and the mapping error is the error in the map features compared to the real scene. Due to the tightly coupled nature of SLAM, disambiguation of these two sources of errors can be challenging and these sources contribute to each other. Hence, both these errors affect the overall performance of XRIV and in lieu of such previous documentation regarding its accuracy, a separate experimental validation of HL2's performance was necessary.

The experiment was designed to evaluate the measurement error of HL2 as a function of the working distance (distance between the camera and measurement target). Different scene lighting conditions were also evaluated. Two 12 cm x 12 cm fiducial (ArUco) markers with a 15 cm center-to-center distance were attached on indoor (Fig. 7a) and outdoor (Fig. 7b) walls to investigate the effect of illumination [55]. The measurements are made possible using the built-in infrared-based depth sensor (Microsoft Azure Kinect [13]), whose performance is known to degrade under direct sunlight due to infrared interference [56].

The procedure was to capture images using HL2 at various distances from the wall. Here, the working distance is defined as the distance between the headset's built-in camera and the markers. The markers were positioned on the wall so that they match the

(a) (b)

Figure 7: Evaluating the measurement accuracy of HL2 depending on a working distance under (a) indoor and (b) outdoor setting. Note that fiducial markers are used for automated detection.

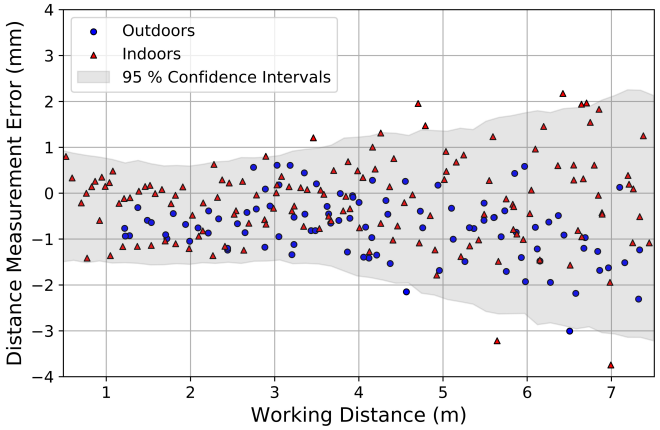


Figure 8: Scatter plot of distance measurement errors at different working distances between 0.5 m and 7.5 m under indoor and outdoor scene lighting conditions. Note that the gray area shows the 95% confidence interval (sliding window = 2 m).

same height as the headset's camera. Experiments were conducted at working distances ranging from 0.5 m to 7.5 m (device specifications limit this to a maximum of 10 m [13]). An ArUco marker detector implemented in OpenCV [55] was applied to each image to detect the four corners of each marker in the images. The 3D world coordinates of the corner points were obtained by ray-casting to the spatial mesh, as described in Section 2.2, and the center of each marker was computed from its four corner locations. Finally, the distance between the center points was computed, and the measurement error was obtained by subtracting the true distance, 15 cm. The working distance was computed from the average distance between the camera center (obtained from its camera pose) and the centers of the two markers. This test was repeated both indoor and outdoor.

Results of 236 trials are shown in Fig. 8 as a scatter plot, where each point represents the distance measurement error at a certain working distance in either indoor or outdoor lighting conditions. The gray area indicates a 95% confidence interval (computed using a sliding window with a 2 m working distance).

In Fig. 8, the graph shows that the average measurement errors and their spread increase as the working distances increase. We can qualitatively observe that the confidence interval of the data at the same confidence level widens when the working distance exceeds 3 m. This indicates that the sample variance of the data is dependent on the working distance and that the data may be heteroskedastic. A Breusch–Pagan test [57] is performed to check the null hypothesis that variance is independent of working distance. As a result, the null hypothesis was rejected with a p-value of 2.11×10^{-7}, which means that the variance increases as a function of working distance, which translates to the measurement accuracy dependence on working distance. This is not surprising because, as the working distances increase, the small-angle estimation errors in the camera matrix and/or the small offset of the spatial mesh can increase measurement errors, and the marker detection errors in pixels contribute to larger measurement errors.

Table 1 shows the mean error and spread at 95% confidence for both indoor and outdoor measurements, as well the aggregate of both cases. The mean errors for indoor and outdoor cases are very small (below 1 mm), which indicates that HL2's SLAM tracking system does not suffer from any significant drift regardless of brightness conditions and depth sensor interference from direct sunlight is not a significant factor for SLAM accuracy. The spread for both conditions is also relatively low (<2 mm), allowing consistent measurements. These experiments show that HL2 can produce sub-centimeter accuracy

Table 1: Aggregate statistics of distance measurement error under indoor and outdoor settings (95% confidence interval).

Location	Mean error ± spread
Indoor	0.16 ± 1.81 mm
Outdoor	0.76 ± 1.34 mm
Indoor & Outdoor	0.41 ± 1.73 mm

(a) (b)

Figure 9: Overview of test spalling regions at a bridge abutment (a) underneath bridge (SA) and (b) outside bridge (SB).

for both indoor and outdoor lighting conditions at working distances below 7.5 m. This level of accuracy is generally acceptable and sufficient for vision-based inspection applications [2, 3].

In this experiment, the difference in brightness between indoor and outdoor settings was compared qualitatively for the purpose of demonstrating that HL2 performs with reasonable accuracy under direct sunlight. Future studies can quantitatively measure the minimum and maximum amounts of brightness (measured in lumens) required to use HL2 with sufficient accuracy for common inspection scenarios where there may be either insufficient or excess amount of lighting.

4.3. Field Test

XRIV was tested at an in-service bridge in Southern Ontario, Canada. In this experiment, two spalling locations on the bridge were chosen from the abutment at inside (Fig. 9a) and outside (Fig. 9b) the testing bridge. Two spalling regions, labeled as SA and SB, have different sizes and approximately 0.5 m and 2 m in a vertical length, respectively. Both regions are present in locations that typically experience different lighting conditions. The working distance for each region was selected so that the entire region can be fit in a single image frame. SA and SB were captured from 1 m and 5 m away from the abutment, respectively.

The ground-truth areas of SA and SB were obtained first to be compared with the measurements

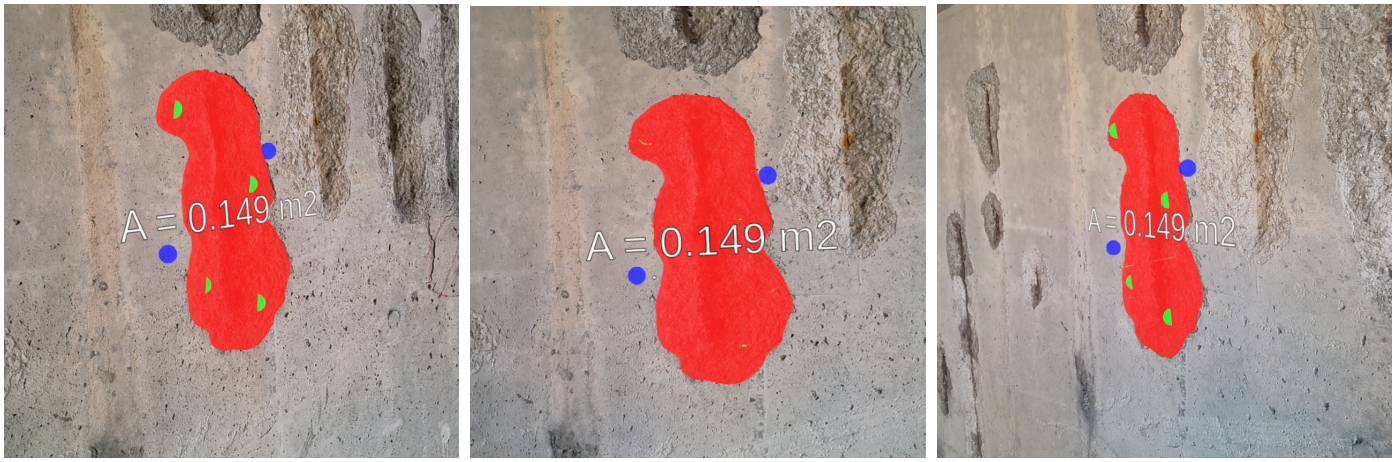
(a) (b)

Figure 10: Measurement of ground-truth spalling areas for (a) SA and (b) SB.

using XRIV. Since the spalling regions have irregular shapes and are often placed at inaccessible locations, like SB, an image-based approach was implemented to obtain their areas, instead of using other tools such as measuring tape. Fiducial markers with a known size were placed near the spalling region, and the entire view of the spalling region with the marker was captured as an image, shown in Fig. 10. Then, using a 4-point homography algorithm, a transformation matrix was obtained to transform all image pixel coordinates to corresponding physical coordinates [50]. The boundary of spalling on the images was manually segmented, which is shown as an outline in Fig. 10a and Fig. 10b for SA and SB, respectively. Then, the pixel coordinates of the boundary in the images were transformed to the physical scale coordinates, and then these coordinates were used

(a) (b)

Figure 11: Seed points selected on SA in (a) and SB in (b): The points inside and outside spalling regions indicate positive and negative seed points, respectively. These images were captured on the HoloLens 2.



(a)

(b)

Figure 12: Visualization of the segmented regions for SA in (a) and SB in (b) and their areas, captured on the HoloLens 2. Since graphic objects are anchored to a spatial mesh of the scenes, the holographic objects are rendered based on the user's viewpoints.

to calculate the area of each spalling region using the Shoelace formula explained in Section 2.3. The ground-truth areas of SA and SB are presented in Table 2.

The performance of XRIV was tested in both real-time and offline modes. For the real-time mode, seed points were selected and fed as inputs into the XR interface for interactive segmentation of spalling boundary (see Section 3). As discussed in Section 4.1, the number of initial seed points required to obtain a satisfactory result is variable and is based on how difficult it is to distinguish the damage region from the background, so it is up to the user to decide how many seed points are needed until a visually satisfactory result was obtained. Fig. 11 shows the scenes of the XR interface after selecting seed points from SA in Fig. 11a and SB in Fig. 11b. Then, the images including a full view of the spalling regions and the selected seed points were captured and sent to a mobile computer (equipped with a GPU: NVIDIA GeForce RTX 2060, which is the computational server outlined in Fig. 3) to segment the spalling. In XRIV, the user can adjust the placement and number of seed points through trial and error to obtain the best segmentation results. However, in this experiment, accurate boundaries for SA and SB could be obtained using six seed points (four inside and two outside spalling regions) without refining the process. Lastly, the areas of the segmented regions for SA and SB were computed, and a holographic overlay of the segmented region with a text overlay of the area was displayed to the XR interface, as shown for both spalling regions in Fig. 12. Note that the XR interface determines the shapes and positions of the graphic objects based on the user's headset position and eye gaze relative to the spatial mesh. Thus, the holograms including the color-filled region and the text look as if they are stationary and are part of the real scene.

In this experiment, HL2 was connected to the computer using a Wi-Fi hotspot. It took around 2-3 seconds (or 0.5s per seed) to run the GPU-accelerated f-BRS image segmentation on an image with 908×511 resolution when six seed points were provided. The speed of the segmentation in XRIV is comparable to the one reported in the original f-BRS study, which can be as fast as 0.32 seconds per seed point when a GTX 1080 Ti GPU was used [44]. Also, it took around 5-10 seconds for the user to select six seed points through the XR interface. The area computation runs in real-time, excluding the time taken to operate the holographic interface, which is around 1 second. Thus, it takes approximately a total of 8-14 seconds to measure the area of a single defect region in the real-time mode, unless the user refines the segmented region which may add a couple of seconds.

An offline mode was also used to estimate the area. Images of SA and SB were captured and stored on the device along with a spatial mesh of the scene. Then, the boundaries of SA and SB in the images were manually segmented, although automated segmentation algorithms could be used for this step. Basically, the measurement errors from the real-time and offline modes are identical if the segmented boundaries of the damage regions are equal in both modes. Thus, the measurement accuracy of both modes was ascertained from this testing to evaluate the error resulting from the interactive segmentation, assuming manual segmentation is the ground-truth. Finally, the areas of SA and SB were computed based on the procedure described in Section 2.3. The results of the experiment are presented in Table 2, which shows the damage size measurements for the real-time and offline modes, compared to ground-truth measurements.

Table 2: Damage size measurements for tested spalling regions using XRIV in real-time and offline modes, compared to ground-truth measurements.

Target damage	SA	SB
Real-time mode	0.149 m^2	1.245 m^2
Offline mode	0.143 m^2	1.176 m^2
Ground-truth	0.143 m^2	1.143 m^2
Error (real-time)	4.2%	8.9%
Error (offline)	<0.7%	2.9%

The results show that the real-time mode produces larger errors compared to the offline mode, which is expected due to the additional error caused by image segmentation since the offline results were segmented manually. Another result that can be observed is that the error for SB is larger than SA,

which is likely because the working distance for SB is larger than SA, which are approximately 5 m and 1 m, respectively. These results are consistent with HL2's performance evaluation experiment in Section 4.2 that shows that distance measurement is dependent on working distance. This experiment also shows that XRIV generally produces errors that are within 10% of the ground-truth, for both real-time and offline modes, which makes it suitable for vision-based inspection applications [2, 3].

5. Conclusion

In this study, a new vision-based damage quantification system, XRIV, using XR technology is presented. XRIV allows the user to accurately measure the size of remote damage on structures by leveraging the built-in sensors and spatial mapping capabilities of a wearable XR device. The novelty of XRIV is that the user can interact with the damage detection process in real-time and obtain quantitative damage information. A DNN-based interactive segmentation algorithm, f-BRS, was deployed to segment damage regions in images using seed points provided by the user through the XR interface. The physical size of the segmented damage region is geometrically evaluated using a ray-casting algorithm to obtain 3D coordinates of the damage regions from pixel coordinates. The performance of XRIV is experimentally demonstrated by conducting a field experiment at an in-service bridge in the case where spalling is the target damage. This system has been shown to successfully estimate the size of spalling damage with less than 10% errors for two different test cases with latency times of 2-3s (or 0.5s per seed point) due to image processing. The XRIV system demonstrates how XR devices can allow expert users to utilize human-machine collaboration to achieve desired outcomes such as improved accuracy, robustness, real-time analysis, and immersive visualization of data. It is expected that XRIV will become an exemplary implementation of XR technology for human-assisted visual inspection.

Acknowledgments

We acknowledge the support from Rogers and the Ontario Centers of Excellence via *Voucher for Innovation and Productivity Program* [OCE-34028] and Natural Sciences and Engineering Research Council of Canada (NSERC), [RGPIN-2020-03979].

References

[1] C. Mcnally, B. Ferreira, D. L. A. Gordon, The canadian infrastructure report card, Canada The State of the Federation 2015 (2015) 27–44.

[2] MTO, Ontario Structure Inspection Manual (OSIM), Ministry of Transportation Ontario, 2008.

[3] AASHTO, Manual for bridge element inspection, American Association of State Highway and Transportation Officials, 2019.

[4] M. Moore, B. M. Phares, B. Graybeal, D. Rolander, G. Washer, J. Wiss, et al., Reliability of visual inspection for highway bridges, volume i, Tech. rep., Turner-Fairbank Highway Research Center (2001).

[5] C. M. Yeum, S. J. Dyke, Vision-based automated crack detection for bridge inspection, Computer-Aided Civil and Infrastructure Engineering 30 (10) (2015) 759–770. doi:10.1111/mice.12141.

[6] B. F. Spencer Jr, V. Hoskere, Y. Narazaki, Advances in computer vision-based civil infrastructure inspection and monitoring, Engineering 5 (2) (2019) 199–222. doi:10.1016/j.eng.2018.11.030.

[7] E. McLaughlin, N. Charron, S. Narasimhan, Automated defect quantification in concrete bridges using robotics and deep learning, Journal of Computing in Civil Engineering 34 (5) (2020) 04020029. doi:10.1061/(ASCE)CP.1943-5487.0000915.

[8] D. Li, Q. Xie, X. Gong, Z. Yu, J. Xu, Y. Sun, J. Wang, Automatic defect detection of metro tunnel surfaces using a vision-based inspection system, Advanced Engineering Informatics 47 (2021) 101206. doi:10.1016/j.aei.2020.101206.

[9] P. Milgram, F. Kishino, A taxonomy of mixed reality visual displays, IEICE TRANSACTIONS on Information and Systems 77 (12) (1994) 1321–1329.

[10] R. K. Ganesan, Mediating human-robot collaboration through mixed reality cues, Ph.D. thesis, Ph. D. Dissertation. Arizona State University (2017).

[11] M. Khatib, K. Al Khudir, A. De Luca, Human-robot contactless collaboration with mixed reality interface, Robotics and Computer-Integrated Manufacturing 67 (2021) 102030. doi:10.1016/j.rcim.2020.102030.

[12] Microsoft, HoloLens (1st gen) hardware (2019).

[13] Microsoft, HoloLens 2 hardware (2020).

[14] Magic Leap, Magic Leap 1 (2019).

[15] A. Kim, R. M. Eustice, Real-time visual slam for autonomous underwater hull inspection using visual saliency, IEEE Transactions on Robotics 29 (3) (2013) 719–733. doi:10.1109/TRO.2012.2235699.

[16] H. Peel, S. Luo, A. Cohn, R. Fuentes, Localisation of a mobile robot for bridge bearing inspec-

tion, Automation in Construction 94 (2018) 244–256. doi:10.1016/j.autcon.2018.07.003.

[17] V. Hoskere, J.-W. Park, H. Yoon, B. F. Spencer Jr, Vision-based modal survey of civil infrastructure using unmanned aerial vehicles, Journal of Structural Engineering 145 (7) (2019) 04019062. doi:10.1061/(ASCE)ST.1943-541X.0002321.

[18] N. Charron, E. McLaughlin, S. Phillips, K. Goorts, S. Narasimhan, S. L. Waslander, Automated bridge inspection using mobile ground robotics, Journal of Structural Engineering 145 (11) (2019) 04019137. doi:10.1061/(ASCE)ST.1943-541X.0002404.

[19] S. Chen, D. F. Laefer, E. Mangina, S. I. Zolanvari, J. Byrne, Uav bridge inspection through evaluated 3d reconstructions, Journal of Bridge Engineering 24 (4) (2019) 05019001. doi:10.1061/(ASCE)BE.1943-5592.0001343.

[20] Y.-z. Lin, Z.-h. Nie, H.-w. Ma, Structural damage detection with automatic feature-extraction through deep learning, Computer-Aided Civil and Infrastructure Engineering 32 (12) (2017) 1025–1046. doi:10.1111/mice.12313.

[21] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhani, O. Büyüköztürk, Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types, Computer-Aided Civil and Infrastructure Engineering 33 (9) (2018) 731–747. doi:10.1111/mice.12334.

[22] C. M. Yeum, J. Choi, S. J. Dyke, Automated region-of-interest localization and classification for vision-based visual assessment of civil infrastructure, Structural Health Monitoring 18 (3) (2019) 675–689. doi:10.1177/1475921718765419.

[23] S. Li, X. Zhao, G. Zhou, Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network, Computer-Aided Civil and Infrastructure Engineering 34 (7) (2019) 616–634. doi:10.1111/mice.12433.

[24] B. Kim, S. Cho, Automated multiple concrete damage detection using instance segmentation deep learning model, Applied Sciences 10 (22) (2020) 8008. doi:10.3390/app10228008.

[25] M. Anitha, R. Hemalatha, S. Radha, A survey on crack detection algorithms for concrete structures, in: Advances in Smart System Technologies, Springer, 2021, pp. 639–654. doi:10.1007/978-981-15-5029-4_53.

[26] T. Taketomi, H. Uchiyama, S. Ikeda, Visual slam algorithms: a survey from 2010 to 2016, IPSJ Transactions on Computer Vision and Applications 9 (1) (2017) 16. doi:10.1186/s41074-017-0027-2.

[27] M. Filipenko, I. Afanasyev, Comparison of various slam systems for mobile robot in an indoor environment, in: 2018 International Conference on Intelligent Systems (IS), IEEE, 2018, pp. 400–407. doi:10.1109/IS.2018.8710464.

[28] B. Huang, J. Zhao, J. Liu, A survey of simultaneous localization and mapping, arXiv preprint arXiv:1909.05214 (2019).

[29] C. Debeunne, D. Vivet, A review of visual-lidar fusion based simultaneous localization and mapping, Sensors 20 (7) (2020) 2068. doi:10.3390/s20072068.

[30] Z. Shang, Z. Shen, Real-time 3d reconstruction on construction site using visual slam and uav, arXiv preprint arXiv:1712.07122 (2017). doi:10.1061/9780784481264.030.

[31] S. Jung, D. Choi, S. Song, H. Myung, Bridge inspection using unmanned aerial vehicle based on hg-slam: Hierarchical graph-based slam, Remote Sensing 12 (18) (2020) 3022. doi:10.3390/rs12183022.

[32] X. Chen, J. Li, S. Huang, H. Cui, P. Liu, Q. Sun, An automatic concrete crack-detection method fusing point clouds and images based on improved otsu's algorithm, Sensors 21 (5) (2021) 1581. doi:10.3390/s21051581.

[33] A. Webster, S. Feiner, B. MacIntyre, W. Massie, T. Krueger, Augmented reality in architectural construction, inspection and renovation, in: Proc. ASCE Third Congress on Computing in Civil Engineering, Vol. 1, 1996, p. 996.

[34] R. T. Azuma, A survey of augmented reality, Presence: Teleoperators & Virtual Environments 6 (4) (1997) 355–385. doi:10.1162/pres.1997.6.4.355.

[35] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani, M. Ivkovic, Augmented reality technologies, systems and applications, Multimedia tools and applications 51 (1) (2011) 341–377. doi:10.1007/s11042-010-0660-6.

[36] A. H. Behzadan, S. Dong, V. R. Kamat, Augmented reality visualization: A review of civil infrastructure system applications, Advanced Engineering Informatics 29 (2) (2015) 252–267. doi:10.1016/j.aei.2015.03.005.

[37] V. R. Kamat, S. El-Tawil, Evaluation of augmented reality for rapid assessment of earthquake-induced building damage, Journal of computing in civil engineering 21 (5) (2007) 303–310. doi:10.1061/(ASCE)0887-3801(2007)21:5(303).

[38] R. Napolitano, Z. Liu, C. Sun, B. Glisic, Combination of image-based documentation and augmented reality for structural health monitoring and building pathology, Frontiers in Built Environment 5 (2019) 50. doi:10.3389/fbuil.2019.00050.

[39] F. Moreu, B. Bleck, S. Vemuganti, D. Rogers, D. Mascarenas, Augmented reality tools for enhanced structural inspection, Structural Health Monitoring 2 (2017). doi:10.12783/shm2017/14221.

[40] D. Maharjan, M. Agüero, D. Mascarenas, R. Fierro, F. Moreu, Enabling human-infrastructure interfaces for inspection using augmented reality, Structural Health Monitoring (2020). doi:10.1177/1475921720977017.

[41] E. Karaaslan, U. Bagci, F. N. Catbas, Artificial intelligence assisted infrastructure assessment using mixed reality systems, Transportation Research Record 2673 (12) (2019) 413–424. doi:10.1177/0361198119839988.

[42] Microsoft, MixedReality-SpectatorView (2019).

[43] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3431–3440.

[44] K. Sofiiuk, I. Petrov, O. Barinova, A. Konushin, f-brs: Rethinking backpropagating refinement for interactive segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8623–8632.

[45] J. E. van Engelen, H. H. Hoos, A survey on semi-supervised learning, Machine Learning 109 (2) (2020) 373–440. doi:10.1007/s10994-019-05855-6.

[46] N. Xu, B. Price, S. Cohen, J. Yang, T. S. Huang, Deep interactive object selection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 373–381.

[47] W.-D. Jang, C.-S. Kim, Interactive image segmentation via backpropagating refinement scheme, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5297–5306. doi:10.1109/CVPR.2019.00544.

[48] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801–818. doi:10.1007/978-3-030-01234-2_49.

[49] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, J. Malik, Semantic contours from inverse detectors, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 991–998. doi:10.1109/ICCV.2011.6126343.

[50] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003. doi:10.1017/CBO9780511811685.

[51] C. Ericson, Real-time collision detection, CRC Press, 2004. doi:10.1201/b14581.

[52] Y. Lee, W. Lim, Shoelace formula: Connecting the area of a polygon and the vector cross product, The Mathematics Teacher 110 (8) (2017) 631–636. doi:10.5951/mathteacher.110.8.0631.

[53] S. Suzuki, et al., Topological structural analysis of digitized binary images by border following, Computer vision, graphics, and image processing 30 (1) (1985) 32–46. doi:10.1016/0734-189X(85)90016-7.

[54] R. Mur-Artal, J. D. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Transactions on Robotics 33 (5) (2017) 1255–1262. doi:10.1109/TRO.2017.2705103.

[55] Itseez, Open source computer vision library, `https://github.com/itseez/opencv` (2020).

[56] M. R. Andersen, T. Jensen, P. Lisouski, A. K. Mortensen, M. K. Hansen, T. Gregersen, P. Ahrendt, Kinect depth sensor evaluation for computer vision applications, Tech. rep., Aarhus University (2012).

[57] T. S. Breusch, A. R. Pagan, A simple test for heteroscedasticity and random coefficient variation, Econometrica: Journal of the Econometric Society (1979) 1287–1294doi:10.2307/1911963.