Enabling Human-Machine Collaborative Inspections through Smart Infrastructure Metaverse

by

Zaid Abbas Al-Sabbag

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Civil Engineering

Waterloo, Ontario, Canada, 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Gunho Sohn
Associate Professor, Dept. of Earth and Space Science and Engineering,
York University

Supervisor(s): Chul Min Yeum
Assistant Professor, Dept. of Civil and Environmental Engineering,
University of Waterloo

Sriram Narasimhan
Professor, Dept. of Civil and Environmental Engineering,
University of California, Los Angeles

Internal Member: Giovanni Cascante
Professor, Dept. of Civil and Environmental Engineering,
University of Waterloo

Internal Member: Carl Haas
Professor, Dept. of Civil and Environmental Engineering,
University of Waterloo

Internal-External Member: William Melek
Professor, Dept. of Mechanical and Mechatronics Engineering,
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Zaid Al-Sabbag was the sole author for Chapters 1, 2, and 6 which were written under the supervision of Dr. Chul Min Yeum and Dr. Sriram Narasimhan and were not written for publication.

This thesis consists in part of three manuscripts submitted for publication. Exceptions to sole authorship of material are as follows:

Research presented in Chapters 3, 4, and 5:

Dr. Chul Min Yeum and Dr. Sriram Narasimhan were the primary co-investigators on the research project funded by Rogers Communications, the Ontario Centers of Excellence via the Voucher for Innovation and Productivity Program, and the Natural Sciences and Engineering Research Council of Canada (NSERC). They are co-authors on any publications relating to this work. Zaid Al-Sabbag performed all the experiments, analysis, and wrote the draft manuscripts, which all co-authors contributed intellectual input on.

Citations:

Chapter 3: Zaid Abbas Al-Sabbag, Chul Min Yeum, and Sriram Narasimhan. Interactive defect quantification through extended reality. *Advanced Engineering Informatics*, 51:101473, Jan 2022.

Chapter 4: Zaid Abbas Al-Sabbag, Chul Min Yeum, and Sriram Narasimhan. Enabling human–machine collaboration in infrastructure inspections through mixed reality. *Advanced Engineering Informatics*, 53:101709, Aug 2022.

Chapter 5: Zaid Abbas Al-Sabbag, Chul Min Yeum, and Sriram Narasimhan. Distributed Collaborative Inspections through Smart Infrastructure Metaverse. *Automation in Construction*, AUTCON-D-23-02332R1, Aug 2023.

Abstract

Over the last ten years, novel Artificial Intelligence (AI) based structure assessment methods and tools have been proposed to identify and quantify structural damage indicators (e.g., cracking, spalling, corrosion, etc.) from visual data (e.g., images, LiDAR). However, despite an urgent need for such technology in managing infrastructure, widespread adoption of these technologies in the field has been quite limited. One of the main obstacles is the lack of real-time communication and interaction between the human inspectors, on- and off-site, and the technologies being deployed to support data collection, processing, and decision-making. This thesis focuses on enabling real-time remote collaborative structural inspections by integrating on-site inspectors, remote experts (e.g., engineers, stakeholders, etc.) responsible for making critical decisions, advanced data collection platforms (e.g., ground robots, drones, etc.), and AI algorithms that can rapidly interpret data, into an automated inspection system that supports human-machine collaboration. The motivation is to solve the technical and scientific challenges that prevent real-time collaboration between human users (on-site inspectors, remote experts) and machine agents (robots, AI).

This thesis proposes a system called the Smart Infrastructure Metaverse (SIM) to enable human users and machine agents to collaborate in real-time by utilizing Mixed Reality (MR) and Virtual Reality (VR) headsets which enable humans to interact with each other and with machine agents remotely in an immersive environment. The SIM system integrates robotic data collection platforms to collect visual data of the site, with critical guidance from human users on how to collect the best data. The data is then analyzed in real-time by AI computer vision algorithms that utilize input from the human users to localize and quantify structural damage. The on-site inspectors and remote experts are then able to collaborate on reviewing the results in a spatially aware context through an immersive environment supported by MR/VR technology, and can utilize machine agents to collect more data from the site and/or re-analyze previous data based on the human users' expert judgement.

Several scientific challenges are addressed in this thesis as part of the process of creating SIM. Each challenge deals with facilitating collaboration between human users and machine agents for a different component of the SIM system. First, input from the on-site inspector must be incorporated into the data analysis step to minimize the gap between data analysis and verification by humans and ensure the high quality of results. The approach is to utilize human-AI collaboration for quantifying the sizes of structural damage regions. This is accomplished by integrating an AI-based interactive image segmentation algorithm with the MR headset which allows for refining the segmentation results interactively through user feedback. Second, accurate spatial alignment between separated devices with heterogeneous sensing and processing capabilities (e.g., MR headsets, robots) is still an open problem that is critical for spatially-aware human-robot collaboration. The approach utilized was to develop an image-based localization algorithm to spatially align the MR headset and robot in real-time, which facilitates human-robot collaboration to enhance the reliability of the data collection process by engaging the MR-equipped human inspector with the data collection platform. Third, seamless integration of VR users into SIM is required for distributed collaboration between remote VR users and on-site MR users. This includes solving the technical challenges related to spatial alignment between VR and MR users, as well as how VR users can interact with other components of SIM such as robots and AI. The approach is to utilize panoramic images to allow VR users to remotely inspect the site, and a novel image-based localization algorithm was developed to spatially align panoramic images with their real locations on-site. Distributed collaboration also includes integrating all of these components into a unified system as part of SIM, with the goal of enabling on-site and remote inspectors to collaborate with each other and with robots and AI through MR/VR. Experimental results are presented for evaluating each component of SIM individually, including lab and field results for evaluating the accuracy of the proposed systems for MR/VR and robotic implementations.

Acknowledgements

I would like to thank my two supervisors, Dr. Chul Min Yeum and Dr. Sriram Narasimhan, for their continued support and guidance throughout my journey and for helping me refine my ideas. I would like to sincerely thank Dr. Yeum for always encouraging me to strive to achieve my best and always supporting me. I would also like to thank members of my committee including Dr. Giovanni Cascante, Dr. Carl Haas, and Dr. William Melek for helping me improve my thesis after my comprehensive exam, and Dr. Gunho Sohn for agreeeing to be my external examiner.

I would also like to thank all of my fellow lab members at the Computer Vision for Smart Structures (CViSS) lab for providing me with technical and moral support which allowed me to reach this milestone. My special thanks goes out to Max Midwinter, Rishabh Bajaj, and JuAn Park who were there with me from the start of my journey as the original members of CViSS, and who helped us define the lab's initial research direction.

I would also like to thank the Natural Sciences and Engineering Research Council (NSERC), Ontario Graduate Scholarship (OGS), and the University of Waterloo for providing scholarship funding for my graduate studies.

Dedication

This thesis is dedicated to my parents, Abbas and Sawsan, who always supported me, and to my two sisters, Sara and Rania, who are a constant joy in my life.

I also dedicate this thesis to my fiancée, Maryam, who encouraged me to keep going and achieve my best. I am lucky to have you in my life, and here's to the beginning of us.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence 2

AR Augmented Reality 9

BRS Back-propagating Refinement Scheme 18

DCNNs Deep Convolutional Neural Networks 2

f-BRS feature Back-propagating Refinement Scheme 15

HL1 HoloLens 1 9

HL2 HoloLens 2 9

HMCI Human-Machine Collaborative Inspection system 37

IoU Intersection-over-Union 27

LLMs Large Language Models 91

MR Mixed Reality 3, 9, 44

MSL Multi-Shot Localization 5, 7

ROI Region-of-Interest 38

SBD Semantic Boundaries Dataset 18

SfM Structure-from-Motion 11

SIM Smart Infrastructure Metaverse 3

SLAM Simultaneous Localization and Mapping 2

SSL Single-Shot Localization 5, 7

VR Virtual Reality 3, 9

XR eXtended Reality 9

XRIV eXtended Reality-based Inspection and Visualization 15

List of Symbols

$\boldsymbol{A_s}$ Area of the planar segmented damage region. 23

$\boldsymbol{A}$ Spatial anchor created for image-based localization as part of SSL or MSL. xiii, 42

$\boldsymbol{C}$ Location of camera center in 3D world coordinates. 20

$\boldsymbol{I_{qi}}$ Image i captured by query camera. 65

$\boldsymbol{I_q}$ Image captured by query camera. 40

$\boldsymbol{I_{rj}}$ Image j captured by reference camera. 64

$\boldsymbol{I}$ 2D image captured by XR headset device. 24

$\boldsymbol{K}$ Camera Intrinsics Matrix: 3×3 matrix for projecting 3D points in camera coordinates to a 2D screen. 19

$\boldsymbol{M}$ Dense 3D point cloud map of the scene created by the robot using SLAM. M is in the m_r coordinate frame. 46

$\boldsymbol{M_s}$ Set of triangles in spatial mesh created by MR headset. 20

$\boldsymbol{N_s}$ Normal vector of the planar segmented damage region. 22

$\boldsymbol{P}$ Projection Matrix: 3×4 matrix for projecting 3D points in world coordinates to a 2D screen. 19

$\boldsymbol{R}$ Camera Rotation Matrix: 3×3 matrix for representing camera rotation relative to world coordinates. 19

$\boldsymbol{T_{m_r}^{m_q}}$ Spatial alignment 4×4 transformation matrix for transforming 3D points from m_r frame to m_q frame. 44

$\boldsymbol{X_p}$ 3D world coordinates of seed points selected by user. 20

$\boldsymbol{X_s}$ 3D world coordinates of points along the boundary of the segmented damage region. 20

$\boldsymbol{X_i^{m_r}}$ 3D coordinates of ROI of defect i in m_r frame. 47

$\boldsymbol{X_i^{m_2}}$ 3D coordinates of ROI of defect i in m_2 frame. 47

$\boldsymbol{Y^{m_r}}$ 3D pose specified by user to control robot in m_r frame. 49

$\boldsymbol{Y^{m_q}}$ 3D pose specified by user to control robot in m_q frame. 49

$\boldsymbol{c_{qi}}$ Query device q's camera frame coordinate system when capturing image i. 65

$\boldsymbol{c_q}$ Query device q's camera frame coordinate system. 43

$\boldsymbol{c_{rj}}$ Reference device r's camera frame coordinate system when capturing image j. 64

$\boldsymbol{c_r}$ Reference device r's camera frame coordinate system. 43

$\boldsymbol{m_q}$ Stationary map coordinate system of query device q. 44, 65

$\boldsymbol{m_r}$ Stationary map coordinate system of reference device r. 64

$\boldsymbol{q}$ Query camera. 40

$\boldsymbol{r}$ Reference camera. 40

$\boldsymbol{t}$ Camera Translation Vector: 3×1 vector for representing camera translation in camera coordinates. 19

$\boldsymbol{x_p}$ 2D pixel coordinates of seed points selected by user. 20

$\boldsymbol{x_s}$ 2D pixel coordinates of points along the boundary of the segmented damage region. 20

Chapter 1

Introduction

1.1 Background

In North America and many regions around the world, vital critical infrastructure that was built in the 20th century such as bridges, airports, treatment facilities, power plants, etc. are nearing, or have already reached, the end of their design service life. For example, the 2016 Canadian Infrastructure Report Card estimates the cost of replacing infrastructure in poor and very poor conditions at $141 billion ($10,000/household) [1]. In the United States alone, 42% of all bridges were built 50 years ago, and 7.5% are classified as structurally deficient [2]. The potential failure of engineered structural components which comprise nearly all such infrastructures poses significantly increased risks to public safety, while simultaneously exacerbated by the effects of climate change. There is a clear need to leverage new technologies that can cost-effectively and reliably assess the conditions of built structures and complement the experience of trained inspectors to evaluate their state objectively and guide repair and prioritization decisions.

Currently, the most common inspection method is the visual inspection. This method involves inspectors first surveying the site and visually observing the structure to identify defects, document their types (e.g., spalling, cracks, corrosion, etc.), and measure their sizes [3, 4]. However, such manual inspection has certain limitations as structures become larger, have more complex geometry, and are built in harsher environments. New inspection guidelines are also beginning to adopt quantitative assessment criteria that require the sizes of the defects to be measured precisely [5]. Repeatability also suffers due to the qualitative nature of visual inspections. Multiple assessments of the same structure have been shown to vary significantly when performed by inspectors with different skill levels or under different field conditions [6]. Therefore, the limitations of visual inspection necessitate the shift to technological solutions to achieve the desired inspection outcomes such as increased accuracy, safety, and efficiency.

Recent advances in Artificial Intelligence (AI) focused on new vision sensors and sensing platforms have helped in a technologically driven transformation of visual inspection. The process of visual inspection using visual data collected and analyzed from advanced sensing platforms and algorithms is often called vision-based inspection. Over the past several years, various vision-based inspection techniques have been proposed for assessing civil infrastructure using image-based methods [7, 8, 9, 10, 11]. Robotic platforms such as drones and ground robots can remotely collect condition data of structures through vision, infrared, and LiDAR sensors [12, 13, 14, 15, 16]. Harnessing the power of computer vision and deep learning algorithms allows automation of interpretation rapidly and reliably to localize and quantify the size and severity of defects using images [17, 18, 19, 20, 21, 22, 23].

Significant literature exists on the topic of alleviating access through the use of aerial drones and terrestrial mobile platforms equipped with vision cameras [24, 12, 15, 13, 14]. 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 [17, 25, 20, 19, 21, 26]. Simultaneous Localization and Mapping (SLAM) algorithms [27, 28, 29, 30] enable inspectors to reconstruct digital 3D point cloud models of the environment to localize and quantify damage in structural elements [31, 32, 33]. However, a majority of these works do not address the human-machine collaborative aspect.

While vision-based methods for structural inspections have shown significant promise, several challenges still exist. Much of the research in the field of vision-based inspections has evolved toward automated processes that aimed at minimizing the intervention of the human in the loop [34, 35, 36, 37, 38]. However, such autonomous systems are unable to contextualize information and reason with the environmental data to the same degree as humans, and they often lack spatial and temporal context when it comes to identifying the most important regions to the structure's integrity and health. They could cause uncertain and unreliable inspection results and may produce false positives. For example, under challenging situations (e.g., poor lighting, shadows, etc.), sensors cannot expect to collect enough high-quality data to ensure reliable outcomes, and inference from the automated system alone may be unsatisfactory. Another issue is that not all damage or changes are meaningful and useful to be considered, and their significance varies depending on their context including visual appearance, damage history, and locations on structures. Hence, the meaningful adoption of new technology in visual inspection should involve and take advantage of domain experts (e.g., inspector, engineer) in both the data collection and analysis phases. Thus, the expertise and intelligence of the inspector should be incorporated into the process by adopting interactive processes (e.g., checking data quality and refining processed data).

Current research has mainly focused on systems that are unable to leverage the best parts of human inspectors and data collection capabilities of these platforms. This is where immersive environments come in, as they form a vital and as yet missing link between humans and robots to enable collaborative environments for visual inspections. Engaging human inspector judgment during the inspection process via Mixed Reality (MR) and/or Virtual Reality (VR) technologies has generated significant interest recently. The idea is to use immersive visualization capabilities in MR and VR (MR/VR) so that inspection results or records can be overlaid on a physical/virtual structure. MR headsets enable users to see and interact with virtual content overlaid and blended on the real scene as holograms. On the other hand, a VR headset immerses users in a fully virtual environment. The concept of multiple MR/VR users interacting synchronously together and/or with machine agents within a 3D virtual environment, often referred to as the *Metaverse*, has been actively researched recently [39]. Beyond traditional 1D and 2D communication modes such as voice and video calls, the metaverse can provide users with immersive 3D awareness and co-presence. These features have gained interest from the industry for engaging remote experts for collaboration and training [40, 41, 42, 43, 44]. The term "*Industrial Metaverse*" was coined in the literature to define a metaverse where industrial/built environments (e.g., facilities, bridges, construction sites, etc.) are digitally reconstructed in virtual space so that users and experts can better collaborate to solve real-world industrial problems both remotely and on-site [45].

Such technological developments can be adopted in the field of vision-based inspections for aiding on-site inspectors by improving documentation, facilitating communication with remote domain experts to improve the quality of results, reducing the barrier of entry to interacting with robots and AI, and assisting with training new inspectors. This thesis aims to address these crucial aspects by integrating superior data collection, spatial mapping, and human-informed computer vision algorithms so that a human-machine collaborative inspection system can be established.

1.2 Objective and contribution

The primary contribution of this thesis is the development of the Smart Infrastructure Metaverse (SIM) system to engage inspectors/engineers, data collection robots, and AI in visual inspection tasks by facilitating collaboration through immersive virtual environments supported by MR/VR. MR headsets enable on-site inspectors to see and interact with digital information (e.g., inspection records) that are overlaid on the physical scene. On the other hand, VR headsets immerse the remote users in a virtual environment replicating the real scene. They can see the real scene using registered photos and perform the same inspection alongside the MR inspector. Robotic data collection platforms such as ground robots and drones are integrated into this system by allowing the on-site and remote inspectors to supervise the collection of high-quality data (images, LiDAR) using MR/VR. The large volume of the data gathered by the robots' equipped sensors is then analyzed using vision-based AI algorithms that rely on inputs provided by the MR/VR users to obtain high-quality results. This human-in-the-loop AI approach for analyzing data is deployed to overcome the limitations of fully automated vision-based algorithms and improve the reliability of inspection results. Fig. 1.1 shows a network diagram of the SIM system, which includes all of the components (MR/VR users, robots, AI server) interacting and collaborating with each other.

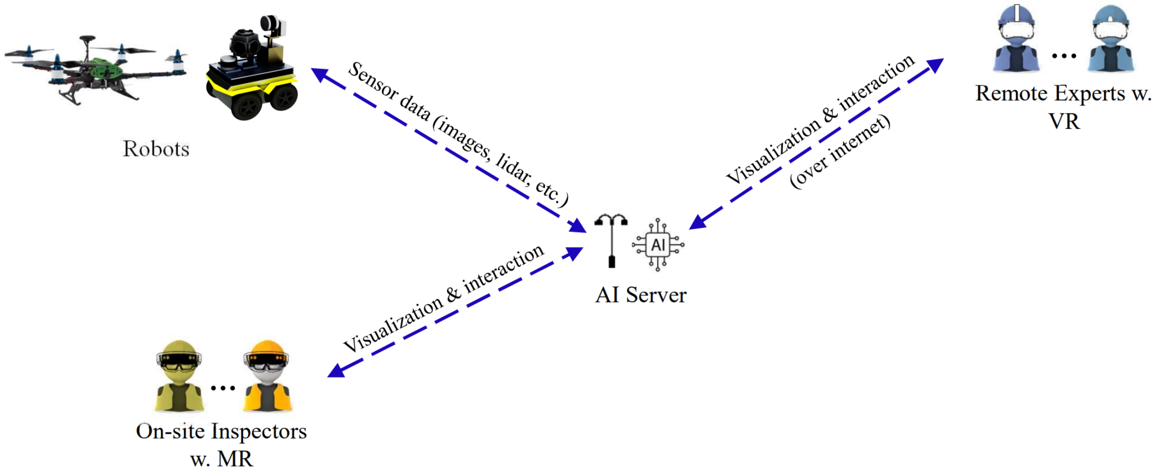

Robots
Sensor data (images, lidar, etc.)
Remote Experts w. VR
Visualization & interaction (over internet)
AI
AI Server
Visualization & interaction
On-site Inspectors w. MR

Figure 1.1: Network diagram for the SIM collaborative inspection system.

The novelty of the SIM system is that it combines and integrates many of the inspection steps such as data collection, processing, and decision-making into a single real-time process which is distributed across human inspectors and powered by machine agents (robots, AI) who perform rapid data collection and analysis. Thus, bridging the existing spatial-temporal gaps in the inspection process and providing a new way of inspecting structures.

SIM solves the issues with existing inspection workflows which are mostly asynchronous and are often not trustworthy. For example, in existing workflows remote experts can review the data (e.g., videos or images) collected by other on-site inspectors only after the data has already been collected. Therefore, they do not have the option to observe any regions that were potentially not captured, or were captured poorly, by the inspector during data collection. Processing of the data also happens asynchronously, so the issues in data collection could only be noticed much later after data collection. On the other hand, **SIM allows real-time communication between on-site inspectors and remote experts who can check that the data was collected properly and can collaborate with robots to gather high-quality data to reduce inspection time. Collaboration between humans and AI also ensures that data collection and analysis are integrated into a single synchronous on-site process so that human experts are able to determine if they need to collect more data and reduce unnecessary revisits to the site.** Such new modes of interaction between humans and machines provide more seamless informative engagement and collaboration during infrastructure inspections.

This thesis also introduces methodological contributions based on the algorithms that were developed as part of creating SIM, which are the following:

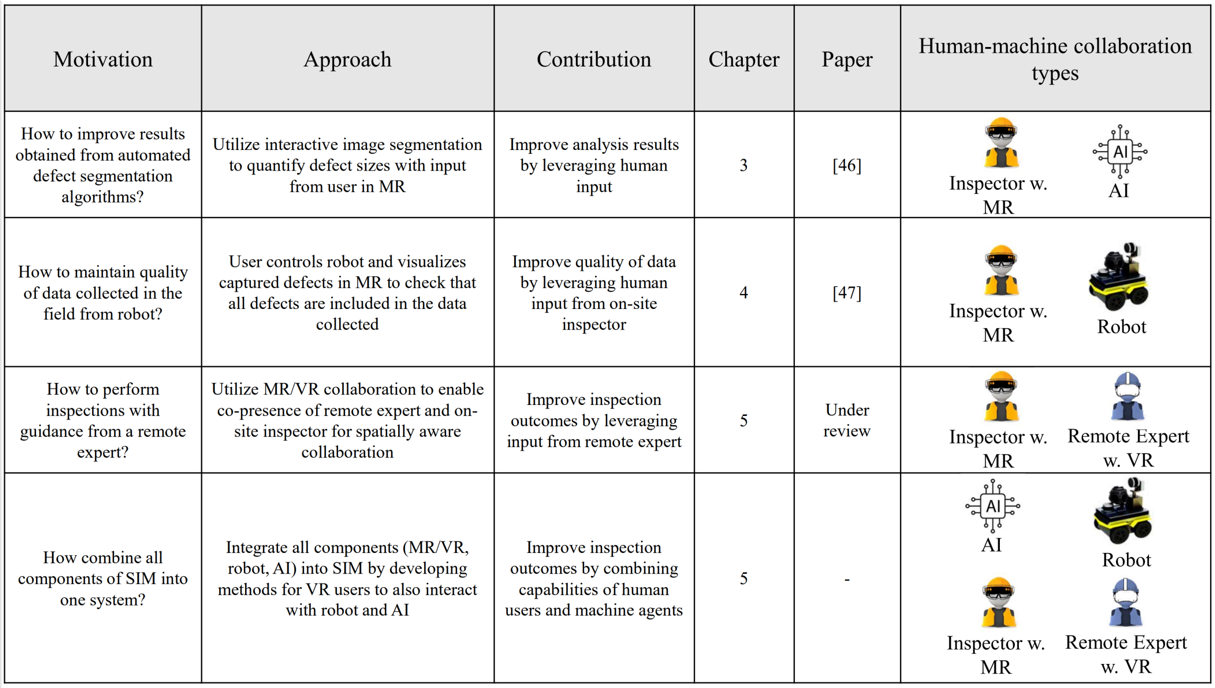
(i) A novel procedure was devised and deployed to create an interactive visual inspection system by involving the expert user to obtain and refine real-time quantitative damage measurements. This is done by taking the 2D boundary of the binary mask produced by the interactive segmentation algorithm and spatially matching 2D image pixels to the 3D scene using a ray-casting algorithm, followed by estimating the physical size of the damage.

(ii) A novel image-based localization algorithm called Single-Shot Localization (SSL) was developed to spatially align perspective images without requiring a pre-built 3D map. SSL can work on locally consistent sub-maps, thereby enabling real-time spatial alignment between data collection robots and MR-assisted human inspectors to maximize collaboration and reduce inspection time.

(iii) A novel image-based localization algorithm called Multi-Shot Localization (MSL) was developed to spatially align panoramic images in a 3D pre-built map to provide an all-around view of the infrastructure site to the VR users. MSL can be used to localize an image from any camera model (pinhole, panorama, etc.), and can be used with standard learning-based features detectors and matchers. Thus, providing enhanced flexibility for many general use applications such as panoramic image localization.

1.3 Scope of work

In this thesis, several algorithms and methods were developed and incorporated to support the SIM inspection system. These techniques fall under the larger umbrella of human-machine collaboration and can be divided into human-AI collaboration, human-robot collaboration, and distributed collaboration. Various configurations and interaction methods between humans and machine agents were developed as new contributions to the literature. These include collaboration between MR/VR users and AI, collaboration between MR/VR users and robots, and collaboration between multiple remote MR/VR users, as outlined in Table 1.1. The scope of the completed works includes the tasks of developing these methods/algorithms, experimentally validating them, and integrating them into the unified SIM system.



Motivation	Approach	Contribution	Chapter	Paper	Human-machine collaboration types
How to improve results obtained from automated defect segmentation algorithms?	Utilize interactive image segmentation to quantify defect sizes with input from user in MR	Improve analysis results by leveraging human input	3	[46]	Inspector w. MR AI
How to maintain quality of data collected in the field from robot?	User controls robot and visualizes captured defects in MR to check that all defects are included in the data collected	Improve quality of data by leveraging human input from on-site inspector	4	[47]	Inspector w. MR Robot
How to perform inspections with guidance from a remote expert?	Utilize MR/VR collaboration to enable co-presence of remote expert and on-site inspector for spatially aware collaboration	Improve inspection outcomes by leveraging input from remote expert	5	Under review	Inspector w. MR Remote Expert w. VR
How combine all components of SIM into one system?	Integrate all components (MR/VR, robot, AI) into SIM by developing methods for VR users to also interact with robot and AI	Improve inspection outcomes by combining capabilities of human users and machine agents	5	-	AI Robot Inspector w. MR Remote Expert w. VR

Table 1.1: A summary of human-machine collaboration types studied in this thesis.

For human-AI collaboration, covered in Chapter 3, a novel damage quantification method using MR and interactive image segmentation was developed to improve the quality of results obtained from automated defect segmentation algorithms [46]. The proposed system improves analysis results by leveraging human input in MR for interactive segmentation. The system's performance of was experimentally demonstrated by measuring surface structural damage of an in-service concrete bridge. The proposed human-AI system underscores the advantages of real-time interaction between expert users and the MR device through immersive visualization to obtain better inspection outcomes in terms of accuracy and robustness.

For human-robot collaboration, covered in Chapter 4, the proposed system developed enables on-site inspectors to collaborate with data collection robots by automatically visualizing the results of defects captured by the robot in MR [47]. The proposed system improves the quality of the data by ensuring that all defects are captured during the data collection session through real-time human supervision. A technique called SSL was developed to create spatial anchors for real-time spatial alignment between the robot and the MR headset. A method was also developed for MR-equipped inspectors to control robots using the MR interface without requiring expert knowledge of how to control the robot. An experimental study was conducted in a lab environment to demonstrate the proposed human-robot system using a MR headset and a robot. Several fiducial markers were used to simulate structural damage for the robot to scan along a predefined inspection path. Then, regions of interest are successfully anchored to the real scene and visualized through the MR headset to measure the accuracy of the spatial alignment.

In Chapter 5, a distributed collaborative system was developed to enable multiple on-site MR-equipped inspectors to collaborate with remote VR-equipped experts, allowing multiple users to leverage co-presence for spatially aware collaboration and improve inspection outcomes. Distributed collaboration also enables all of the previously proposed components of SIM to work together because it allows VR users to leverage the support of robots and AI. A novel algorithm called MSL was developed for localizing 360° panoramic images and aligning them with a pre-built 3D map to facilitate co-presence between MR and VR users and allow VR users to collaborate with robots and AI. Panoramic images are also utilized for allowing VR users to control robots remotely, as well as leveraging the power of AI for interactive image segmentation to support synchronous data collection and analysis. An experimental study was performed to demonstrate the feasibility of the distributed collaborative system for remote MR/VR collaboration by testing it on a railroad bridge where two on-site MR inspectors and one remote VR inspector perform a visual inspection. The accuracy of MSL for panoramic image localization was evaluated to confirm its viability for remote collaboration.

Chapter 2

Literature Review

2.1 Autonomous vision-based inspections

Much of the research in the vision-based inspection field has focused on creating autonomous robotic platforms (e.g., ground robots, drones) that can navigate and scan structures to locate structural defects with minimal intervention from human inspectors. These workflows often start with the inspector planning a route for the robot to collect visual data that maximizes coverage of the site, then the robot navigates this route using path-planning algorithms. The most common approach is for the inspector to plan the data collection mission using aerial or satellite images of the site [48]. However, this approach cannot account for complex geometries of the structure which may require navigation in the vertical direction [49]. GPS-based path-planning algorithms also cannot work in areas with limited coverage such as under bridges, but this issue could be mitigated through SLAM-based path-planning if the robot can be localized to an existing 3D map of the site [36]. After collecting the data, a 3D map is reconstructed from images and/or LiDAR, and automated detection algorithms are applied to the images to localize defects on the structure. To account for false positives in the detection results, a common approach is for the inspector to perform a "virtual walk" of the reconstructed site to verify the defect detection results [50]. However, this approach cannot deal with incomplete data in cases where the robot did not scan the entire site, or if the quality of the data is insufficient, which may require multiple iterations of data collection sessions. The introduction of immersive environments promises to reduce the gap between automated data collection and human data verification by combining the two steps to allow inspectors to better reason with environmental data captured in real-time from robots.

2.2 MR/VR technology

The term eXtended Reality (XR) encompasses Augmented Reality (AR), VR, and MR [51]. AR enables the 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 [52, 53]. 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), Magic Leap 2, and Varjo XR-3. On the other hand, commercial VR devices include Oculus Quest 2, HTC Vive Pro 2, and Varjo VR-3.

2.3 MR for vision-based inspections

MR technologies have seen applications in medical, construction, manufacturing, education, entertainment, aerospace, and other fields [54, 55, 56, 57]. 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 [58]. AR was also studied as a means to annotate, organize and visualize images and metadata during inspections here [59]. Another research work [60] demonstrated the application of Microsoft's HL1 for bridge inspections by allowing users to measure physical distances between any two points on the structure. The HL1 headset 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 [61]. In a related work, a framework for MR enabled inspection system using AR/MR devices was recently presented in [62]. Their system can detect and segment damage using a semi-supervised SegNet-based DCNN.

However, all of the previously discussed studies did not harness the spatial mapping or 3D sensing capabilities of MR devices for quantitative evaluation. This is because MR devices such as the HL2 or Magic Leap typically have limited spatial mapping capability due to the short range of their depth sensors (<5m). This issue limits their effectiveness for the localization of damages in the real scene beyond this short range, which can limit the potential usage of MR devices in the actual inspection process.

2.4 MR for human-robot collaboration

To address the limitations of using MR headsets for visual inspections, there have been attempts to integrate external sensing platforms equipped with mobile robots for data collection. These sensors complement the MR headsets to increase their sensing capabilities. Thus, human-robot collaboration between the MR inspector and the data collection robot is investigated in the literature.

A human-infrastructure interface was developed to allow inspectors to visualize and interact with real-time data received from low-cost smart sensors attached to structures [61]. The data received from the embedded sensors are then localized to the user's environment using QR codes on physical assets. However, such fiducial markers are often not practical because they require manual setup and can be damaged over time. Thus, marker-less pose estimation and spatial alignment techniques have been proposed to enable MR headsets to be localized relative to prior 3D maps generated by external sensors. A joint point cloud registration/image-based localization technique was proposed to allow a HL2 headset to be localized relative to a prior 3D map generated by a terrestrial laser scanner, which uses visual and depth data generated from the HL2's onboard sensors for localization [63]. This technique allows data annotated in the 3D map to be visualized to the HL2 user by localizing the HL2 headset to the 3D map, and then projecting the positional data in the HL2 headset's camera frame.

However, these technique, as well as other state-of-the-art image-based localization techniques found in the literature, require that the prior 3D map be globally consistent and complete before the MR headset's position can be localized relative to the map, which limits the applicability of these methods for real-time applications where mapping and spatial alignment must be performed simultaneously [64, 65, 66, 67]. Thus, the inspector must wait until the 3D scanning procedure is complete before visualizing any detected ROIs on the structure, and thus, with this system, real-time collaboration is limited because localization cannot be performed synchronously during data collection. Such processes incur a long delay in the interaction between the robot and the MR headset and cannot support real-time supervision of data collection through MR.

2.5 Distributed collaboration through MR/VR

Another component to build SIM is distributed collaboration between multiple MR/VR users to engage remote experts in the inspection process in real-time. Thus, studies that discuss multi-user VR and MR collaboration are investigated in this section.

In the initial phases of XR technology, research focused on using VR headsets to enable multiple users to collaborate together in a shared virtual environment [68, 69]. The virtual environment could be based on either BIM models or 3D maps reconstructed using Structure-from-Motion (SfM) or LiDAR scans of the real sites, so that they resemble the as-built states of the site [70, 71, 72, 73]. VR-based collaboration is the first logical step of creating the metaverse since it does not require the implementation of a spatial alignment algorithm to align the coordinate systems of multiple users. However, a VR-based metaverse cannot allow users to be on-site because VR headsets completely block out the real world. Thus, to facilitate collaboration between on-site and remote users, on-site users equipped with MR headsets must be incorporated into the metaverse.

The next step was to incorporate the remote VR users with on-site MR users by using a front-facing camera carried by the MR user. The MR user attaches the camera to their MR headset which captures images or videos that are streamed to the VR user in real-time [74, 75, 76, 77, 78, 79]. The VR user can then visualize the on-site physical environment and collaborate with the MR user. However, using a camera that is fixed to the MR headset means that the VR user can only see what the MR user is seeing and cannot independently look in any other directions. Thus, another approach was to use a panoramic camera to capture a 360°view of the scene, which allows the VR user to look at other directions that the MR user may not be currently looking at [80, 81, 82]. However, both of these approaches require manual spatial alignment of the camera so that it is fixed to MR headset. Strictly speaking, this application is to utilize the MR headset as a mobile head mount display, and does not leverage real-time pose estimation in MR. This limitation does not allow the VR user any freedom to move around the scene or away from the MR user and can only support one MR and one VR users in the same session.

To fix this issue, another approach was combining the panoramic camera and a pre-built 3D map to allow the VR user to be able to move around the reconstructed scene [83].

A 6 DOF motion tracker was attached to the panoramic camera to track its pose, which automatically performs the spatial alignment between the MR headset, panoramic camera, and VR user [84]. However, motion trackers require extensive setup and have a limited range, which is not suitable for outdoor applications [85]. All of the previously mentioned approaches have also not been demonstrated to support multiple MR and/or multiple VR users. So, if multiple MR users are present on-site, then spatial alignment between the multiple MR users must also be considered. Unfortunately, spatially aligned MR-based collaboration is much more technically challenging to implement than just VR-based collaboration, since all MR users must be spatially aligned relative to each other so that they are all using the same coordinate system relative to the physical environment. Early forms of MR-based collaboration did not incorporate spatial alignment, which greatly limited their utility [86].

However, spatial alignment is necessary for MR-based collaboration so that all annotations are anchored to the same location in the physical environment and all MR users can visualize them in the same location. The most common method to perform spatial alignment is to use QR or ArUco markers that are viewed by all the MR headsets' onboard cameras, which is then used to calculate the pose of each MR headset relative to the physical environment. Markers have been demonstrated to work well for MR-based collaboration in small indoor environments such as building facilities or medical settings [87, 88]. However, markers are not practical for larger environments, since many markers would have to be placed in the scene to avoid drift, and is not practical for civil engineering structures such as bridges. Markers would also not be useful for spatially aligning MR headsets relative to pre-built 3D maps since they cannot be placed on the structure for long periods because they might get damaged.

To address these issues, non-marker-based localization techniques have been proposed which utilize natural visual features in the scene [89]. An example is Microsoft's Azure Spatial Anchors (ASA). ASA allows users to create anchors that automatically encode visual and spatial data from a physical space into a set of 2D feature points [90]. The anchor can be shared with multiple devices to track their poses by registering images captured from the headsets' cameras. ASA has been utilized in multiple studies for multi-user MR/VR collaboration because it supports cross-platform compatibility and ease of development [91]. While the technical details of ASA have not been disclosed, the available information suggests that ASA relies entirely on visual features extracted from 2D images for spatial alignment. This approach functions effectively in small-scale indoor environments where conditions are largely static and unaffected by lighting alterations due to weather or the time of day. However, when it comes to outdoor applications, 2D feature-based spatial alignment is not sufficient to provide robust and precise localization. Hence, the development and integration of a more sophisticated spatial alignment strategy into the system are essential for broader, outdoor applications, which are the proposed target applications.

2.6 Gaps

Several gaps were identified in the literature review. There have been very few studies that have tried to embrace the individual strengths of automated systems and humans into a unified system which can deal with complex tasks and under situations where each one on their own cannot solve. First, the previous MR-based methods for performing vision-based inspections do not utilize a framework for incorporating input from the user to modify/improve analysis results. Thus, they are not able to quantify the sizes of defects accurately, especially for classes of defects that have insufficient training data. MR headsets provide the opportunity for visualizing and interacting with results in real time, which must be utilized in the proposed SIM system for human-AI collaboration.

The methods investigated in the literature also were not able to effectively integrate external sensors with MR headsets because of the difficulty of the spatial alignment problem. The spatial alignment techniques presented in the literature to align MR headsets and external sensors have been shown to be impractical due to the inadequacy of fiducial markers in civil engineering environments, or because of algorithmic limitations that prevent real-time use cases. Therefore, these spatial alignment issues prevent effective collaboration between MR users and robots equipped with external sensors. Thus, the proposed SIM system must utilize a marker-less spatial alignment method that overcomes these limitations through image-based localization.

Unfortunately, there is also limited work in the literature on spatially aware MR/VR collaboration, since spatially aligned MR/VR-based collaboration is more technically challenging than VR-based collaboration. The works discussed in the literature utilize either motion trackers or physical markers to perform spatial alignment between MR and VR users, which have only been demonstrated to work in small indoor spaces or outdoor spaces under ideal conditions. Thus, none of the existing approaches are feasible for aligning multiple MR/VR users in large outdoor spaces often found in civil engineering structures.

The contributions of this thesis address each of the previously stated gaps, with the goal of supporting human-machine collaboration by relying on the individual strengths of each component in the SIM system. First, interactive image segmentation through MR is proposed for improving analysis results through human-AI collaboration. Second, spatial integration of multiple external sensors and headsets for human-robot collaboration is enabled by the proposed image-based SSL algorithm. Third, spatially aware MR/VR collaboration is enabled through 360° panoramic images that allow the VR user view the site remotely and facilitate co-presence between MR and VR users. The MSL algorithm was developed for localizing 360° panoramic images to facilitate spatially-aware collaboration between MR/VR users.

Chapter 3

Interactive Defect Quantification Through Mixed Reality

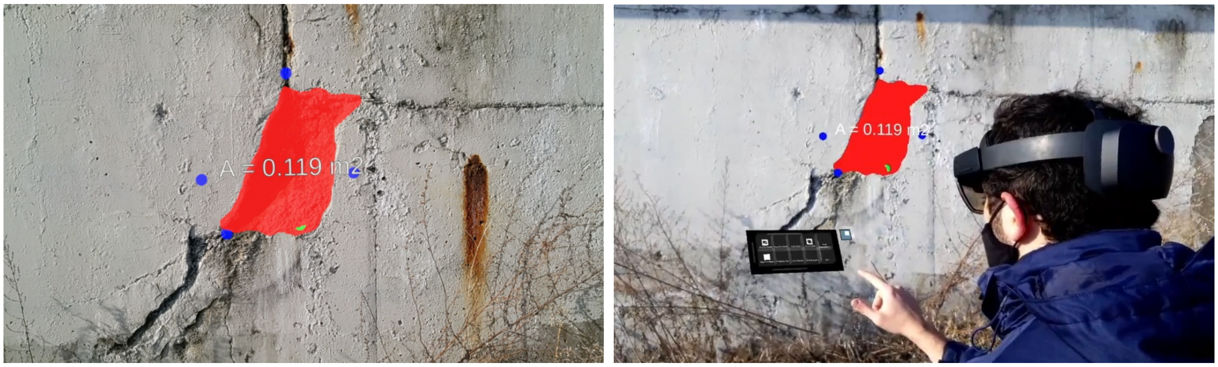
3.1 Overview

In this chapter, a novel real-time visual interactive inspection system is proposed as a component of SIM, which can detect and measure the area of surface damage enabled through MR (HL2), called eXtended Reality-based Inspection and Visualization (XRIV). This chapter presents a summarized version of the work submitted to a journal paper [46]. Throughout this chapter the terms XR and MR will be used interchangeably. This is because the technique developed here is applicable for both MR and VR headsets. This chapter will focus mainly on its implementation in MR, while Chapter 5 discusses its VR implementation briefly.

3.2 Approach and contribution

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. Fig. 3.1a shows the user's point-of-view when using XRIV implemented on HL2, while Fig. 3.1b shows the same scene from a third-person view through a two-way interface on a connected Android smartphone.



(a) (b)

Figure 3.1: 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.

3.3 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 [17, 25, 26], or to perform automatic pixel-wise segmentation of the damage region [19, 21, 92]. 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 [93], 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 [94]. 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 [95]. 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 [95]. 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 [96]. 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 [97] containing the two intermediate convolutions layers that were trained on the benchmark Semantic Boundaries Dataset (SBD) [98]. 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 3.5.1.

In addition to the variation in the network configuration, a key difference between the proposed and the original implementation [93] 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.

3.3.1 Spatial Computation

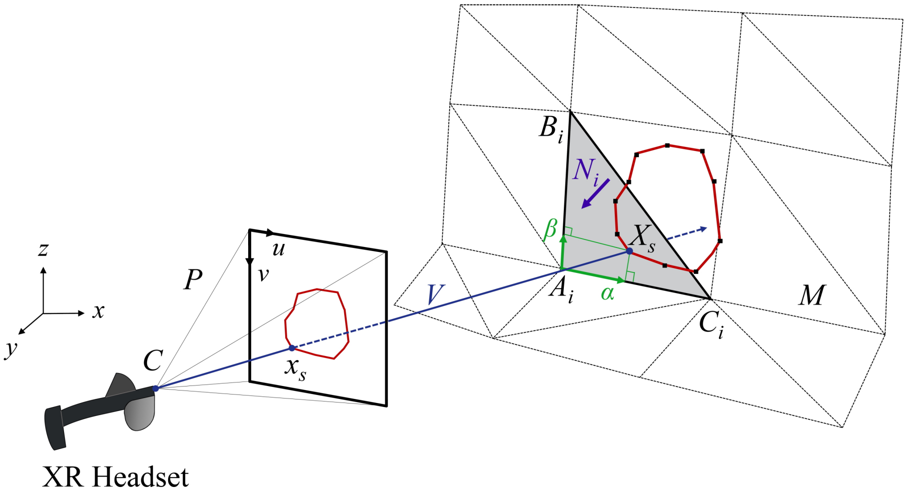


Figure 3.2: Pinhole camera geometry.

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. 3.2. 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 (provided 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 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. 3.1.

$$P = K\left[\ R \mid t\ \right] \tag{3.1}$$

The 2D pixel coordinates ($\boldsymbol{x_s}$) of points along the boundary of the segmented damage region on the image plane and the corresponding 3D points ($\boldsymbol{X_s}$) in the world coordinates are related through the projective transformation in Eq. 3.2:

$$x_s = PX_s \tag{3.2}$$

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 ($\boldsymbol{X_p}$) onto the image plane to obtain their image pixel coordinates ($\boldsymbol{x_p}$).

The projective transformation in Eq. 3.2 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 study are described in Algorithm 1. With reference to Fig. 3.2, the process of ray-casting locates the intersection point of the ray extended from the camera optical center ($\boldsymbol{C}$) from a point on the image plane with the spatial mesh ($\boldsymbol{M_s}$) 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. 3.3 [99]:

$$X(\lambda) = C + \lambda V \quad \text{where} \quad V = P^{+}x \tag{3.3}$$

where P^{+} is the pseudo-inverse of P. M_s consists of a set of triangles in the world coordinates that are joined along their edges, as seen in Fig. 3.2.

The ray-casting algorithm takes C, V, and M_s as inputs and iterates over every triangle in M_s 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 [100]. 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_s at multiple points since the real scene of M_s 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_s 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.

Algorithm 1 Ray-casting

```
Input
    $C$: Ray origin point (Camera Optical Center)
    $V$: Ray direction unit vector ($P^+x$)
    $M_s$: Set of $m$ triangles in spatial mesh
Output
    $X$: Intersection point between ray and spatial mesh
$\lambda_{min} \leftarrow$ empty    ▷ Initialize $\lambda_{min}$ as a placeholder value
for triangle $i = 1, 2, \ldots, m$ in $M_s$ do    ▷ Iterate over triangle $i$ joined by points $A_i$, $B_i$, $C_i$ with normal $N_i$
    $\lambda_i \leftarrow \frac{(C - A_i) \cdot N_i}{V \cdot N_i}$    ▷ Find $\lambda_i$ 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 $\alpha$, $\beta$ 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 $\lambda_i$ to $\lambda_{min}$ 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 $\lambda_{min}$
```

3.3.2 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 ($\boldsymbol{N_s}$). 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. 3.4. 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.

$$y_s = T(X_s - \bar{X}_s) \text{ where } T = \begin{bmatrix} n_u & n_v \end{bmatrix}^T \tag{3.4}$$

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 ($\boldsymbol{A_s}$) is then calculated using the Shoelace formula [101] in Eq. 3.5, 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,

$$A_s = \frac{1}{2}\sum_{i=1}^{k}\begin{vmatrix} u_i & u_{i+1} \\ v_i & v_{i+1} \end{vmatrix} \tag{3.5}$$

3.4 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. 3.1) 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

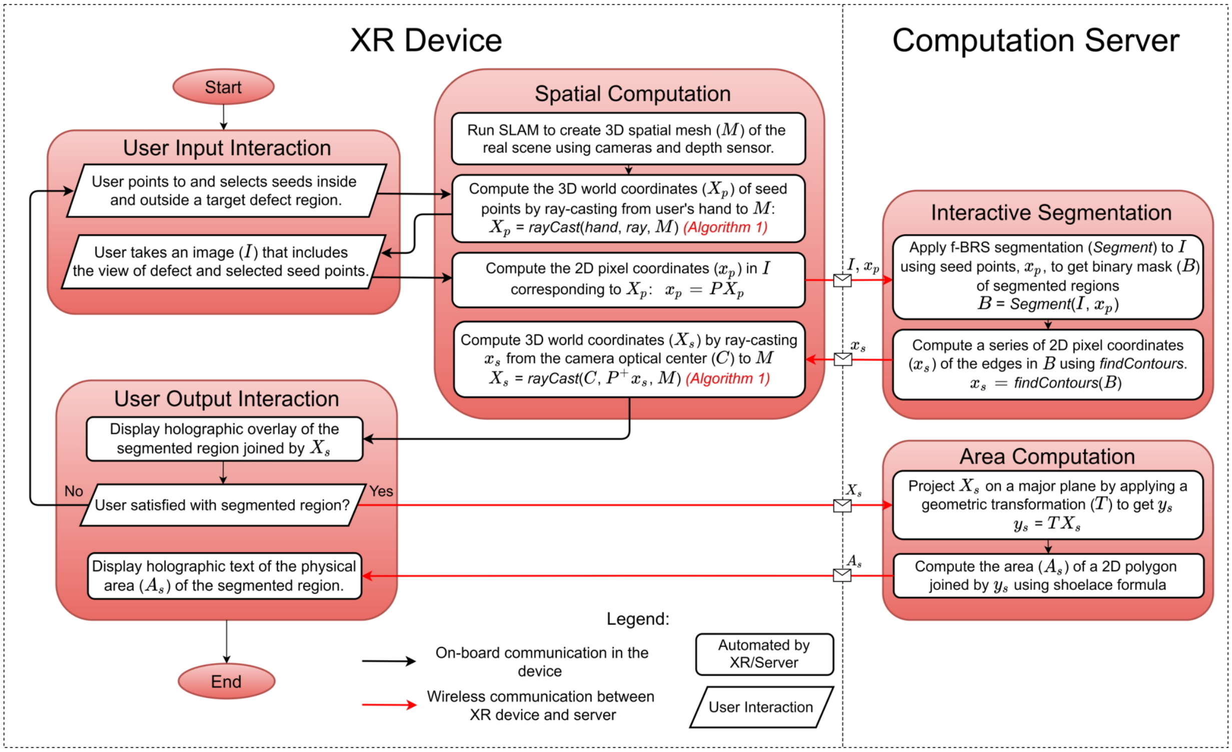

XR Device
Computation Server
Start
User Input Interaction
User points to and selects seeds inside and outside a target defect region.
User takes an image (I) that includes the view of defect and selected seed points.
Spatial Computation
Run SLAM to create 3D spatial mesh (M) of the real scene using cameras and depth sensor.
Compute the 3D world coordinates (X_p) of seed points by ray-casting from user's hand to M:
X_p = rayCast(hand, ray, M) (Algorithm 1)
Compute the 2D pixel coordinates (x_p) in I corresponding to X_p: $x_p = PX_p$
Compute 3D world coordinates (X_s) by ray-casting x_s from the camera optical center (C) to M
X_s = rayCast(C, P^+x_s, M) (Algorithm 1)
I, x_p
x_s
Interactive Segmentation
Apply f-BRS segmentation (Segment) to I using seed points, x_p, to get binary mask (B) of segmented regions
B = Segment(I, x_p)
Compute a series of 2D pixel coordinates (x_s) of the edges in B using findContours.
x_s = findContours(B)
User Output Interaction
Display holographic overlay of the segmented region joined by X_s
No
User satisfied with segmented region?
Yes
Display holographic text of the physical area (A_s) of the segmented region.
X_s
A_s
Area Computation
Project X_s on a major plane by applying a geometric transformation (T) to get y_s
y_s = TX_s
Compute the area (A_s) of a 2D polygon joined by y_s using shoelace formula
End
Legend:
On-board communication in the device
Wireless communication between XR device and server
Automated by XR/Server
User Interaction

Figure 3.3: Real-time XRIV pipeline using XR device and computational server.

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. 3.3. 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. 3.4.

One of the key inputs for interaction is actualized in the component *Input User Interaction* in Fig. 3.3, 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. 3.4a), 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_s) (Algorithm 1). Then, the user captures an image ($\boldsymbol{I}$) using the XR device's built-in

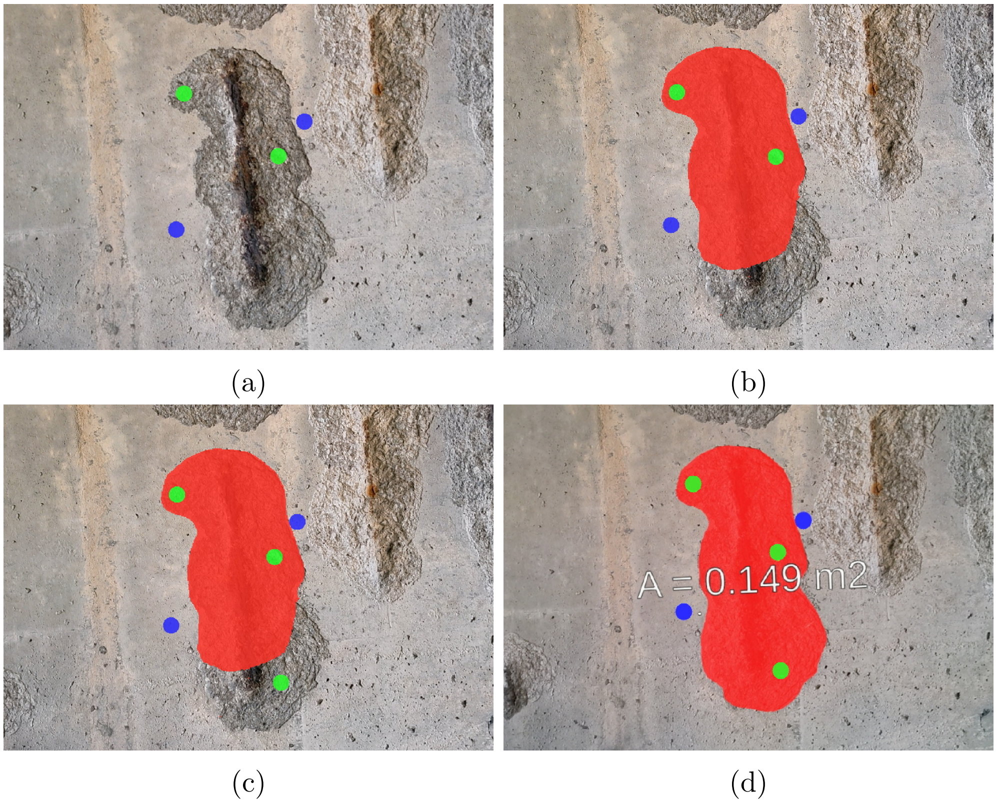

(a) (b) (c) (d)

Figure 3.4: 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.

camera that must show the full target damage region as well as all the seed points selected, initiating Step 2 in Fig. 3.4b. 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. 3.2 in Section 3.3.1).

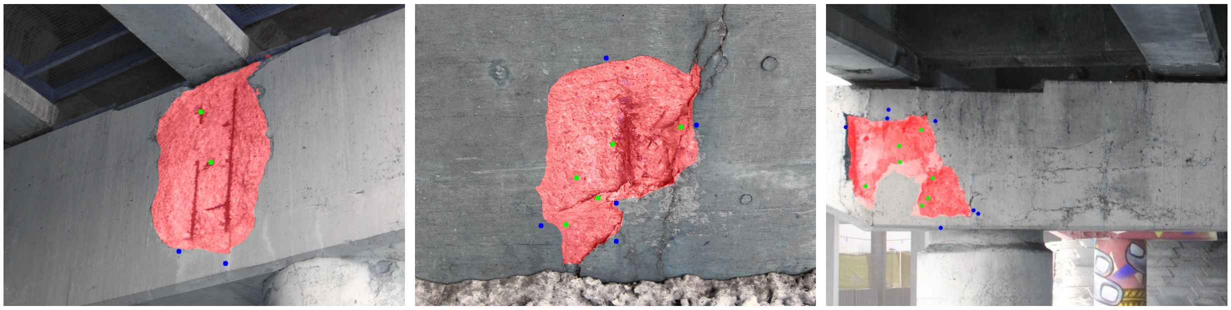
Next, the XR device sends I and x_p to the computational server to segment the damage region using the f-BRS segmentation algorithm, denoted as *Segment* in Fig. 3.3 (Section 3.3). *Segment* takes x_p as input seed points to segment I and produces a binary mask (B) that indicates where the damage is 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 [102]. 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. 3.3) 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_s (red colored region in Fig. 3.4b).

In *User Output Interaction* in Fig. 3.3, 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. 3.4b. This component allows the user to interactively improve the quality of segmentation, by adding more positive or negative seed points and then repeating the segmentation process (Step 3 in Fig. 3.4c). After obtaining a satisfactory segmented damage region, the user presses a button on the holographic menu (shown in Fig. 3.1) to obtain the physical area (A_s), which is mathematically represented as a closed polygon connected by X_s (see Section 3.3.2). 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. 3.4d).

3.5 Experimental Results

3.5.1 f-BRS Segmentation Evaluation



(a) 2 pairs

(b) 5 pairs

(c) 7 pairs

Figure 3.5: 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.

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 [93]. However, since the area measurement accuracy is a major concern rather than IoU in this study, 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. 3.5 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. 3.6 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 [4, 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 approximately 4.5 times faster than BRS while still maintaining a similar level of accuracy, which conformed to the findings of the original study [93]. 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

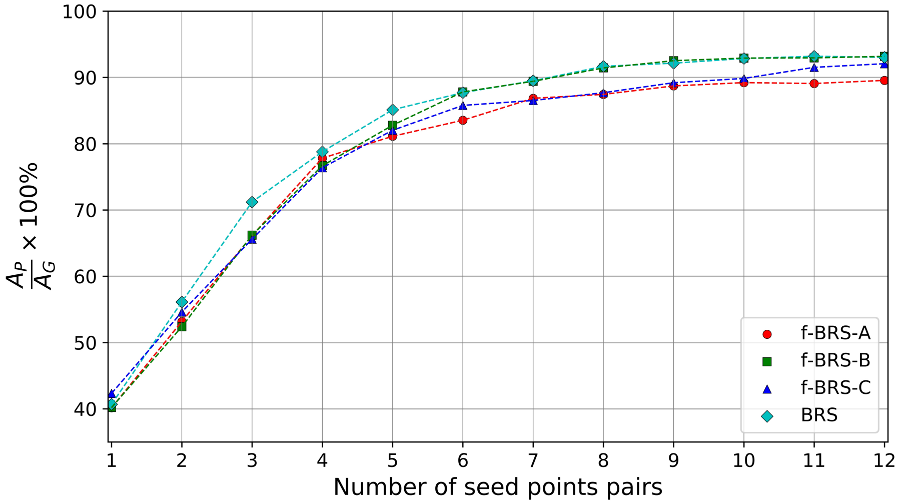


Figure 3.6: Evaluating area measurement accuracy as a function of the number of seed point pairs used for the segmentation models: f-BRS-A, f-BRS-B, f-BRS-C and BRS.

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 I 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. 3.5. Thus, although I 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.

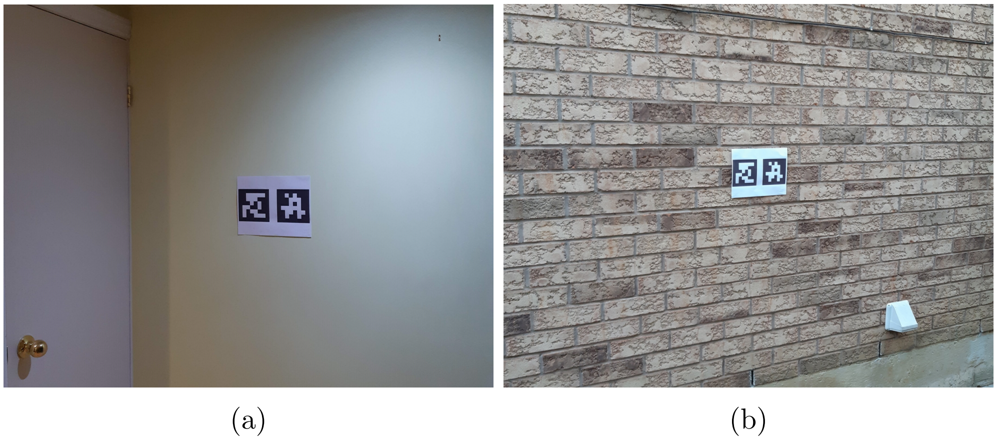
3.5.2 HoloLens 2 Performance Evaluation

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 3.3.1, 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 [27, 28, 29, 30] and SLAM applications for infrastructure [24, 31, 12, 13, 32]. 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 [103]. 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. 3.7a) and outdoor (Fig. 3.7b) walls to investigate the effect of illumination [104]. The measurements are made possible using the built-in infrared-based depth sensor (Microsoft Azure Kinect), whose performance is known to degrade under direct sunlight due to infrared interference [105].



(a)

(b)

Figure 3.7: 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.

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 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). An ArUco marker detector implemented in OpenCV [104] 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 3.3.1, 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. 3.8 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).

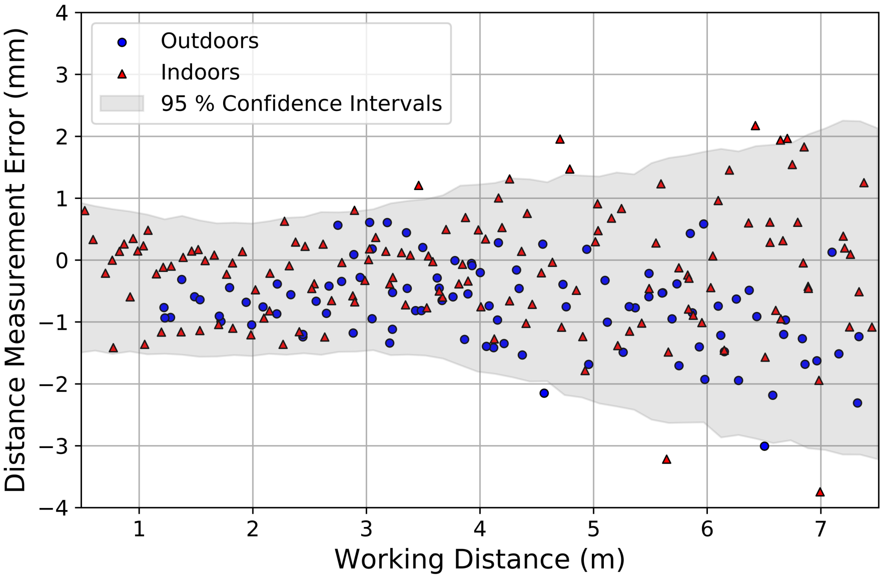


Figure 3.8: Scatter plot of distance measurement errors at different working distances between 0.5 m and 7.5 m under indoor and outdoor scene lighting conditions. Note that the gray area shows the 95% confidence interval (sliding window = 2 m).

In Fig. 3.8, the graph shows that the average measurement errors and their spread increase as the working distances increase. I 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 [106] 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 $\times 10^{-7}$, 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 3.1 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

Table 3.1: Aggregate statistics of distance measurement error under indoor and outdoor settings (95% confidence interval).

Location	Mean error ± spread
Indoor	0.2 ± 1.8 mm
Outdoor	0.8 ± 1.3 mm
Indoor & Outdoor	0.4 ± 1.7 mm

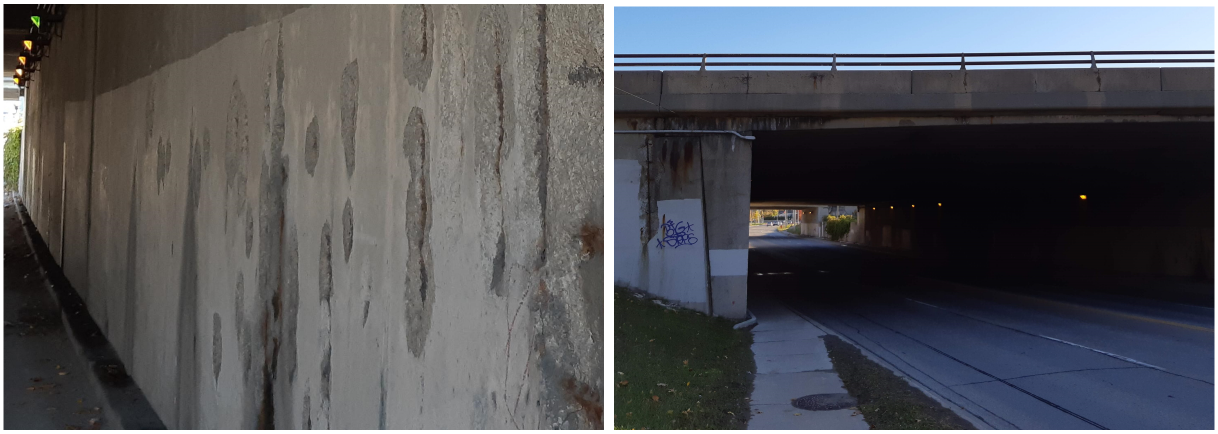
a significant factor for SLAM accuracy. The spread for both conditions is also relatively low (<2 mm), allowing consistent measurements. These experiments show that HL2 can produce sub-centimeter accuracy for both indoor and outdoor lighting conditions at working distances below 7.5 m. This level of accuracy is generally acceptable and sufficient for vision-based inspection applications [4, 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.

3.5.3 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. 3.9a) and outside (Fig. 3.9b) 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 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. 3.10. Then, using a 4-



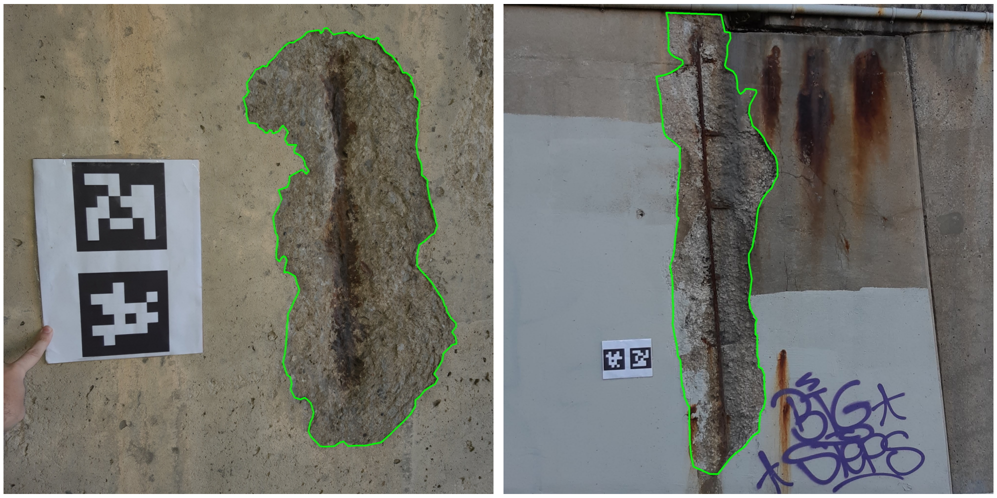
(a)

(b)

Figure 3.9: Overview of test spalling regions at a bridge abutment (a) underneath bridge (SA) and (b) outside bridge (SB).

point homography algorithm, a transformation matrix was obtained to transform all image pixel coordinates to corresponding physical coordinates [99]. The boundary of spalling on the images was manually segmented, which is shown as an outline in Fig. 3.10a and Fig. 3.10b 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 to calculate the area of each spalling region using the Shoelace formula explained in Section 3.3.2. The ground-truth areas of SA and SB are presented in Table 3.2.

The performance of XRIV was tested in its real-time mode. Seed points were selected and fed as inputs into the XR interface for interactive segmentation of spalling boundary (see Section 3.4). As discussed in Section 3.5.1, 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. 3.11 shows the scenes of the XR interface after selecting seed points from SA in Fig. 3.11a and SB in Fig. 3.11b. 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. 3.3) 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



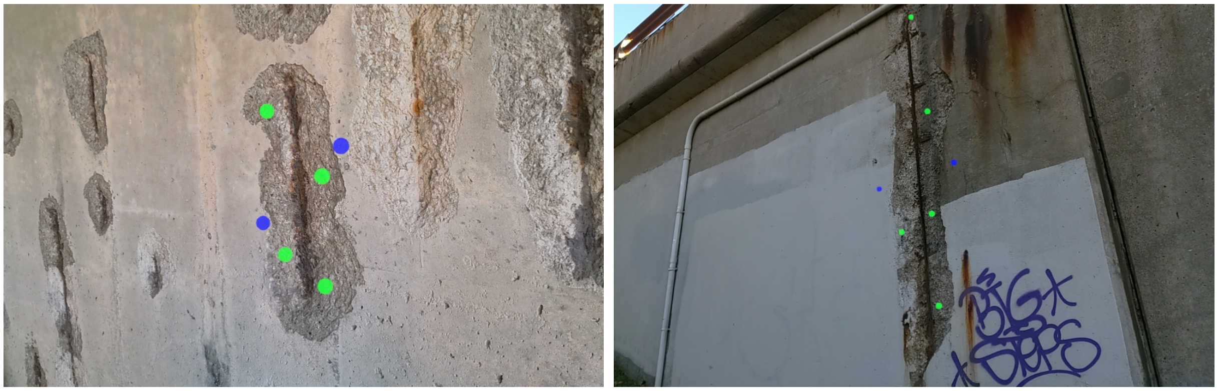
(a) (b)

Figure 3.10: Measurement of ground-truth spalling areas for (a) SA and (b) SB.

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. 3.12. 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 [93]. 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.

The measurement accuracy of XRIV 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 3.3.2. The results of the experiment are presented in Table 3.2, which



(a) (b)

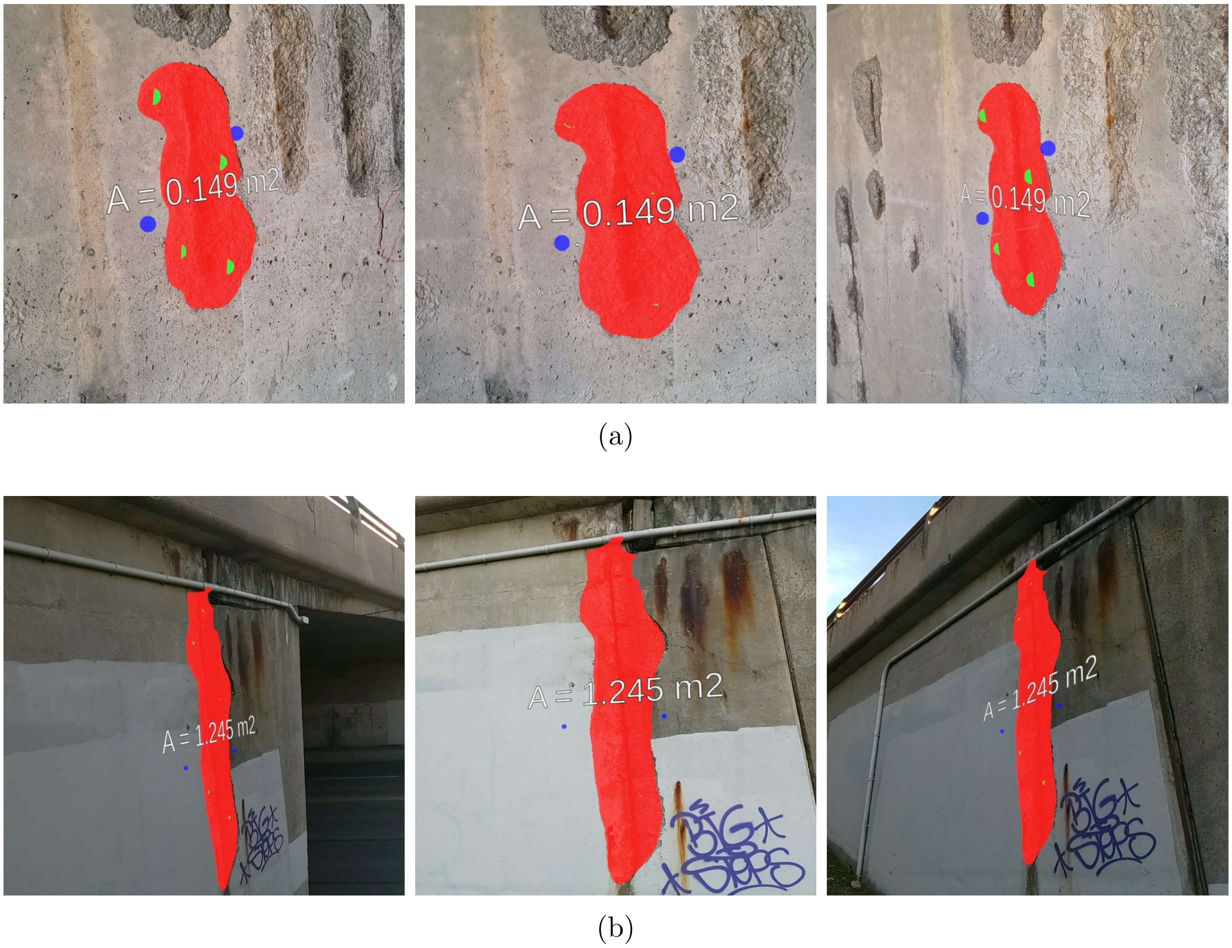
Figure 3.11: Seed points selected on SA in (a) and SB in (b): The points inside and outside spalling regions indicate positive and negative seed points, respectively. These images were captured on the HoloLens 2.

shows the damage size measurements compared to ground-truth measurements.

Table 3.2: Damage size measurements for tested spalling regions using XRIV in real-time compared to ground-truth measurements.

Target damage	SA	SB
XRIV	0.15 m^2	1.25 m^2
Ground-truth	0.14 m^2	1.14 m^2
Error	4.2%	8.9%

The results show 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 3.5.2 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 which makes it suitable for vision-based inspection applications [4, 3].



(a)

(b)

Figure 3.12: 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.

Chapter 4

Enabling Human-Machine Collaboration in Infrastructure Inspections through Mixed Reality

4.1 Overview

In this chapter, a Human-Machine Collaborative Inspection system (HMCI) is proposed as a component of SIM to enable the MR-equipped inspector to visualize, supervise, and improve results gathered by a robotic data collection platform in near real-time. This chapter presents a summarized version of the work submitted to a journal paper [47].

4.2 Approach and contribution

The novelty of HMCI is to spatially align the MR headset and the robot in real-time to enable an interactive and immersive environment for the user to perform the tasks of visual inspection with the aid of robotic platforms equipped with non-wearable and powerful visual and range sensors. A new image-based localization technique called SSL is proposed, which enables the robot and the MR headset to achieve locally consistent spatial alignment in real-time using only one image from each device and depth information from a depth scan captured using the robot. While this workflow utilized one robot and one MR headset, the methodology proposed here is scalable to multiple robots and devices. The main contributions are the following:

(i) HMCI fuses state-of-the-art computer vision, Simultaneous Localization and Mapping (SLAM), and mixed reality technologies to enable human and robot collaboration to enhance infrastructure inspection quality, reliability, and safety.

(ii) A novel SSL algorithm that can work on locally consistent sub-maps, thereby enabling real-time simultaneous data collection and MR-assisted human inspection to maximize collaboration and reduce inspection time.

A general workflow of HMCI is described here: The robot collects data from a target structure using its equipped 2D sensors including visual cameras, thermal cameras, or stereo cameras, and 3D sensors such as LiDAR. A large volume of the data gathered by the equipped sensors is transmitted to a remote computer server and then vision-based visual inspection algorithms are applied to those data to detect and localize defects on the structure. A 3D map of the structure and its surroundings is reconstructed from the scene using SLAM. SLAM also estimates the poses of the equipped sensors in the reconstructed 3D map. The MR headset is equipped with its own visual and depth sensors that allow it to spatially track its position. SSL is then deployed to spatially align the MR headset relative to the 3D map generated by the robot. Once the defect is detected, this information is then sent to the MR headset, which then renders holograms of the defect regions and anchors them on top of the actual location of the defects, along with the associated information such as the type of defects, estimated sizes, date of the inspection session, notes from previous inspections, etc, as illustrated in Fig. 4.1. This information can be also saved to the reconstructed 3D map and inspectors can re-localize their MR headsets when revisiting the site in future inspections and view the saved annotations from previous sessions.

The novelty of this image-based localization technique, SSL, is that it does not require the 3D map to be globally consistent or complete before localization can start, since users only require the locally consistent 3D map in their immediate vicinity to visualize a Region-of-Interest (ROI) on the MR headset. Thus, both the robotic data collection and MR-assisted human inspection are performed simultaneously to maximize collaboration and reduce inspection time.

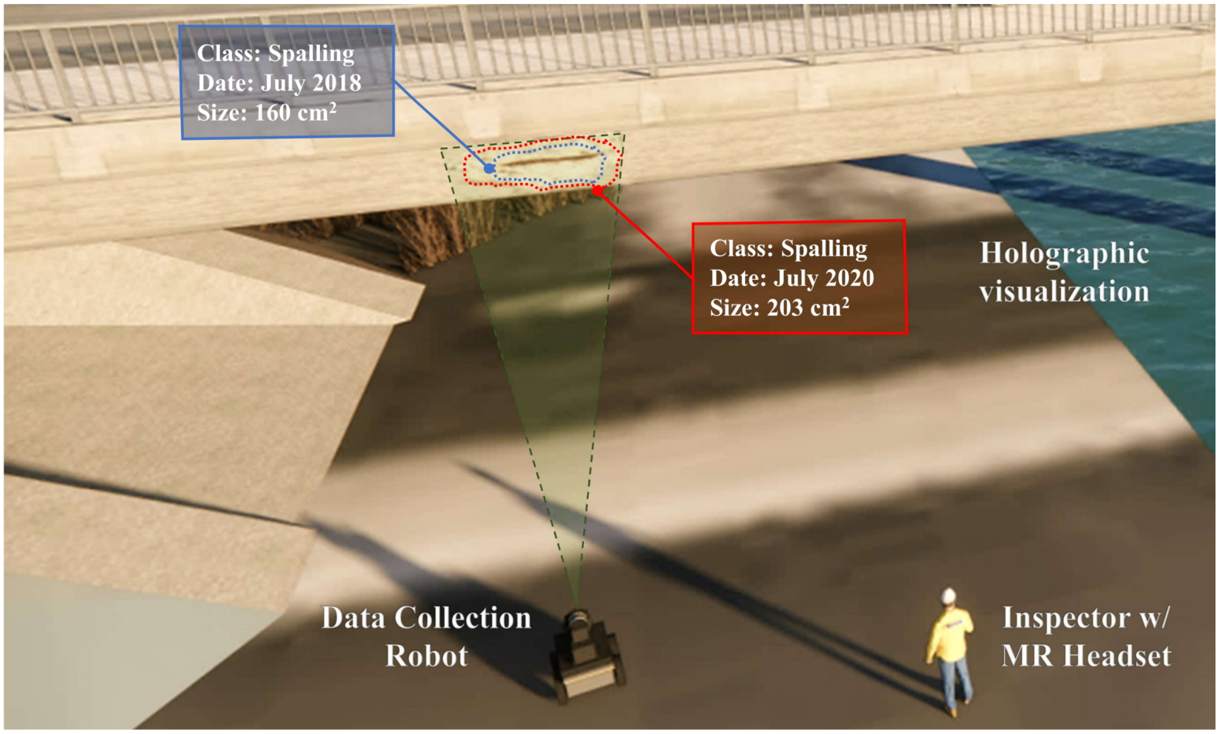


Figure 4.1: Illustration of HMCI for bridge inspection: The detected defect (spalling here) information is visualized to the inspector using the holographic MR display.

4.3 Spatial Alignment

Spatial alignment between multiple devices using visual sensors is a common challenge in computer vision [107, 108, 109]. For the alignment of data from the multiple devices, the coordinate system of each non-stationary sensor (sensor frame) should be linked to a stationary frame anchored in the physical environment (map frame). Spatial alignment is the process of localizing the sensor frames of multiple devices into a single common reference map frame of the physical environment. Spatial alignment is a crucial step to share and localize the data from multiple devices and analyze them with spatial context. In the context of spatial alignment in this paper, a reference data collection device (robot) gathers visual data and builds a 3D map of the scene, while a query device (MR headset) aims to localize its pose relative to the map frame.

In this study, a marker-less spatial alignment method is proposed, which utilizes natural visual features in the scene instead of fiducial markers to perform image-based localization [110, 89, 111]. To start, multiple (m) reference images of the scene are obtained from the reference device's calibrated camera (reference camera, $\boldsymbol{r}$), and then stored in a database (I_{r1}, I_{r2},... I_{rm}). The poses (T_1^r,T_2^r,... T_m^r) of the reference camera relative to the map frame are computed when each image is captured after processing them using SLAM. The reference device's onboard 3D sensors also capture 3D depth scans (D_1^r, D_2^r,... D_m^r) obtained from RGB-D cameras or LiDAR synchronized with each reference image to estimate real distances in the scene. A query device's calibrated camera (query camera, $\boldsymbol{q}$) then captures an image ($\boldsymbol{I_q}$) as input. A generalized image-based localization algorithm then uses the data gathered by the reference device as a prior and I_q gathered by the query camera to estimate the pose of the query camera (T^q) in the map frame. There are two types of image-based localization methods: 2D-2D methods [112, 113] and 2D-3D methods [114, 64]. Both are using a set of unique features and their descriptors (x_i^r, d_i^r) for the reference image (I_{ri}) and (x^q, d^q) for the query image (I_q).

In 2D-2D methods, an image retrieval algorithm is first performed to find the reference image (I_{rk}) among an image database (I_{r1}, I_{r2},... I_{rm}), which exhibits the largest overlap with I_q when viewing the scene. This is performed by clustering the descriptors of the reference images (d_1^r, d_2^r,... d_m^r) using bag-of-visual-words [115] or a vector of locally aggregated descriptors [116], to encode the descriptors of every reference image into a searchable indexed vector. I_{rk} can then be identified based on the closest matching (e.g., L1, L2 distance) index vector obtained from d_k^r and d^q. Then, features from the query image (x^q, d^q) and the closest reference image (x_k^r, d_k^r) are matched to estimate the relative pose between I_{rk} and I_q, which can be transformed into the map frame. An essential matrix (E) is computed from epiploar geometry and internal matrices of both reference and query cameras. The relative pose (extrinsic transformation) between the query and reference images are computed from E [99]. The advantage of 2D-2D image localization methods is that they do not require a globally consistent 3D map of the scene which can be computationally expensive to create. However, pose estimation often produces unreliable results when the scene overlap is small or the features are present in a single dominant plane (degenerate configuration) [117].

In 2D-3D methods, the database reference images (I_{r1}, I_{r2},... I_{rm}) and corresponding depth scans (D_1^r, D_2^r,... D_m^r) are used to construct a 3D map of the scene (M) using either standard Structure-from-Motion (SfM) or SLAM procedures. All feature points (x_i^r, d_i^r) extracted from I_{ri} are then back-projected to M to compute 3D points (X_i) corresponding to x_i^r. X_i is not denoted with r or q because its coordinates are defined in the stationary map frame, and not associated with sensor frames. X_i is estimated by triangulating co-visible features in at least two reference images, which is a part of the SLAM or SfM procedures [118, 103]. Depth constraints from (D_1^r, D_2^r,... D_m^r) can be applied to improve the accuracy of triangulation by bounding the triangulation error within a range obtained from the depth values [119]. Thus, a visual 3D features map of the scene (M_f) is obtained by triangulating co-visible 3D points (X_1, X_2,... X_m) and storing their associated descriptors (d_1^r, d_2^r,... d_m^r). Correspondences between (x^q, d^q) and M_f including (X_i, d_i^r) can then be obtained using "direct" 2D-3D features matching, where d^q are directly compared with M_f to find matching features to build 2D-3D correspondences between x^q and (X_1, X_2,... X_m) [64]. Indirect 2D-3D localization can also be used, where the closest matching reference image (I_{rk}) viewing the same scene as I_q is located through image retrieval. This type of method is called "indirect" because the set of potential 2D-3D features correspondences between query and a smaller set of reference images, which are narrowed down significantly by limiting the feature correspondence from the closest matching reference image (I_{rk}) after the image retrieval. Generally, indirect 2D-3D localization methods have been proven to be more efficient and accurate compared to direct 2D-3D localization methods because the image retrieval step eliminates 2D-3D correspondences from irrelevant reference images [120, 67].

In 2D-2D methods, the epipolar geometry between I_{rk} and I_q and matched features in 2D are used to compute E, followed by pose estimation. However, in the 2D-3D methods, I can compute the pose from 3D coordinates (X_n) in the map frame corresponding to each 2D feature point (x_n^q) in I_q. The closest matching reference image (I_{rk}) is first located using image retrieval, and then a feature matching algorithm is used to locate matches between I_{rk} and I_q, like 2D-2D methods.

A set of correspondences (n) is established between 3D reference feature points (X_n) and 2D query feature points (x_n^q) through direct or indirect matching. Since X_n and x_n^q describe the same set of points in the physical environment, X^n can then be projected onto the query camera's image (I_q) to obtain x_n^q using a projection matrix (P_q) encoding T^q and the query camera matrix (K_q), as seen in Eq. 4.1:

$$x_n^q = P_q X_n \tag{4.1}$$

Computing T^q that satisfies Eq. 4.1 using n 2D-3D correspondent points is called the perspective-n-point problem (PnP) [99]. There are several methods to solve the PnP problem, depending on a set of conditions. For $n = 3$, the problem is reduced to P3P, which can be solved using trigonometry to obtain up to four valid closed-form solutions [121]. A unique solution can be obtained if four or more points are available. For $n \geq 4$, the Efficient PnP (EPnP) algorithm formulates the general PnP as an optimization problem, where the goal is to estimate the best T^q that minimizes the re-projection error for n 2D-3D correspondent points [122]. However, EPnP performs poorly when there are outlier points. Thus, it has been shown that P3P used alongside RANdom SAmple Consensus (RANSAC) provides better performance in the PnP problem with $n \geq 4$ to obtain robust results when outliers are present compared to EPnP [123]. Therefore, P3P RANSAC is often used in 2D-3D image-based localization methods to estimate T^q.

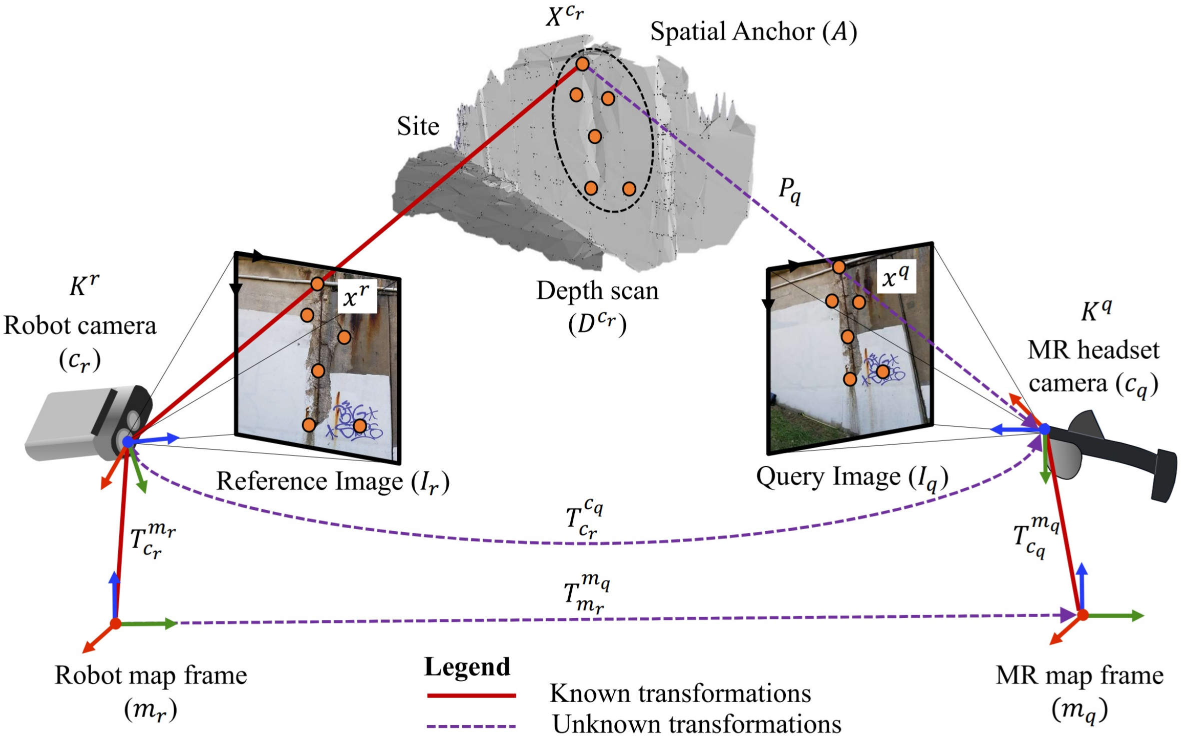


Figure 4.2: Frame diagram for spatial alignment between the robot and MR headset using SSL: Solid lines in red indicate known transformations between frames, while dashed lines in purple indicate unknown transformations that are estimated using SSL. The spatial anchor (***A***) created by the data from the robot consists of unique visual features detected in the reference image (plotted as orange dots) that are directly back-projected to the depth map to estimate their 3D points. The MR headset uses its 2D-3D correspondences for localization.

The advantage of 2D-3D image localization methods is that they are more accurate and robust than 2D-2D methods in general [64]. However, one of the major drawbacks is that the 3D map (M), 3D features map (M_f), and reference camera poses (T^r_1, T^r_2,. . .

T^r_m) must all be globally consistent before the process of localization can start. Otherwise, large deformations to the map such as bundle adjustments or loop closures during the 3D map generation will interfere with previously obtained localization results. Current localization methods do not address this issue because they assume that 3D mapping and spatial alignment are not performed simultaneously and that the reference and query devices operate during different sessions. They require the 3D map to be available before localization starts.

Therefore, traditional 2D-3D image-based localization approaches are not suitable for HMCI because they are not designed to co-register multiple devices to collect data simultaneously. With such traditional methods, the robot (reference device) is first required to collect as many images as possible of the scene to build M_f by detecting, tracking, and triangulating co-visible feature points from multiple images while performing SLAM. Such processes incur a long delay in the interaction between the robot and the MR headset.

The MR headset (query device) also remains long idle until the robot captures sufficient images of the scene to start localization, to ensure that loop closures or bundle adjustments do not affect localization results. Thus, the inspector with the MR headset must wait until the 3D scanning procedure is complete before visualizing any detected ROIs on the structure, and thus, real-time collaboration is limited and could be lagging.

To address this issue, I propose a new method, SSL, to enable spatial alignment between the reference and query devices in near real-time. SSL needs only a single image from each of the device's cameras and a depth scan (D^{c_r}) obtained from the robot's 3D sensor in the robot camera's frame ($\boldsymbol{c_r}$). The robot creates a "*spatial anchor*" (A) consisting of unique visual features found in the scene and anchored to the 3D space, and the MR headset utilizes the spatial anchor to localize its camera frame ($\boldsymbol{c_q}$) relative to c_r. The spatial relationship of these components is described in Fig. 4.2.

SSL starts by using feature detection to find 2D reference features (x^r, d^r) in the reference image I_r captured by the robot's camera. The 2D points in x^r are then back-projected to D^{c_r} to obtain the 3D coordinates of the reference feature points (X^{c_r}) in the c_r frame. The quantities X^{c_r}, d^r, and $T^{m_r}_{c_r}$ are then bundled together into what I call a spatial anchor (A), that can be shared between any number of query devices. The process of creating A is labeled as SSL_A in Eq. 4.2:

$$A = \text{SSL}_A(I_r, D^{c_r}, T^{m_r}_{c_r}) \tag{4.2}$$

When the MR headset is close to A, the query features (x^q, d^q) are then detected from the query image (I_q). Feature matching is then performed to find corresponding points between x^r and x^q by comparing their descriptors, d^r and d^q. I then deploy P3P RANSAC to match the points in x^q and X^{c_r} and obtain the relative pose ($T^{c_r}_{c_q}$) from c_q to c_r. The 4×4 transformation matrix ($\boldsymbol{T^{m_q}_{m_r}}$) that is used to convert the 3D points from the robot map frame (MR) to the MR map frame ($\boldsymbol{m_q}$) is then estimated using intermediate transformations from the relative poses of each device using Eq. 4.3:

$$T^{m_q}_{m_r} = T^{m_q}_{c_q} T^{c_q}_{c_r} T^{c_r}_{m_r} = T^{m_q}_{c_q} (T^{c_r}_{c_q})^{-1} (T^{m_r}_{c_r})^{-1} \tag{4.3}$$

The three transformations on the right-hand side of Eq. 4.3 are known: $T^{m_q}_{c_q}$ results from SLAM onboard the MR headset, $T^{c_r}_{c_q}$ is obtained from P3P RANSAC, and $T^{m_r}_{c_r}$ is computed from SLAM in the robot. The process of estimating $T^{m_q}_{m_r}$ using I_q and $T^{m_q}_{c_q}$, given A obtained from the reference device, is defined in Eq. 4.4 as SSL_Q:

$$T^{m_q}_{m_r} = \text{SSL}_Q(I_q, T^{m_q}_{c_q}, A) \tag{4.4}$$

SSL algorithm combines the advantages of 2D-2D and 2D-3D localization methods while overcoming key shortcomings in each method. First, the laborious triangulation step in most 2D-3D algorithms is replaced with back-projection using a depth map obtained from the robot perception kit. Thus, the robot does not need to collect multiple reference images from the scene to start spatial alignment. This approach brings faster localization by minimizing complexity and computational costs. 2D-2D image feature detection algorithms are directly deployed to find corresponding features instead of building a visual features map (M_f) through triangulation, which means that SSL can be deployed while the robot is collecting data without waiting for the global map completion and can work with at least one reference and one query images.

Note that since robotic 3D mapping and MR spatial alignment are performed concurrently, post-processing steps such as bundle adjustments or loop closures may cause large deformations to the global map as well as refinement of the robot's camera poses ($T^{m_r}_{c_r}$) computed from SLAM. This issue may affect localization results ($T^{m_q}_{m_r}$) computed before post-processing is applied but would not negatively impact SSL because it relies on locally consistent maps. In SSL, spatial alignment results are re-computed by identifying which quantities in the localization process are affected by post-processing and isolating them to re-compute $T^{m_q}_{m_r}$, instead of re-performing all the localization steps after refining the map through post-processing. In Eq. 4.3, $T^{c_r}_{c_q}$ and $T^{m_q}_{c_q}$ are invariant to bundle adjustments and loop closures because they are local transformations relative to c_r, and are independent of $T^{m_r}_{c_r}$ and the global map. Thus, $T^{c_r}_{c_q}$ and $T^{m_q}_{c_q}$ are not changed by the post-processing, and the processes such as P3P RANSAC and feature matching, which are computationally demanding, do not need to be re-performed. When post-processing is applied, a new value for $T_{m_r}^{m_q}$ is computed using Eq. 4.3 by updating $T_{c_r}^{m_r}$ only.

At each robot location, all the information needed to perform localization (X^{c_r}, d^r, $T_{c_r}^{m_r}$) can be stored as a spatial anchor (A). The spatial anchor can also be used for localization when the MR-equipped inspector revisits the site in future sessions to visualize prior inspection records, and for co-localization of multiple MR headsets. When the users utilize spatial anchors for these applications, the image retrieval process is not required because only a single A is stored per ROI detected by the robot. To localize relative to previously created anchors, I only iterate through previous anchors to find the A with the highest number of matched 2D feature points compared to I_q.

In sum, SSL can streamline the workflow in the proposed HMCI by speeding up the spatial alignment process to support real-time collaboration between the robot and the MR headset. Once the spatial alignment procedure is complete and the location of the anchor is known in both devices, I can then convert the coordinates of any point from the robot map frame (m_r) to the MR map frame (m_q) using $T_{m_r}^{m_q}$ so that the MR headset can then render ROIs or segmentation masks detected by the robot into the MR user's perspective.

4.4 System Overview

The detailed workflow of HMCI is described in Fig. 4.3. An inspector with the MR headset, robot, and computation server can interact with each other to perform collaborative visual inspection tasks. The proposed system is designed to integrate all the core processes such as data collection, analysis, visualization, and interactive supervision conducted in these three components. In Fig. 4.3, the onboard communications in each component are marked as black solid arrows, and wireless communications are marked as red arrows. The phrases '*server request*' and '*server response*' indicate inputs and outputs to/from the computational server, respectively, that are sent/received by either the robot or MR headset. All the processes in this workflow are automated, except for machine-aided interactive damage refinement (marked as a parallelogram), which requires input from an inspector [46]. The detailed description and process for each component including the prerequisite for operating HMCI are provided in the following paragraphs:

Robot: The robot first collects an image (I_r) from the scene using its onboard visual camera (c_r), as well a 3D depth scan (D^{c_r}) in the c_r frame from a LiDAR or RGB-D camera with known external calibration to c_r. The robot also runs a SLAM algorithm to obtain the pose ($T_{c_r}^{m_r}$) of the camera relative to m_r when I_r is captured, while simultaneously

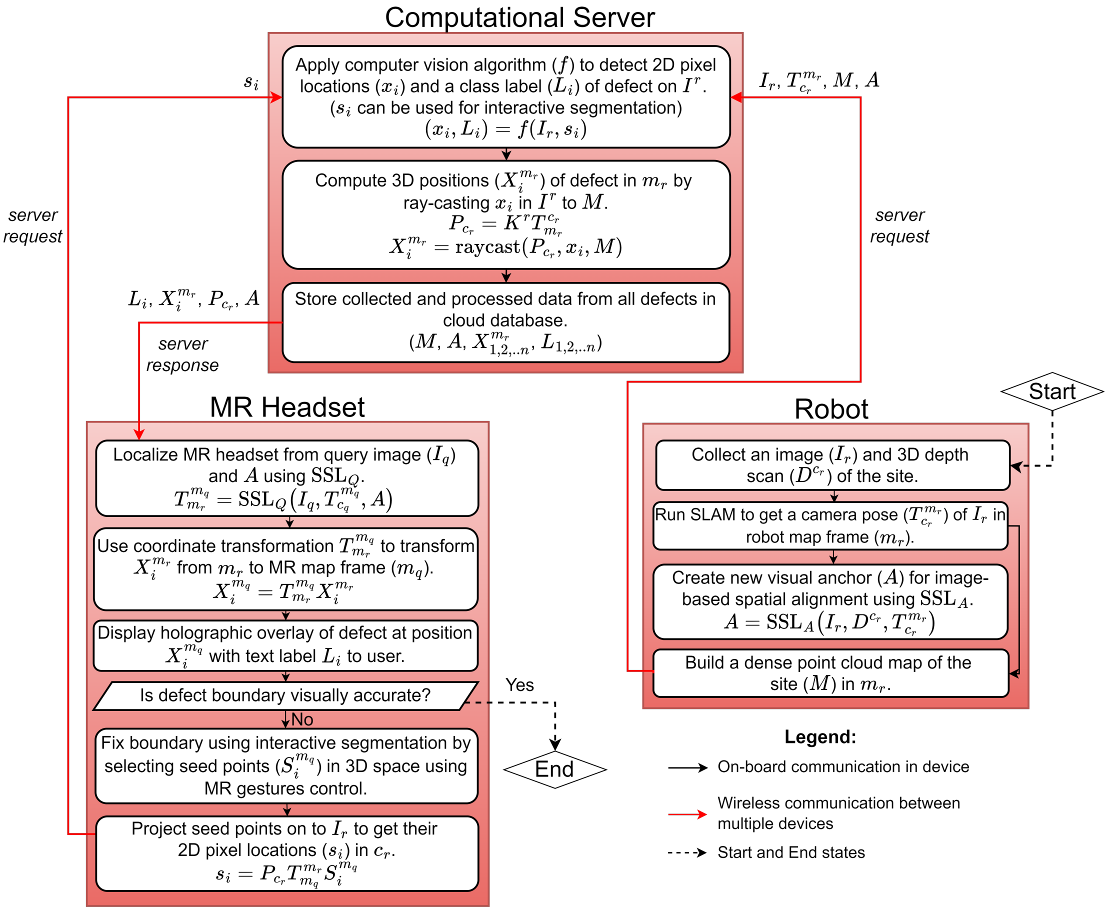

Computational Server
Apply computer vision algorithm (f) to detect 2D pixel locations (x_i) and a class label (L_i) of defect on I^r. (s_i can be used for interactive segmentation) $(x_i, L_i) = f(I_r, s_i)$
Compute 3D positions ($X_i^{m_r}$) of defect in m_r by ray-casting x_i in I^r to M. $P_{c_r} = K^r T_{m_r}^{c_r}$ $X_i^{m_r} = \text{raycast}(P_{c_r}, x_i, M)$
Store collected and processed data from all defects in cloud database. $(M, A, X_{1,2,..n}^{m_r}, L_{1,2,..n})$
s_i
server request
I_r, $T_{c_r}^{m_r}$, M, A
server request
L_i, $X_i^{m_r}$, P_{c_r}, A
server response
MR Headset
Localize MR headset from query image (I_q) and A using SSL_Q. $T_{m_r}^{m_q} = \text{SSL}_Q(I_q, T_{c_q}^{m_q}, A)$
Use coordinate transformation $T_{m_r}^{m_q}$ to transform $X_i^{m_r}$ from m_r to MR map frame (m_q). $X_i^{m_q} = T_{m_r}^{m_q} X_i^{m_r}$
Display holographic overlay of defect at position $X_i^{m_q}$ with text label L_i to user.
Is defect boundary visually accurate?
Yes
No
End
Fix boundary using interactive segmentation by selecting seed points ($S_i^{m_q}$) in 3D space using MR gestures control.
Project seed points on to I_r to get their 2D pixel locations (s_i) in c_r. $s_i = P_{c_r} T_{m_q}^{m_r} S_i^{m_q}$
Start
Robot
Collect an image (I_r) and 3D depth scan (D^{c_r}) of the site.
Run SLAM to get a camera pose ($T_{c_r}^{m_r}$) of I_r in robot map frame (m_r).
Create new visual anchor (A) for image-based spatial alignment using SSL_A. $A = \text{SSL}_A(I_r, D^{c_r}, T_{c_r}^{m_r})$
Build a dense point cloud map of the site (M) in m_r.
Legend:
On-board communication in device
Wireless communication between multiple devices
Start and End states

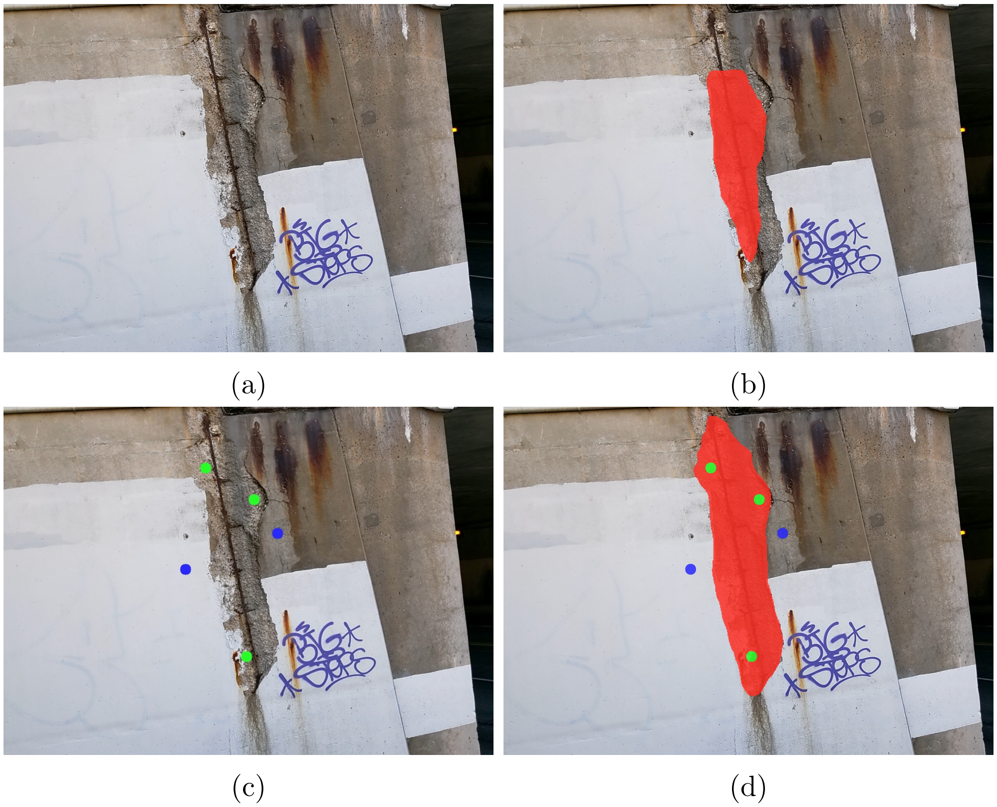
Figure 4.3: Process pipeline of the proposed human-robot collaboration.

building a 3D map of the scene ($\boldsymbol{M}$). For this step, the SLAM algorithm must incorporate depth information from the 3D sensor, such as RGB-D SLAM [103, 124] or visual-LiDAR SLAM [125, 126]. The output of the SLAM algorithm provides the camera pose ($T_{c_r}^{m_r}$) in an absolute scale relative to m_r. The robot then uses these reference data (I_r, D^{c_r}, $T_{c_r}^{m_r}$) to create a spatial anchor (A) using SSL described in Section 4.3, which is labeled as SSL_A in Fig. 4.3. The dense point cloud map (M), which is the output from SLAM is also stored for future inspection sessions.

Computational Server: The data collected from and processed in the robot such as I_r, $T_{c_r}^{m_r}$, A, and M are then sent to the server, shown as *server request* in Fig. 4.3, to start the analysis process. A pre-trained deep learning model (f) analyses I_r to locate defects in the image using an image-based semantic segmentation model. In this framework, f can be a supervised model (e.g. SegNet) that provides a boundary of a damage region automatically [127]. Alternatively, a semi-supervised model can be implemented to interactively improve its boundary, *developed by the myself* [46]. For interactive segmentation, users select seed points, s_i, that segment the entire image to extract the pixel-wise boundary (ROI) of the defect. f then outputs the 2D pixel coordinates (x_i) representing the defect boundary in the image and produces its class label (L_i) representing the type of defect (spalling, crack, etc.). The server then computes the 3D coordinates of the defect by back-projecting x_i to M using ray-casting. In this instance, ray-casting uses the projection matrix of the robot's camera, P_{c_r}, to produce a set of rays that start from the camera center of c_r and with direction vectors specified by x_i. The 3D coordinates of the defect's ROI ($\boldsymbol{X_i^{m_r}}$) in m_r are then computed by determining where the rays intersect with M. Both $X_i^{m_r}$ and L_i are then stored in a cloud database along with M and A so that inspectors can re-localize their MR headsets when revisiting the site in future inspections.

MR Headset: The server then sends $X_i^{m_r}$ and L_i to the MR headset, indicated as *server response* in Fig. 4.3, to allow the inspector to view the detected region and be able to check the results in real-time through visualization in the MR headset. The MR headset also receives the spatial anchor, A, to enable spatial alignment through image-based localization. The MR headset first captures a query image (I_q) for spatial alignment. The MR headset's internal visual-spatial tracking function also automatically provides the pose of the MR headset's camera ($T_{c_q}^{m_q}$) relative to the MR map frame (m_q) when I_q is captured. SSL$_Q$ then uses I_q and $T_{c_q}^{m_q}$ with the prior spatial anchor, A, created by the robot to obtain the transformation $T_{m_r}^{m_q}$ using Eq. 4.4. Once $T_{m_r}^{m_q}$ is obtained, I consider the MR headset's map frame (m_q) to be localized relative to the robot's map frame (m_r). Thus, 3D points ($X_i^{m_r}$) indicating defect boundary can be transformed from the robot map frame (m_r) to the MR map frame (m_q), so that any detected ROI can be visualized by the MR headset directly. $X_i^{m_r}$ is then multiplied with $T_{m_r}^{m_q}$ to obtain the 3D coordinates ($\boldsymbol{X_i^{m_2}}$) of the defect's ROI in the MR map frame (m_q). The MR headset then renders a holographic overlay over the location of the defect with the ROI specified by $X_i^{m_q}$. A holographic text overlay specified by L_i is also placed beside the hologram to inform the inspector about the type of defect. This process is repeated for every defect present in I_r, and defects present in other images captured by the robot's camera at different locations on-site.

However, in cases where the automated detection model does not produce an accurate boundary for the defect, like Fig. 4.4b, the inspector can fix the segmentation result using interactive segmentation enabled through the MR interface (Yes/No in the parallelogram in Fig. 4.3) [46]. Suppose that the robot captures the image presented in Fig. 4.4a and gets incomplete segmented results (red transparent region) like Fig. 4.4b from the automated method. Then, the inspector places virtual seed points by using the headset's hand gesture control inside (green) and outside (blue) the defect region, as shown in Fig. 4.4c. Finally, the algorithm automatically improves the segmentation accuracy and improves the quality of the segmentation results, as shown in Fig. 4.4d.



(a)

(b)

(c)

(d)

Figure 4.4: Interactive damage segmentation: An inspector with the MR headset can refine incorrect defect segmentation boundary in (b) processed by automated algorithms (applied to (a)) in the robot. The inspector selects the seed points inside (green) and outside (blue) the defect in (c) and segmentation with points can improve segmentation results in (d).

To perform interactive segmentation, the inspector first selects the 3D locations of the seed points ($S_i^{m_q}$), which are then anchored in the scene (M) to the defect region in the MR map frame (m_q) coordinate system (as seen in Fig. 4.4c). $S_i^{m_q}$ is then multiplied with $T_{m_q}^{m_r}$ (inverse of $T_{m_r}^{m_q}$) to transform the seed points' locations from the MR map frame (m_q) to the robot map frame (m_r). The seed points ($S_i^{m_r}$) are then projected to the image plane of the original image captured by the robot, I_r, using P_{c_r}, to obtain the 2D image pixel coordinates of the seed points (s_i) in I_r. The seed points, s_i, are then sent to the computational server, indicated as *server request* in Fig. 4.3, to start the segmentation process again. This process is iterative until the inspector decides that the segmented mask of the defect region is sufficiently accurate like Fig. 4.4d. With this process, the inspector can refine the results of the analysis and confirm the quality of the inspection results on-site.

Note that HMCI in this study is mainly designed for in-situ visual inspection where both the robot and MR headset are working in the same session and interacting in real-time. However, thisframework also allows inspectors to revisit the site and compare changes from previous inspections using the MR headset. Once the initial data collection is completed and the inspection data is collected and uploaded to the cloud, inspectors can re-load the data to their MR headset to locate the closest spatial anchor (A) from the database created in the previous sessions. Once A is located, the MR headset is then re-localized relative to the 3D map (M) created by the robot from the previous session, and all the stored annotations can then be visualized and overlayed in their original locations. This framework can also be expanded to multiple users with MR headsets to collaborate during the inspection once they are all localized relative to M.

4.5 Mixed Reality Interface for Robot Control

Another component of human-machine interaction between the MR user and the robot is the ability to directly control the robot in real-time using the MR interface. Once the MR headset and the robot are co-localized, spatial data can be sent between them in either direction. This also includes the MR user sending spatial data to the robot so that it execute certain tasks such as navigating to a desired location in the scene. Thus, a MR interface was developed for the user to be able to control the robot by selecting a target pose ($\boldsymbol{Y^{m_q}}$) indicating where the user would want the robot to be positioned. In this interface, the user selects Y^{m_q} by placing a MR hologram of the robot in the desired position. This is illustrated in Fig. 4.5. The implementation of this approach is shown in Fig. 4.6.

After the user selects Y^{m_q}, it is then sent to the computational server where the pose is multiplied by $T_{m_r}^{m_q}$ to obtain $\boldsymbol{Y^{m_r}}$, which transforms it from the m_q frame to the m_r frame. The robot will then navigate itself in m_r to attempt to match its current pose with Y^{m_r}. This method is compatible with ground robots where there are 3 Degrees of Freedom (DOF) and drones where there are 6 DOF.

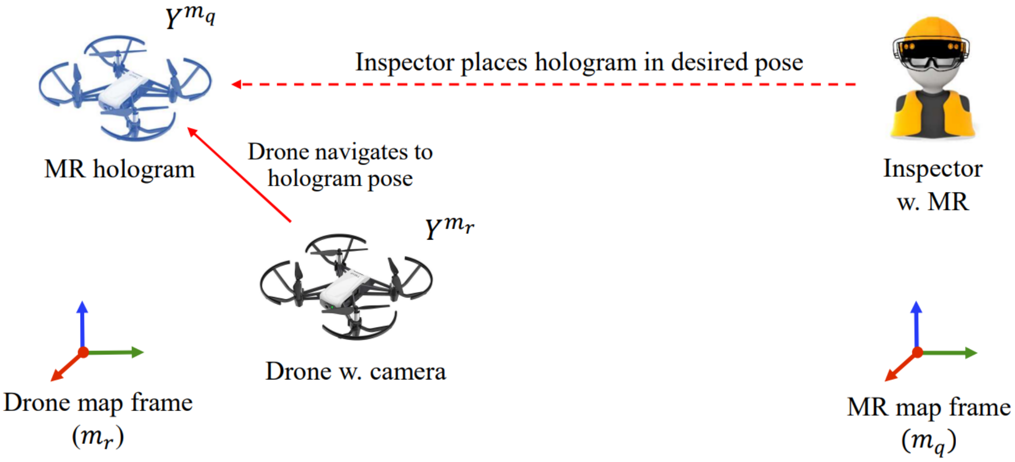


Figure 4.5: Illustration of MR interface for controlling the robot. The user places a hologram of the robot (drone in this case) in the specified pose, and the robot will try to match that pose in the real environment.

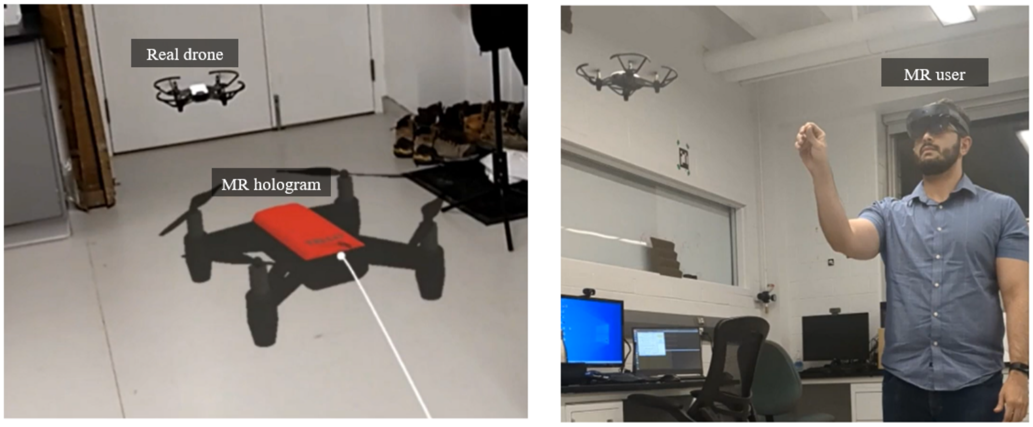
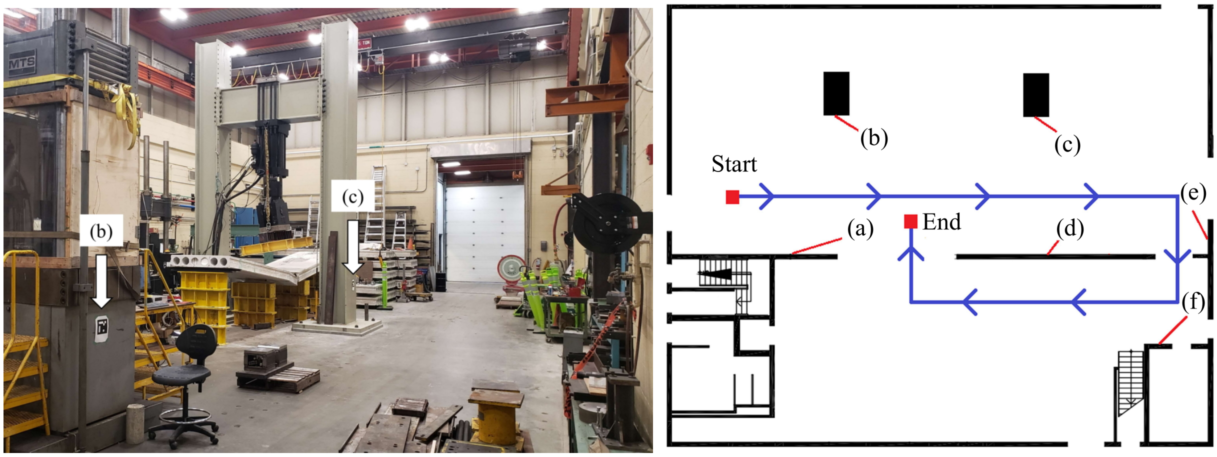


Figure 4.6: Implementation of MR interface for controlling the robot. The MR user (equipped with HL2) places the hologram in the desired pose, and the robot (DJI Tello drone) navigates the scene to match its current pose with the hologram.

4.6 Experimental Validation

4.6.1 Experimental setup



(a)

(b)

Figure 4.7: Test site (Structures Lab at the University of Waterloo): (a) site photo and (b) floor plan with a data collection path: A total of six markers are placed as ROIs as shown in Fig. 4.8.

Experiment validation of HMCI and the spatial alignment algorithm was conducted in the Structures Laboratory at the University of Waterloo (Structures Lab), shown in Fig. 4.7a. The size of the Structures Lab is 83 ft (25.3 m) × 68 ft (20.75 m) and its layout is present in Fig. 4.7b. The scene of the Structures Lab provides rich visual features in the scene which are necessary for spatial alignment. In Fig. 4.7b, the robot followed a pre-planned path and detected six targets (a)-(d) that are located at different locations across the test area. The robot traveled along the direction of the arrows from "Start," to "End" in Fig. 4.7b. Six fiducial (ArUco) markers of size 12 cm × 12 cm in Fig. 4.8 were placed as ROIs [128]. These markers constitute proxies for structural defects that can be automatically detected using computer vision algorithms. These six markers were taped to flat surfaces on walls or columns, as shown in Fig. 4.8, at various heights ranging from 40 cm to 180 cm above the ground.

Note that I would like to clarify the scope of the proposed experimental study. In fact, marker detection is not computationally expensive and can be conducted in the MR headset onboard. This means markers could be detected and localized without the integration

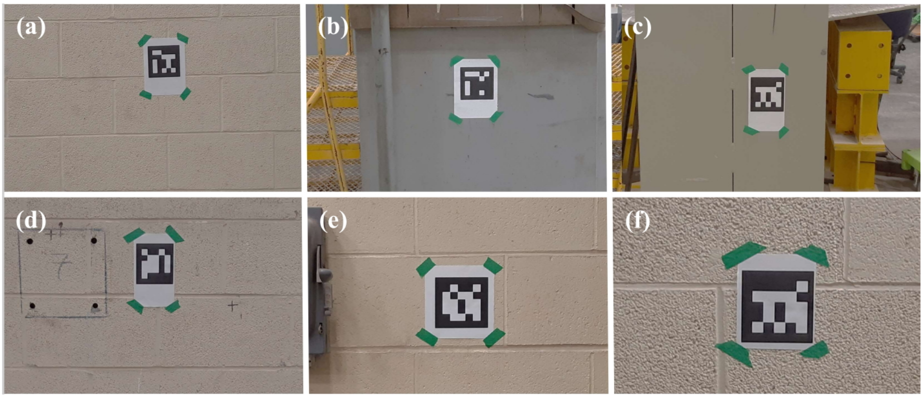
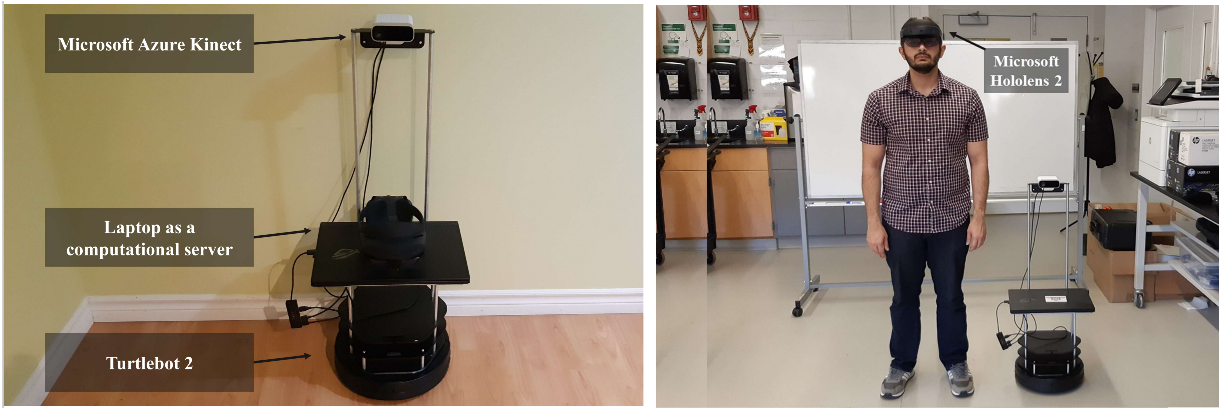


Figure 4.8: Fiducial markers (a)-(f) taped on various locations at the Structures Lab: Note that these markers are proxies for structural defects in real-world applications envisioned for HMCI.

of the robot and computational server. However, this experimental demonstration is to show how the robot and computational server off-site can support the inspection process of the human engineer with the MR headset. In an actual scenario, damage detection and segmentation are conducted using large computer vision models (e.g., convolutional neural network), which are computationally expensive processes and cannot be conducted in the MR headset rapidly and efficiently. In addition, typical MR headsets (e.g., HL2) support only a limited range for data collection and pose estimation (<5m), so LiDAR, which has a maximum 3D sensing range of 450 m, and high-resolution cameras might need to be deployed onboard the robot to conduct long-range inspections for real-world structures [105, 129, 130]. High-quality sensors also require higher computational power for data analysis, which means a higher-cost system. In this experiment, I have focused on demonstrating the entire workflow of the HMCI system and its scalability and efficiency. Regions-of-interest in actual inspections are represented by fiducial markers as a case study. The proposed system can be used in conjunction with any damage quantification algorithm, such as the interactive defect quantification tool developed by the myself [46].

4.6.2 Implementation of HMCI

A Turtlebot2 [131] equipped with an Azure Kinect RGB-D [132] sensor was used as the robot for data collection, while the HL2 was used as the MR headset for the experiment. The Azure Kinect can capture color images with a resolution of 1,280 × 720 pixels, while the HL2's camera can capture color images with a resolution of 2,272 × 1,278 pixels in the



(a)

(b)

Figure 4.9: Hardware used in the experiment: (a) robotic data collection platform and (b) a user (inspector) with the MR headset (HL2).

video mode. The Azure Kinect's depth sensor has a resolution of 512 × 512 pixels, and the operating range is between 0.25m to 2.88 m [132]. Since the depth resolution is lower than the image resolutions, the depth images were upscaled by interpolating the depth values for the missing pixels. The Azure Kinect was attached to the Turtlebot2 at the height of 1 m above the top plate of Turtlebot2 to secure sufficient field-of-view for inspections. In this setup, a laptop (ROG Strix G17, equipped with an NVIDIA GeForce RTX 2060 GPU and an Intel Core i7 processor) functioned as both the processing unit on the robot and the computational server to analyze data. All the hardware used in the experiment is present in Fig. 4.9a, and a user (inspector) wearing HL2 standing beside the robot is shown in Fig. 4.9b.

The laptop was connected through USB cables to the Turtlebot2 and Azure Kinect camera. The computational server running on the laptop was wirelessly connected to the HL2 through a shared Wi-Fi network. For a field inspection scenario, either 4G LTE or 5G should be available to allow the robot, MR headset, and remote server to be connected remotely. This setup facilitated the two-way transfer of data between the two devices (as described in Fig. 4.3). A Rosbridge server was chosen to be the edge server in this configuration because it allows devices that do not support Robot Operating System (ROS) (e.g., HL2) to communicate with ROS-based robots [133]. To develop the MR application, the Unity game engine, and the Mixed Reality Toolkit (MRTK) were selected because of their compatibility with other popular MR headsets, including Microsoft's HoloLens series, and Magic Leap [134, 135, 136]. The Unity-based application running on the HL2 sends and receives messages from the Rosbridge server over a local network using the ROS-sharp package [137]. The Rosbridge server used persistent TCP/IP connections to transport JSON messages that encode texts, images, and point cloud streams.

In the HMCI workflow described in Section 4.4, the TurtleBot2 first gathered color (I_r) and depth (D^{c_r}) images of the scene using the Azure Kinect sensor and performed RGB-D SLAM, RTAB-MAP, to continuously track the pose of the camera (c_r) relative to the robot map frame (m_r) and to create spatial anchors (A) [124]. At the same time, the HL2 performed localization by capturing query images (I_q) from its equipped camera (c_q) by automatically finding nearby anchors and localizing itself relative to m_r. For the detection of ROIs, a lightweight ArUco marker detection algorithm was running onboard the laptop, which is implemented in OpenCV to continuously analyze every image frame produced by the robot's camera [104].

In thisimplementation of SSL, the TurtleBot2 and HL2 automatically initiate the co-registration process whenever a target defect is located. The HL2 then automatically searches through all previously created spatial anchors in the same session to localize itself relative to the Turtlebot2 using SSL. Robust feature detection and matching algorithms are required for accurate pose estimation of the HL2 headset using spatial anchors. In this experiment, I first evaluate four different feature detection algorithms including ORB [138], SIFT [139], R2D2 [140], and SuperPoint [141] in the SSL framework. The performance of the localization is directly related to the accuracy of ROI visualization in the HL2 headset. Next, for the feature matching, I tested two algorithms, the nearest neighbor search (NN) [142] and SuperGlue [143]. For NN, the number of classes (k) is set to k=2 to find the best matches through brute force. For ORB, SIFT, and R2D2 feature types, a distance ratio test with the threshold of 0.75 (ORB, SIFT) and 0.8 (R2D2) was used to filter out incorrect matches. For SuperPoint, a distance threshold of 0.7 was used instead of a ratio test. SuperGlue uses a pre-trained graph neural network (outdoors pre-trained model) to match SuperPoint features [144]. Thus, SuperGlue is only used for matching SuperPoint features. These feature detection and matching algorithms are all implemented in the Hierarchical Localization toolbox, while parameters other than thresholds are set to default values [67]. Lastly, I set the minimum allowable number of feature matches between the I_q and I_r to 10. The function in OpenCV, *solvePnPRansac* was deployed to conduct P3P RANSAC where the maximum iterations and allowable re-projection error are set to 100 guesses and 0.8 pixels, respectively [104].

4.6.3 Spatial alignment error measurement

I evaluated feature detection and matching algorithms for spatial alignment by measuring the 3D physical offset distance (e) of the marker's visualized location (X^M) compared to the marker's ground-truth location (Y^M). Ideally, if the HL2 is precisely aligned with the TurteBot2 spatially, then e should be zero. However, depending on the background textures of the scene, the performance of feature detection and matching algorithms varies, followed by varying ROI localization accuracy. In this experiment, I quantitatively evaluated the spatial alignment errors (e) depending on feature detection and matching algorithms that I used for SSL. The evaluation process of the spatial alignment is illustrated in Fig. 4.10.

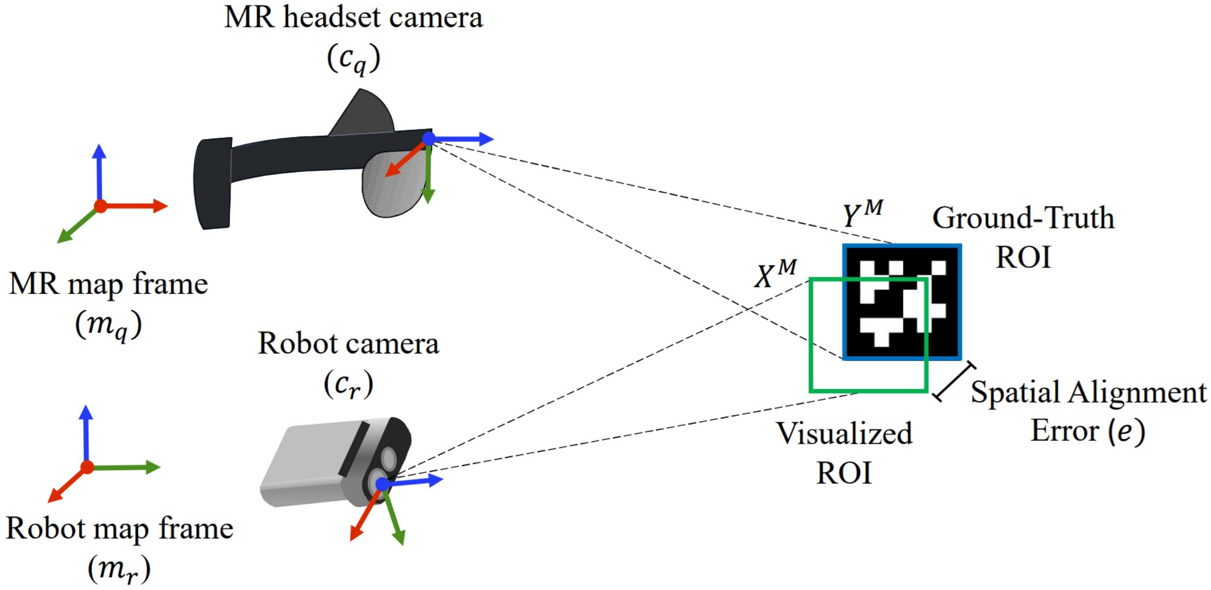


Figure 4.10: Illustration of spatial alignment error (e) measurement process: X^M is the visualized ROI (marker boundary) measured from the robot's sensors (c_r) and visualized by the MR user when a defect is located. Y^M is the ground-truth marker location obtained from I_q for directly localizing the marker's ROI. Note that X^M and Y^M are in the robot map frame and represented by a physical unit.

Once the marker ROI is visualized in the HL2, the 3D locations of the marker detected by the Turtlebot2 (X^{m_r}) are projected on the HL2 image (I_q) to get their 2D image pixel locations (x^M) in I_q using the pose of the HL2 (projection matrix) estimated through SSL. The ground-truth ROI (y^M) is then computed directly from I_q using the marker detector so that it can be compared against x^M. To convert x^M and y^M to physical coordinates, a four-point homography algorithm was used to calculate the homography matrix (H) using marker dimensions (12 cm × 12 cm). $x^M$and y^M were then transformed using H to their real physical coordinates, X^M and Y^M, respectively. Homography was used in this procedure to get an image scale and evaluate e in a physical unit (e.g., cm, inch). e was then computed by subtracting X^M from Y^M to calculate the Euclidean distance between them and average those error values from multiple test images (I_q). Five sample query images from each marker were captured from a 1-2 m standoff distance, which becomes a total of 30 images (six markers) collected from the HL2 in each test.

4.6.4 Experimental results

HMCI demonstration

Major steps of HMCI in Section 4.4 are experimentally demonstrated in this subsection including reconstructing the 3D map of the site, localizing defects onto the map, localizing the MR headset relative to the map using the SSL, and then visualizing the defect ROIs on the MR headset.

The process of scanning the Structures Lab using the TurtleBot and Azure Kinect took approximately 10 minutes. The robot was remotely controlled by the MR-equipped inspector using a handheld controller to follow the path shown in Fig. 4.7b All defects (markers) were automatically localized once they were in the field-of-view of the camera, and the resulting 3D map was reconstructed in real-time. The map was built by fusing the camera images and depth scans obtained from the Azure Kinect, and the camera poses were estimated using RTAB-MAP. These scans were then combined using the estimated camera poses and colorized using the color images, to produce a full point cloud map of the Structures Lab. Fig. 4.11 shows the colorized point cloud of the Structures Lab and the dotted line in light blue indicates the robot's trajectory.

The ROIs (markers in this experiment) were then detected from the 2D images collected from the robot, and associated with their labels (fiducial marker ID). The 2D corners detected were then projected onto the 3D maps using the ray-casting process described in Section 4.4 to obtain the 3D coordinates of the ROIs. Fig. 4.12 visualizes ROIs and marker IDs which are overlaid onto the 3D maps. In the actual scenario, equivalently, the damage boundaries and their classes will be detected.

Once the spatial alignment is completed between the robot and HL2, the HL2 receives the information about the 3D coordinates of the ROIs and their labels. The HL2 then displays a hologram on the real scene for the inspector to visualize the location of the

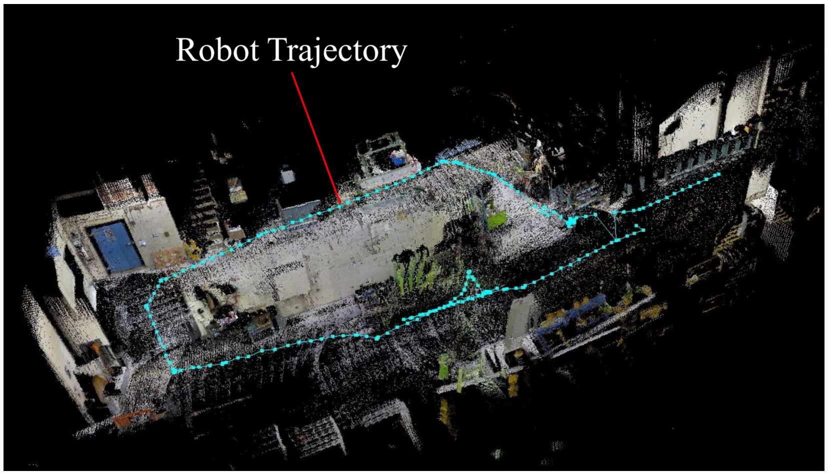


Figure 4.11: A colorized point cloud map of the Structure Lab: Images and depth scans were collected from the Azure Kinect RGB-D camera and RTAB-MAP SLAM was used to produce this 3D map. The light blue dotted line indicates the robot's data collection path.

defects and their labels. The images in Fig. 4.13 were captured from each ROI where the boundary and its ID are overlaid.

Fig. 4.14 shows the scene of how the inspector interacts with the graphic overlay through hand gestures. This scene was captured from a secondary device that is spatially aligned using Spectator View [145]. The inspector can select the ROI and modify its size and location in case this information is not accurate. Also, the inspector can correct and add text overlay for documentation.

Performance of feature detection and matching for spatial alignment

Table 4.1 presents the experimental results of the spatial alignment error (e) evaluation for each of the feature detection and matching configurations for SSL. The first column shows the mean spatial alignment errors and their standard deviation in a physical unit. The mean time per query is the time to process a query image and localize the HL2 relative to the robot. This statistic is important for processing more queries and staying aligned with the robot's map frame. I also evaluate the performance of feature matching because it is directly related to spatial alignment. The third column shows the mean number of matched features per query image in the test. When the number of matched features is small (10 in this experiment), the alignment likely fails so I excluded those cases for error

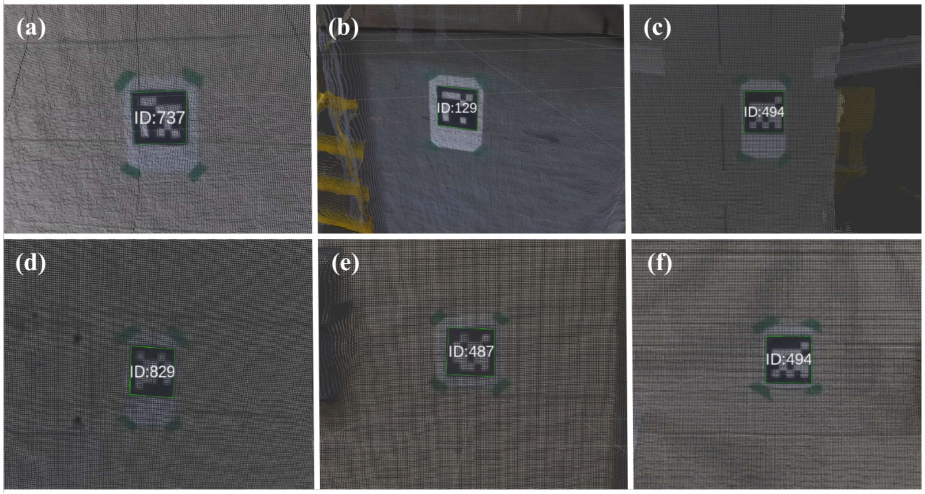


Figure 4.12: Visualization of the detected ROI in the 3D maps from the robot: The boundary (ROI) of each marker in green is displayed on the 3D map. The text overlay shows the type of defect (fiducial marker ID).

measurement in the first column. Instead, I evaluate them as pass/fail ratio in the fourth column.

In Table 4.1, experiment results show that all the feature detection and matching algorithms produced spatial alignment errors of around 1 cm. The R2D2 + NN configuration produced the best overall accuracy results of 1.08 cm. However, only the SIFT + NN and SuperPoint + SuperGlue strategies managed to perform successful localization of the HL2 for all 30 sample query images with no failure requests and with standard deviations less than 1 cm, making them the most consistent and reliable implementations. Also, SuperPoint + SuperGlue managed to produce the most matched feature points on average (371 matches) between query and reference images. This is because CNN-based methods such as SuperPoint and R2D2 can detect more features even in low or repeating texture environments such as flat surface walls. They use the full image as input for their CNN backbone, which is compared to patch-based methods such as SIFT or ORB. Thus, both SuperPoint and R2D2 can better contextualize the locations of low texture regions in the image. However, SuperGlue outperforms R2D2 overall because R2D2 prioritizes the quality of detected features over the number of features, which can be detrimental if not enough features are detected and matched with other images. This is evident in the fact that 7% of query images found less than 10 matches between reference and query image for the R2D2 + NN configuration, and thus, localization could not be performed for those cases.

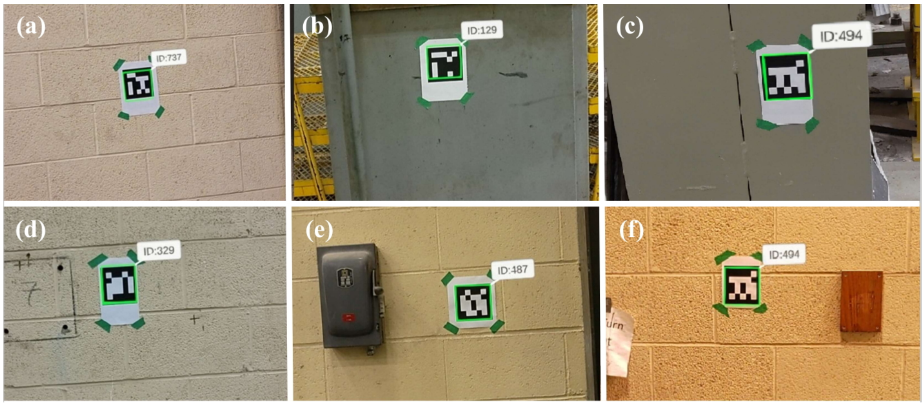


Figure 4.13: Overlaying the ROI of each defect in the scene of the HL2: These images show that all six defects detected by the TurtleBot2 are successfully localized in both the robot and MR map frame.

SuperGlue matching has been shown to produce the largest number of feature point matches as can be seen in Fig. 4.15. This is because SuperGlue uses graph neural networks to attempt to learn the underlying 3D scene when matching two sets of features points. Thus, it outperforms traditional purely descriptor-based methods such as Nearest Neighbour which may be too strict when rejecting potential matches. However, since SuperGlue uses CNN networks to compute matches, it has a longer lead time compared to Nearest Neighbour. Nevertheless, in civil engineering applications, bridges or walls that were built using concrete or bricks often have insufficient unique features due to repeating textures. In this sense, the configuration of SuperPoint + SuperGlue is optimal for detecting and matching feature points when performing image-based localization in these challenging environments. Note that sample scenes from the HL2 in Fig. 4.15a were captured after estimating its pose using SuperPoint + SuperGlue.

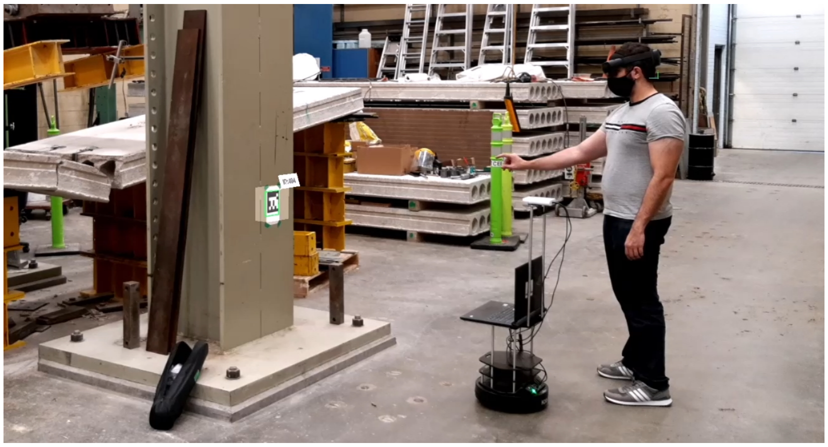
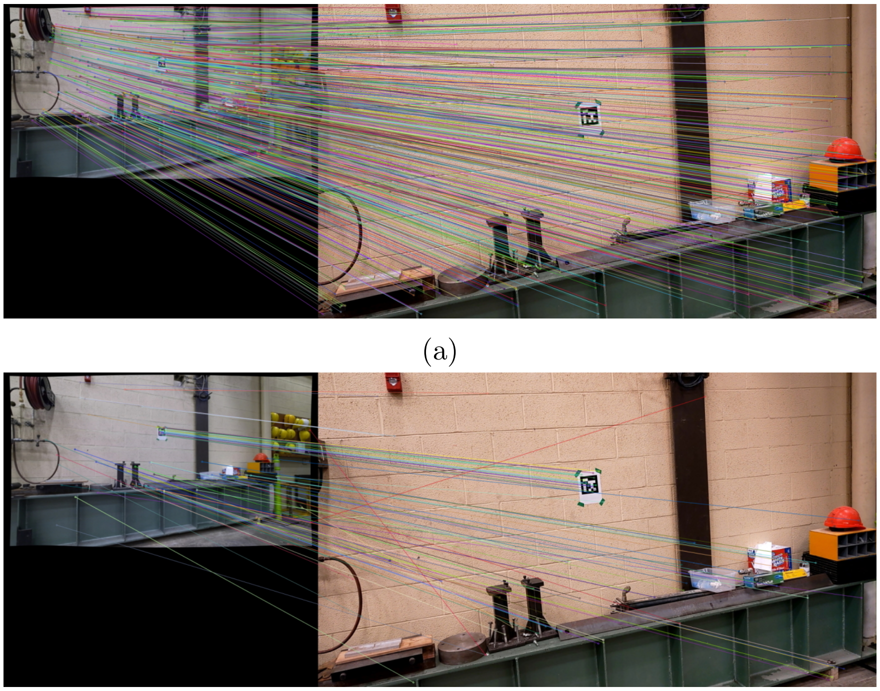


Figure 4.14: The scene of the HMCI demonstration was captured by a secondary device, which is spatially aligned with the scene.

Table 4.1: Comparison of the spatial alignment errors depending on feature detection and matching algorithms implemented in SSL.

Feature detector + matcher	Mean spatial alignment error ± standard deviation	Mean process time per query	Mean # of matches per query	Ratio of successful queries
ORB + NN	1.5 ± 1.7 cm	0.44 s	35	77%
SIFT + NN	1.2 ± 0.9 cm	1.70 s	81	100%
R2D2 + NN	1.1 ± 1.1 cm	1.17 s	78	93%
SuperPoint + NN	1.1 ± 0.9 cm	0.65 s	239	90%
SuperPoint + SuperGlue	1.1 ± 0.9 cm	2.23 s	371	100%



(a)

(b)

Figure 4.15: Sample feature detection and matching results using (a) SuperPoint + SuperGlue and (b) SIFT + NN: 2D feature points from the robot's camera (left) are matched with the ones from the query image from the HL2's camera (right) used for SSL. Each pair of feature point matches is shown as a line between the two images.

Chapter 5

Distributed Collaboration through Mixed/Virtual Reality

5.1 Overview

In this chapter, a novel real-time distributed collaborative system where on- and off-site inspectors perform synchronous structural inspections through remote MR/VR collaboration is proposed as a component of SIM. This chapter also discusses integrating the previously discussed components of SIM such as human-AI and human-robot collaboration into a unified SIM system. This is accomplished by extending the methods already developed for MR users such as XRIV and HMCI to the VR user so that they can also interact and collaborate with robots and AI remotely. This chapter presents a summarized version of the work submitted to a journal paper [146].

5.2 Approach and contribution

In this chapter, distributed collaboration through SIM provides an advanced platform to perform collaborative inspections between multiple on-site MR-equipped inspectors and remote VR-equipped experts. The proposed system allows inspectors to interact with each other remotely and share information such as defect annotations in a spatially aware virtual environment. SIM is scalable and designed to support multiple MR/VR users present in physical or virtual environments. Also, I propose a novel multi-image-based spatial localization algorithm called MSL to determine the positions of multiple images with known relative poses. The proposed MSL can be used for localizing panoramic images by decomposing them into multiple perspective images. Panoramic images localized in the pre-built map provide the VR users with a 360° view of the site for making measurements of defect regions or changes in structural components. Fig. 5.1 shows screenshots of an experimental demonstration of distributed collaboration in this study. Two MR on-site inspectors (HL2) and one VR remote inspector (Oculus Quest 2) conduct a bridge inspection remotely using SIM. User locations are registered to the pre-built map and shared with other users, and all users are aware of what other users are doing and can view annotations or comments anchored on the structure or virtual environment.

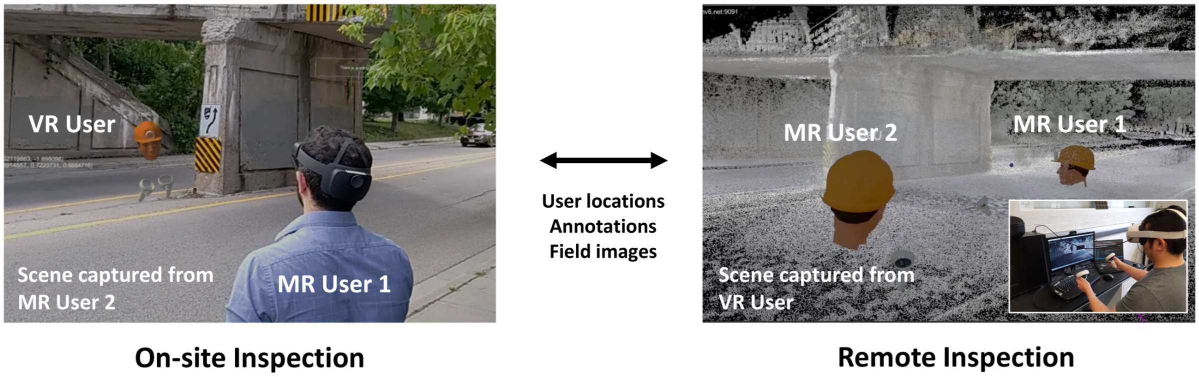


Figure 5.1: Proposed system to support distributed collaboration between on-site MR and remote VR users.

The idea of enabling structural inspections through the industrial metaverse, wherein on- and off-site users can concurrently and collaboratively execute inspection tasks is an innovative concept not yet covered in the literature. Current teleoperation systems primarily enable remote engineers to stream a video that the on-site inspector is looking at [78]. However, in such cases, remote engineers only rely on the image/video data captured by the inspector, posing a potential risk of overlooking crucial areas, mispositioning, or encountering low-quality data, not suitable for thorough inspection. Additionally, their communication relies solely on 2D video/image frames, restricting the information needed for interaction. On the other hand, SIM allows real-time communication between on-site inspectors and remote experts. These remote experts can ensure that the data is collected correctly and can share their comments and annotations with all inspectors in a shared environment. This novel 3D mode of communication in SIM offers more seamless and information-rich engagement and collaboration for infrastructure inspections.

5.3 Multi-shot localization

SSL has certain limitations when applied to SIM. SIM utilizes a panoramic camera to facilitate collaboration between MR and VR users. The issue with SSL is that it cannot be directly used to localize panoramic images because they do not follow a pinhole camera geometry. A panoramic image could be decomposed into multiple perspective images by converting the equirectangular projection to a cubemap projection, resulting in six perspective images with pinhole geometries [147]. However, SSL was designed to employ only a single query image to estimate the camera's pose. In fact, the accuracy of the spatial alignment requires multiple trustworthy feature matches between the query image and the anchor, followed by accurate pose estimates from the query device. Thus, SSL can only localize a single image out of the six perspective images at a time and cannot provide an accurate localization result that aligns the entire panoramic image with M. This means that other regions of the panoramic image would not be aligned correctly with M. Thus, an improved spatial alignment method is needed by incorporating multiple query images to localize panoramic images.

This study proposes a new image-based spatial alignment algorithm called MSL, which incorporates multiple query images with known relative poses for spatial localization. The main advantage of MSL is to use multiple query images and spatial anchors to estimate a coordinate transformation to minimize the overall re-projection error across all the query images and obtain a globally consistent spatial alignment result. In MSL, a reference device (r) equipped with a calibrated camera and a 3D sensor (e.g., LiDAR) captures N images ($\boldsymbol{I_{rj}}$) of the scene, denoted with index j for each reference image. I assume that all the images were captured with the same reference camera with identical intrinsic camera parameters (e.g., principal points, focal length), and the camera was preliminarily calibrated to obtain the camera intrinsic matrix (K^r). Each reference image has its camera coordinate system (reference camera frame) denoted as $\boldsymbol{c_{rj}}$. The reference device is also able to localize itself (using SLAM) to simultaneously build M and track its pose relative to its map frame ($\boldsymbol{m_r}$). The pose of the camera is calculated using the 4×4 coordinate transformation matrix ($T_{m_r}^{c_{rj}}$) that relates to the c_{rj} to m_r for each I_{rj}.

Next, a spatial anchor from I_{rj} is created. The process of creating spatial anchors in MSL is the same as SSL. First, an image retrieval algorithm decomposes I_{rj} into a bag-of-visual-words vector (V_{rj}) for performing image retrieval. A feature detection algorithm is then used to find unique 2D features (x_k^{rj}) and descriptors (D_k^{rj}) in I_{rj}, denoted with index k for each feature point extracted from I_{rj}. These reference features and descriptors are matched with features and descriptors in query images during the localization process. The 2D pixel coordinates of x_k^{rj} are then back projected from I_{rj} to 3D coordinates ($X_k^{c_{rj}}$) in the

c_{rj} frame. This is done by using the 3D sensor on-board of the reference device to obtain corresponding 3D points. Therefore, I can map every 2D point in x_k^{rj} to a corresponding 3D point in $X_k^{c_{rj}}$. The 3D points, $X_k^{m_r}$, corresponding to $X_k^{c_{rj}}$ in the m_r frame is computed from pre-multiplying $X_k^{c_{rj}}$ by $T_{c_{rj}}^{m_r}$, which is the inverse coordinate transformation of $T_{m_r}^{c_{rj}}$. The points in $X_k^{m_r}$ denote the 3D locations of visual feature points in m_r. In MSL, a spatial anchor (A_j) in Eq. 5.1 is encoded by a set of data computed from I_{rj} , which stores V_{rj}, $X_k^{m_r}$, and D_k^{rj}.

$$A_j = \{(X_k^{m_r},\ D_k^{rj})_{k=1}, V_{rj}\} \tag{5.1}$$

In the process of scanning the structure, the reference device creates many spatial anchors from the images that it collects so that it can encode visual information about different parts of the scene. A database is then constructed to store the spatial anchors (A_1, A_2, $\ldots$ A_N).

Once I create spatial anchors as pre-processing, I am ready to align the query devices. The query device (q) captures multiple query images ($\boldsymbol{I_{qi}}$) of the scene, where each query image is denoted with index i. Each query image also has its camera frame denoted as $\boldsymbol{c_{qi}}$. In MSL, the query images do not have to be captured with the same camera and can have different intrinsic parameters such as a different camera matrix (K^{qi}) per query image. The query device also tracks its position relative to a local stationary frame, denoted as $\boldsymbol{m_q}$. The pose of the query device relative to m_q is represented using the transformation $T_{m_q}^{c_{qi}}$ that is recorded for when each I_{qi} was captured.

In MSL, image retrieval is first used to obtain a set of spatial anchors that have the same visual features from I_{qi}. A bag-of-words vector (V_{qi}) for I_{qi} is computed, and V_{qi} is then compared to V_{rj} for every A_j in the database to compute similarity scores. The set of spatial anchors with high similarity scores is then used to perform the localization process. A predefined minimum similarity score is chosen to determine the number of spatial anchors retrieved for each query image. The specific value of this minimum similarity score can be determined through experiment, considering the desired robustness of the localization outcome. A higher number of retrieved spatial anchors can improve the localization result in cases where visual features in I_{qi} overlap with multiple spatial anchors, increasing the number of feature matches. However, retrieving too many spatial anchors may lead to situations where not all of the anchors have visual overlap with I_{qi}. This could potentially introduce false positive matches that can lower localization accuracy. Therefore, determining the optimal number of anchors retrieved per query image requires a pre-selection process.

Feature detection is then applied to I_{qi} to obtain 2D feature points (x_k^{qi}) and descriptors

(D_k^{qi}) for each query image. This process compares the visual descriptors of D_k^{qi} and D_k^{rj} to find correspondence (p_{ik}, p_{jk}) between x_k^{qi} and x_k^{rj}. Here, p_{jk} represents the index of matched feature point k in I_{rj} to refer to the point coordinates in x_k^{rj}, while p_{ik} represents the index of matched feature point k in I_{qi}. Thus, the 2D points in $x_{p_{ik}}^{qi}$ and $x_{p_{jk}}^{rj}$ represent the 3D point in the scene but viewed from different images I_{qi} and I_{rj}, respectively.

A set of 2D-3D correspondences is then established by linking the 2D points in $x_{p_{ik}}^{qi}$ with the 3D points in $X_{p_{jk}}^{m_r}$, since both sets of points represent the same locations in the 3D scene. $X_{p_{jk}}^{m_r}$ can be projected onto the image plane of I_{qi} to obtain a projected set of 2D points, denoted as $\tilde{x}_{p_{ik}}^{qi}$, using Eq. 5.2:

$$\tilde{x}_{p_{jk}}^{qi} \;=\; K^{qi}\, T_{m_q}^{c_{qi}}\, T_{m_r}^{m_q} X_{p_{jk}}^{m_r} \tag{5.2}$$

In Eq. 5.2, $X_{p_{jk}}^{m_r}$ in the m_r frame are first transformed to the points in the c_{qi} frame using the coordinate transformations of $T_{m_r}^{m_q}$ and $T_{m_q}^{c_{qi}}$. K^{qi} is then used to project those 3D points in the c_{qi} frame to I_{qi} to obtain $\tilde{x}_{p_{jk}}^{qi}$. Spatial alignment is the process of computing the transformation between m_q and m_r, denoted as $T_{m_r}^{m_q}$ by minimizing the distance between$x_{p_{ik}}^{qi}$ and $\tilde{x}_{p_{jk}}^{qi}$, which indicates the same 3D point, $X_{p_{jk}}^{m_r}$. Some of these variables and their geometric relations to each other are graphically represented as a frames diagram in Fig. 5.2:

In Fig. 5.2, the solid red lines represent known transformations, while the dashed line represents the unknown spatial alignment transformation ($T_{m_r}^{m_q}$). The query images are shown from left to right ($qi = q1,\ q2,\ \ldots$). Some query images may either include a single anchor, or multiple anchors at the same time. The re-projection errors (e_{sum}) between each pair of points in $x_{p_{ik}}^{qi}$ and $\tilde{x}_{p_{jk}}^{qi}$ from all query images and anchors are calculated in Eq. 5.3. The objective of MSL is to solve the optimization problem to estimate $T_{m_r}^{m_q}$ that can minimize the re-projection error between $x_{p_{ik}}^{qi}$ and $\tilde{x}_{p_{jk}}^{qi}$ across all combinations of i (query image), j (anchor), and k (correspondence). This problem can be formulated mathematically as a convex optimization problem in Eq. 5.3 to minimize e_{sum} by varying $T_{m_r}^{m_q}$:

$$e_{sum} = \sum_i \sum_j \sum_k \left|\; x_{p_{ik}}^{qi} \;-\; K^{qi}\, T_{m_q}^{c_{qi}}\, T_{m_r}^{m_q}\, X_{p_{jk}}^{m_r} \;\right| \tag{5.3}$$

This formulation is similar to the PnP problem [148]. Thus, the RANdom SAmple Consensus (RANSAC) algorithm is used to solve for $T_{m_r}^{m_q}$ [149]. In RANSAC, the problem is re-formulated to maximize the number of inlier points ($N_{inliers}$) that have re-projection

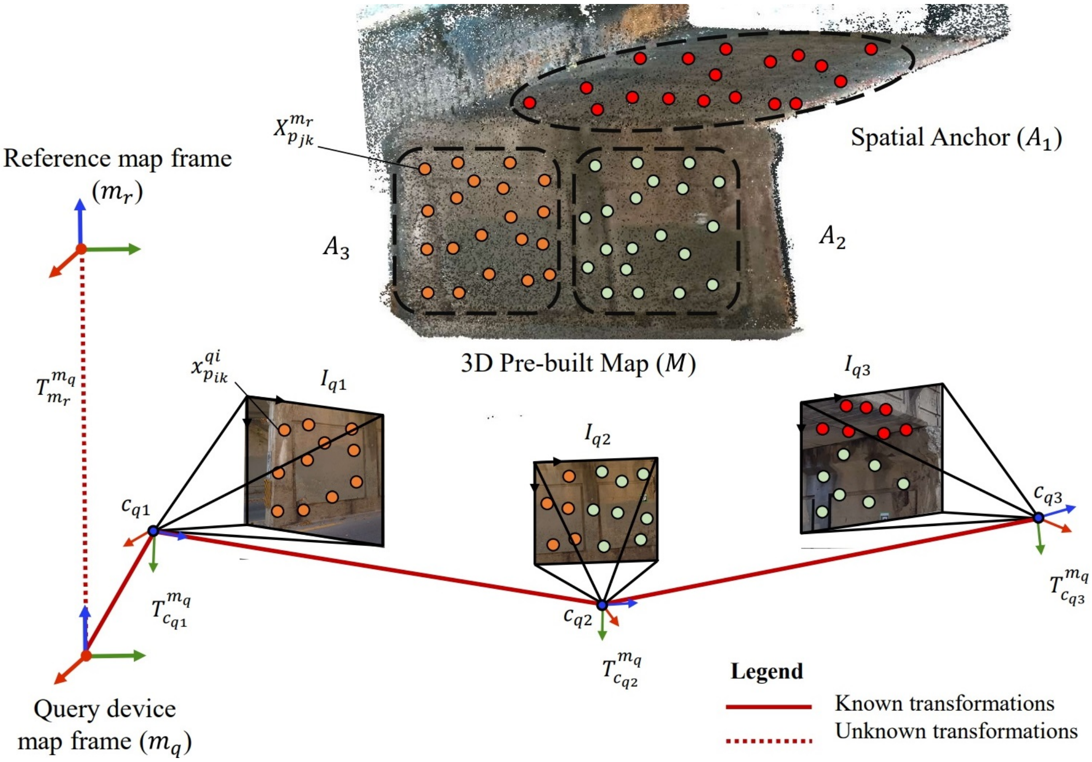


Figure 5.2: Geometric relationship between query images and anchors: Each spatial anchor (A_j) is visually represented as a set of colored 3D points anchored to the 3D map (M). The spatial alignment is to compute a transformation between the query device map frame (m_q) and the reference map frame (m_r). Note that adjacent camera transformations are obtained through SLAM for a single camera, or a camera calibration process for multiple cameras.

errors below a predefined threshold (e_{tol}). In MSL, first, four random feature points are picked in a random query image to obtain a solution for $T_{m_r}^{c_{qi}}$ that satisfies a P3P condition [121]. $T_{c_{qi}}^{m_q}$ is then multiplied with $T_{m_r}^{c_{qi}}$ to obtain an initial solution for $T_{m_r}^{m_q}$. The re-projection error is then calculated for all feature points and then compared with e_{tol} to calculate $N_{inliers}$. After a certain number of iterations, $T_{m_r}^{m_q}$ with the highest $N_{inliers}$ is selected that can minimize the re-projection error for the majority of matched feature points across all query images.

A major assumption of MSL is that the $T_{m_q}^{c_{qi}}$ transformations are highly accurate and have almost no drift. This assumption is required to guarantee that $T_{m_r}^{m_q}$ is accurate enough to align all of the query images simultaneously relative to m_r. Otherwise, drift errors in

$T^{c_{qi}}_{m_q}$ would cause only one query image at most to be localized accurately, while the reprojection error in the rest of the query images would be above e_{tol}. In such a case, MSL would have a similar level of accuracy as SSL.

There are two promising applications for MSL. In the first case, multiple query images are captured from a single camera, but at different time instances. In this case, the query device uses the SLAM algorithm to track its camera poses relative to m_q for each image. The accuracy of $T^{c_{qi}}_{m_q}$ must be maintained by using a highly accurate SLAM algorithm or other external sensors to track the camera's pose. The camera also has the same camera matrix (K^{qi}) for all the query images since the camera's intrinsic parameters do not change. In this case, MSL is used to spatially align the local SLAM stationary frame (m_q) to m_r.

For the second application case, there can be multiple cameras that are attached to a fixed rig. All cameras capture one image individually at precisely the same time. The images from these cameras in the rig are then used as query images for localization in MSL. In this case, the local camera poses required for MSL are obtained from extrinsically calibrating the cameras. In other words, the extrinsic transformations ($T^{c_{qi}}_{m_q}$) are known between each camera frame (c_{qi}) relative to an arbitrary stationary camera rig frame, which is assumed as m_q. Thus, MSL is used to spatially align the camera rig frame (m_q) to m_r. The cameras can also have different K^{qi} values since not all cameras are associated with the same intrinsic parameters.

Comparing the proposed MSL with SSL, MSL improves upon the SSL algorithm because it accounts for multiple query images and spatial anchors and provides an optimal solution that minimizes error for all of the query images simultaneously. As a result, the estimation of $T^{m_q}_{m_r}$ is more robust and accurate. It is noteworthy that MSL is not a process of simply repeating SSL multiple times. If SSL was simply applied to all the query images independently, then any error in the localization results would mean that $T^{m_q}_{m_r}$ would not be consistent with $T^{c_{qi}}_{m_q}$ for every other query image. This is because in SSL the result form $T^{m_q}_{m_r}$ is only valid for one query image at a time. Therefore, SSL does not allow for finding the optimal $T^{m_q}_{m_r}$ that aligns all of the query images simultaneously. SSL cannot guarantee that all of the query images will be successfully localized if some of them do not have visual overlap with spatial anchors. In contrast, MSL can still find a localization solution if at least some of the query images have visual overlap with spatial anchors.

One of the key advantages of MSL within the proposed SIM system is its adaptability to various camera models. For instance, it can handle a generalized camera model decomposed into a rig of multiple pinhole cameras with known transformations between them. These transformed images can then serve as input for MSL, enabling the localization of panoramic images to m_r.

5.3.1 MSL application: panoramic image localization

In the proposed SIM system, panoramic images are sent to the remote VR user with an all-around view of the site in real-time. These images serve multiple purposes, not limited to mere observation; they facilitate making measurements and annotating inspection items for the VR user. Thus, it is necessary to localize each panoramic image relative to the pre-built map so that annotations placed on the panoramic images are anchored to the pre-built map and shared with other users with spatial context.

To address this issue, MSL is used for localizing panoramic images. The panoramic images are decomposed into multiple perspective images and these images become query images in MSL. A panoramic image (I_q) is decomposed into a set of four non-overlapping, undistorted, square, perspective images (I_{qi}) using the cubemap projection [147]. Each image in I_{qi} is viewing the left, right, forwards, and backwards directions of the panoramic image. For this study, the up and down perspectives direct to sky and ground so they are not included as query images due to lack of unique features. An example of the cubemap projection can be seen in Fig. 5.3:

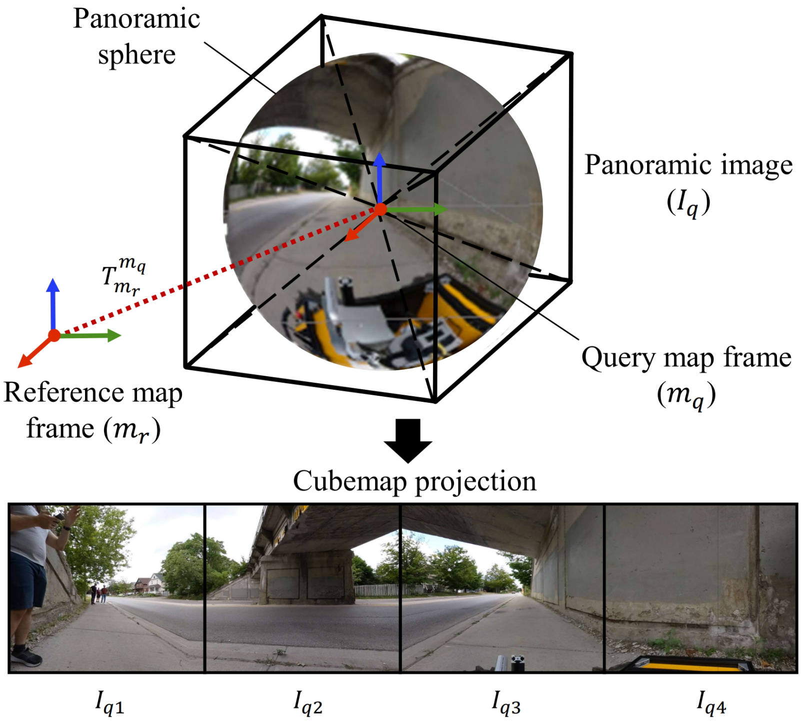


Figure 5.3: Converting panoramic image (I_q) to the cubemap projection to produce four square perspective query images (I_{qi}) for MSL for spatial alignment.

Assume that there exists a camera rig consisting of four cameras capturing each image in I_{qi} simultaneously. MSL is used to spatially align the camera rig frame (m_q) to m_r, where m_q is the center of the panoramic sphere facing the local forward direction. The extrinsic transformations ($T_{m_q}^{c_{qi}}$) are known between each camera frame (c_{qi}) relative to m_q from the cubemap projection. In the cubemap projection, each of the four perspective images is captured from the center of the panoramic sphere and the directions of the principal axis are placed at 0º, 90º, 180º, and 270º with respect to the local y axis of the panoramic sphere. Thus, each $T_{m_q}^{c_{qi}}$ matrix is represented by a rotation matrix with a specific rotating angle for each I_{qi}. MSL uses a calibrated camera matrix for the query images. In the cubemap projection, K^{qi} is computed for an equivalent pinhole camera for each I_{qi}, where the camera center is placed at the center of the panoramic sphere. The focal length value in K^{qi} is calculated from the field-of-view (fov) of I_{qi}, which is set to 90º for the cubemap projection [150]. After computing all the required query inputs for MSL from the panoramic image such as I_{qi}, K^{qi}, and $T_{m_q}^{c_{qi}}$, MSL can then be used to localize the panoramic image relative to m_r and finally estimate $T_{m_r}^{m_q}$.

5.4 SIM Components and System

This section describes various components that constitute the SIM system and how these individual components work together to enable collaboration between multiple MR and VR users in real-time, which can perform common inspection tasks such as annotation and co-visualization. Other components of SIM such as interactive segmentation and data collection robots are incorporated here and are modified to work for VR users as well as MR users. Fig. 5.4 shows the workflow of the SIM system. The components in SIM include MR/VR devices, panoramic camera, data collection robots, and a computational server that performs localization, interactive segmentation, and synchronization between multiple devices. All the tasks in this workflow are automated, except for annotating defects by MR/VR users, which require input from an inspector, such as XRIV [46]. All computational tasks regarding spatial alignments, interactive segmentation, and synchronization are performed on a single, centralized server.

Pre-requisites: A pre-requisite for the SIM is to create a colorized 3D point cloud map of the site using a 3D scanning system equipped with an RGB camera and LiDAR. This can be done either during or prior to the inspection, and the scanning procedure could be performed by either a ground robot, drone, or a handheld scanner. The camera must first be intrinsically calibrated using a checkerboard procedure to obtain the camera matrix and distortion parameters [151]. The on-board cameras of the MR headsets may also be calibrated in the same way. The camera and LiDAR must also be extrinsically calibrated to obtain fixed relative poses between the camera and LiDAR sensors [152, 153].

Robot (r)**:** The integration of the robot into SIM is similar to how it was discussed in Chapter 4. After the collecting visual and LiDAR data of the site, a SLAM algorithm is used to fuse the data to generate the pre-built map (M) and output the poses of the sensors relative to the map frame (m_r) from each time-step [154, 125, 126]. M and the data used to construct it such as reference images (I_{rj}), camera poses ($T^{c_{rj}}_{m_r}$), and camera intrinsic parameters (K^r) are then used to create a database of N spatial anchors (A_1, A_2, $\ldots$ A_N). This database is stored in the server to perform localization requests. During the scanning procedure, the robot may be controlled by the VR or MR inspectors by specifying the desired pose for a target pose (P^{m_r}) for the robot to follow in the m_r frame. SIM allows for this scanning procedure to be performed concurrently while the MR/VR inspectors perform the inspection.

MR headset (i)**:** Once the inspection starts, each MR headset (designated as i in Fig. 5.4) automatically captures a query image (I_i) through its on-board camera (c_i) with camera intrinsic matrix (K^i). I_i and K^i are then sent to the server to perform image-based localization to spatially align the MR headset with M. The MR headset also records the local pose ($T_{m_i}^{c_i}$) of c_i relative to the MR stationary frame (m_i) using the headset's on-board head-tracking SLAM system. The server then computes the spatial alignment transformation ($T_{m_i}^{m_r}$) of m_i relative to m_r. The MR users can add annotations of defect regions, such as text, bounding boxes, and/or area measurements, which are designated as X^{m_i} and placed in the m_i frame. Annotations received from other MR/VR users, designated as Y^{m_i}, are visible to the MR users and are also placed in the m_i frame.

Panoramic camera (p)**:** The panoramic camera enables the VR users to see a real-time all-around view of the infrastructure site through panoramic images. The camera will be carried by the on-site MR user or attached to a robot or drone. Once the camera captures an image (I_p), it is then sent to the localization server to estimate the pose ($T_{m_r}^{m_p}$) of the panoramic camera frame (m_p) relative to m_r. Then, I_p will be anchored in M.

VR headset (j)**:** The VR headset (designated as j in Fig. 5.4) first receives the pre-built map (M) and positions it into Mas the start of the inspection session to visualize the site as it was scanned. The VR headset also receives I_p and $T_{m_r}^{m_p}$ from the server after completing the localization of I_p to m_r. The VR users then perform the inspection using I_p and make annotations. These annotations are anchored to a 3D location (Y^{m_r}) in m_r through ray-casting. The ray-casting algorithm searches for the closest 3D point in M that corresponds to any 2D point on I_p that is selected by the VR user [13]. The VR user can also use interactive segmentation for area measurements, where the input image is I_p. Annotations that are added by other users (X^{m_r}) are also placed in the m_r frame and visualized to the VR user.

Localization module: The localization module is to spatially align all MR/VR headsets and panoramic images into m_r using image-based localization. The server first accesses the database of spatial anchors for the site, which were created in previous sessions. Any device can send a localization request to the server, comprising a query image (I_q) captured by the query device's camera (q). The localization module then determines if I_q was captured from a MR headset (a single perspective image) or a panoramic image. If I_q is captured by the MR headset, then supplementary calibration data including the camera matrix (K^q) and local pose ($T_{m_q}^{c_q}$) should be provided. For such a case, the SSL algorithm can be used for localizing the image relative to m_r [47]. Otherwise, the MSL algorithm is executed to localize the panoramic images into m_r. Once $T_{m_r}^{m_q}$ is estimated, it is then stored alongside with I_q for the duration of the session.

Synchronization module: The synchronization module in the server is to synchronize the spatial information (e.g., defect annotations, user head/hand poses, etc.) between multiple MR/VR users in the same session. Any spatial data such as 3D locations of annotations that are generated by each device will be anchored in M and synchronously visualized to other devices. The synchronization module first receives the spatial data such as annotations and head/hand poses and converts them to the m_r frame. These data are then converted to each device's local frame before sending them to that device. For example, when the MR user places an annotation (X^{m_i}), it is created in the m_i frame, the local frame of the MR headset. The synchronization module then retrieves $T^{m_r}_{m_i}$ from the localization module and transforms X^{m_i} to X^{m_r} so that the annotations are aligned in the m_r frame and can be shared with the VR users. Similarly, VR users' annotation in the m_r frame (Y^{m_r}) can be shared with the MR user. As a result, each user can show the data, which is shared by other MR/VR users. Once the inspection is over, all annotations produced by the MR/VR inspectors which were already converted to the m_r frame are then stored in the database for future sessions.

Interactive segmentation module: The interactive segmentation module, discussed in Chapter 3 is also integrated in SIM here so that human inspectors can perform defect area measurements with the aid of AI. f-BRS is deployed to the server to receive captured images and seed points selected by the human users through MR/VR. The image is first segmented by f-BRS and then the segmentation mask is refined based on the judgement of the human user who can modify the seed point selection to improve the quality of the results. The resulting boundary mask is then ray-casted into M to get the 3D locations of the boundary points, which are then used to calculate the segmented area of the defect. This area measurement is then displayed to all users as an annotation in the m_r frame.

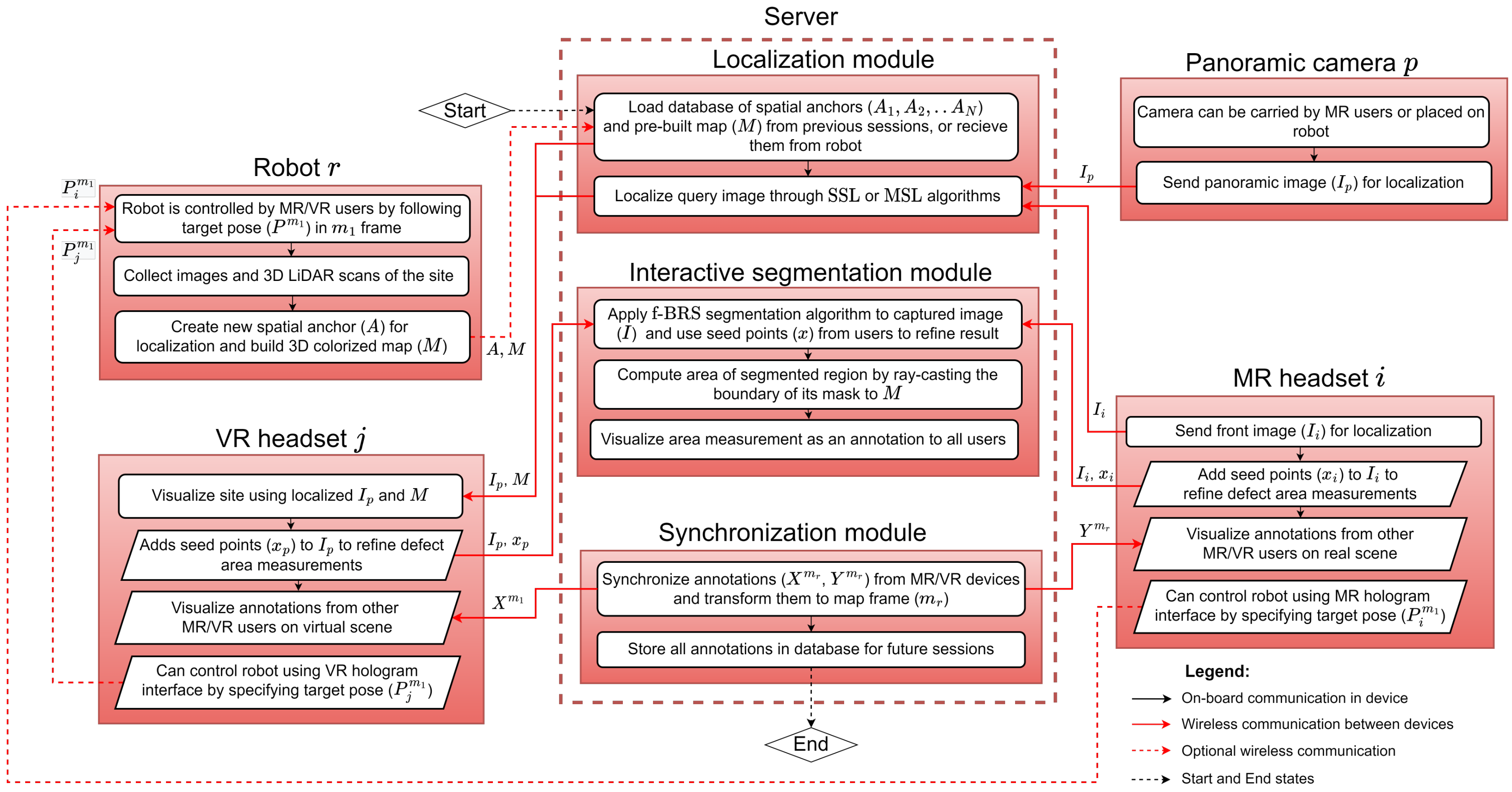

Server
Localization module
Load database of spatial anchors ($A_1, A_2, .. A_N$) and pre-built map (M) from previous sessions, or recieve them from robot
Localize query image through SSL or MSL algorithms
Interactive segmentation module
Apply f-BRS segmentation algorithm to captured image (I) and use seed points (x) from users to refine result
Compute area of segmented region by ray-casting the boundary of its mask to M
Visualize area measurement as an annotation to all users
Synchronization module
Synchronize annotations (X^{m_r}, Y^{m_r}) from MR/VR devices and transform them to map frame (m_r)
Store all annotations in database for future sessions
Start
End
Panoramic camera p
Camera can be carried by MR users or placed on robot
Send panoramic image (I_p) for localization
I_p
Robot r
Robot is controlled by MR/VR users by following target pose (P^{m_1}) in m_1 frame
Collect images and 3D LiDAR scans of the site
Create new spatial anchor (A) for localization and build 3D colorized map (M)
A, M
$P_i^{m_1}$
$P_j^{m_1}$
VR headset j
Visualize site using localized I_p and M
Adds seed points (x_p) to I_p to refine defect area measurements
Visualize annotations from other MR/VR users on virtual scene
Can control robot using VR hologram interface by specifying target pose ($P_j^{m_1}$)
I_p, M
I_p, x_p
X^{m_1}
MR headset i
Send front image (I_i) for localization
Add seed points (x_i) to I_i to refine defect area measurements
Visualize annotations from other MR/VR users on real scene
Can control robot using MR hologram interface by specifying target pose ($P_i^{m_1}$)
I_i
I_i, x_i
Y^{m_r}
Legend:
On-board communication in device
Wireless communication between devices
Optional wireless communication
Start and End states

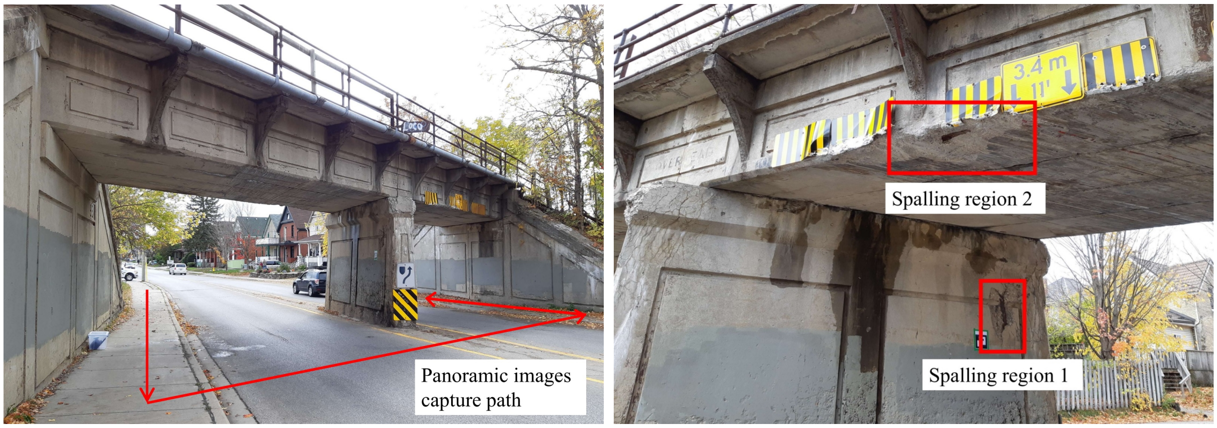
Figure 5.4: Process pipeline of the proposed SIM: Major components in the system include MR/VR devices, a computational server, data collection robot, and a panoramic camera.

]

5.5 Experimental Validation

5.5.1 Details of the inspection site

The feasibility of SIM has been tested and evaluated from an in-service reinforced concrete railway bridge located in Waterloo, Ontario, shown in Fig. 5.5. The main span of the bridge is 20 m, and 3.4 m high, with two large spalling regions (approximately 0.5m width each) located at the deck and pier of the bridge, respectively, shown in Fig. 5.5b.



(a)

(b)

Figure 5.5: Test bridge structure: (a) a path for collecting panoramic images and (b) two spalling regions at the side and top of the bridge abutment.

The experiment was designed to demonstrate real-time collaborative annotation and sharing capability in SIM. During the experiments, two MR users inspected the bridge on-site while one VR user inspected the bridge remotely. One MR user carried a panoramic camera to capture all-around views of the site and share them with the remote VR user, while the second MR user assisted with labeling the spalling annotations. The MR user carrying the panoramic camera traversed the path shown in Fig. 5.5a. The VR user annotated spalling regions 1 and 2 on the panoramic images. The annotations were then visualized in real-time by both the MR users on-site, which are anchored to the real locations of the defects. The accuracy of MSL for the panoramic image localization was then tested to demonstrate that the annotations placed by the VR user were accurately anchored to their correct physical defect locations on the structure.

5.5.2 Mapping of the bridge

SIM requires a prebuilt 3D map of the test bridge site. I utilized a custom handheld scanner built by myself, equipped with an Intel Realsense D455 RGB camera and a Livox Avia LiDAR in Fig. 5.6. The Intel Realsense D455 camera captures color images with a resolution of 1,280 × 720 and a field of view of 90º [155]. A Livox Avia is a solid-state LiDAR and collects 3D point clouds, at the rate of 240,000 points per second and a maximum range of 200 m with an average error within 2 cm [156]. The Livox Avia also contains an integrated Inertial Measurement Unit (IMU) to improve tracking performance through pre-integration between lidar frames. The camera and LiDAR sensors were combined using a 3D printed component, allowing me to use fixed external calibration parameters. For outdoor applications, a Neutral Density (ND) filter was attached to the front of the camera lens to reduce infrared interference in the RGB sensors from sunlight. An Intel NUC mobile computer was used for data collection from the sensor. The camera and LiDAR sensors were calibrated to obtain the intrinsic camera matrix and the extrinsic camera-LiDAR transformation.

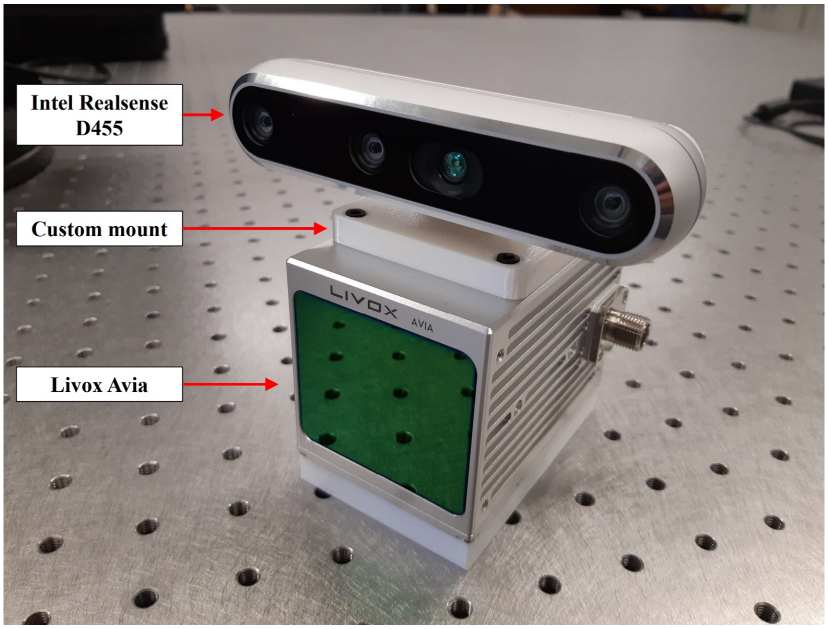
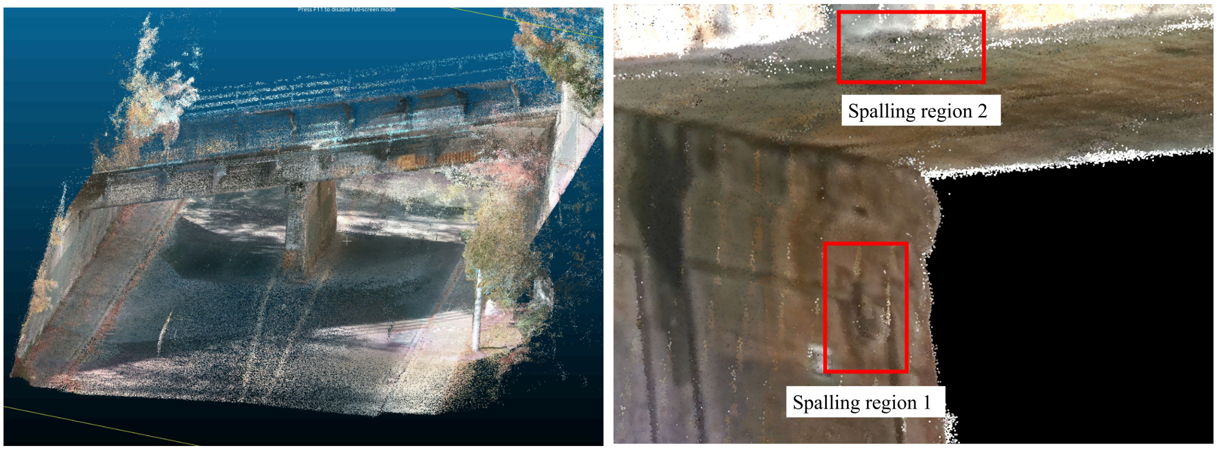


Figure 5.6: Custom-built handheld scanner equipped with Intel Realsense D455 camera and Livox Avia LiDAR for creating pre-built 3D maps for SIM.

In this study, a SLAM algorithm called R3Live was used to process the LiDAR scans, images, and IMU measurements to produce the sensor trajectory and the 3D map of



(a) (b)

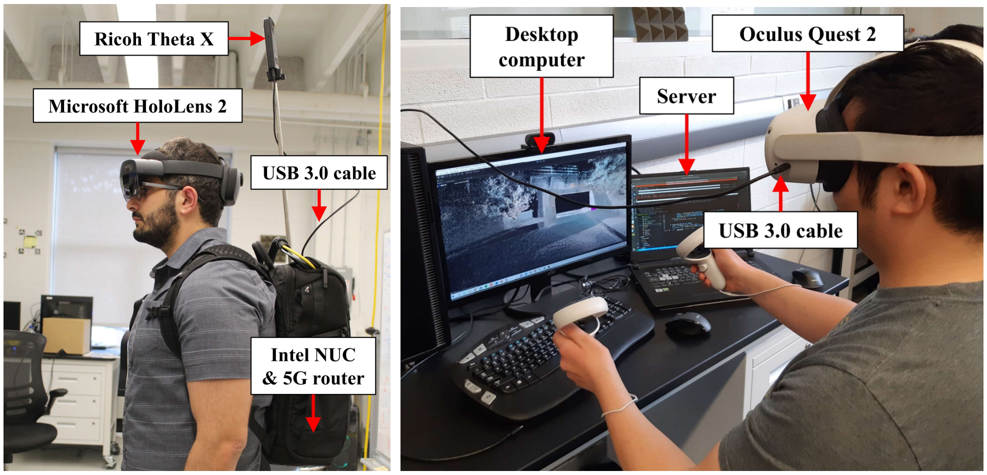
Figure 5.7: Colorized pre-built map of bridge site generated from data captured using custom-built LiDAR scanner: (a) full map of bridge site and (b) close-up view of spalling regions.

the bridge [126]. The 3D map reconstructed in Fig. 5.7a, was then used as the virtual environment for the VR user in SIM to explore various regions of the site and measure the sizes of defect regions. In terms of overall accuracy, the R3Live algorithm produces point cloud maps that are accurate up to 5 cm in terms of drift on average, as claimed in their study [126]. However, still, fine details such as the spalling regions in Fig. 5.7b are not visible because of the noise in the LiDAR measurements and insufficient resolutions of the point cloud reconstructed. Also, there is a noise range of 1-2 cm in the map, which leads to obscuring details of the map such as spalling, making them difficult to discern. Thus, the 3D map had to be down-sampled using a voxel size of 1cm to mitigate LiDAR noise, despite the cost of reducing spatial resolution. Although point cloud maps serve as useful tools to navigate the structure and identify inspection regions, they are not suitable for conducting detailed examinations. Hence, the panoramic images were captured and provided on the map to inspect the bridge structure in VR.

5.5.3 Implementation of SIM

Hardware configuration

The two on-site users were equipped with HL2 as the MR headsets. HL2 has on-board front cameras with a resolution of 3,904 × 2,196 pixels during the photo capture mode [129]. The intrinsic camera calibration parameters of the HL2 were provided by the manufacturer. One of the MR users was also equipped with a Ricoh Theta X camera attached to a backpack for collecting panoramic images, as shown in Fig. 5.8a. The Ricoh Theta X automatically produced stitched panoramic images in the equirectangular projection with a resolution of 11,008 × 5,504 pixels [157]. The images were then sent through a USB 3.0 cable to an Intel NUC computer placed inside the backpack. The camera is triggered every second to capture the panoramic image. The HL2 and Intel NUC were wirelessly connected to a 5G router in the backpack, which provided connectivity to both MR users on-site through a Wi-Fi hotspot.



(a)

(b)

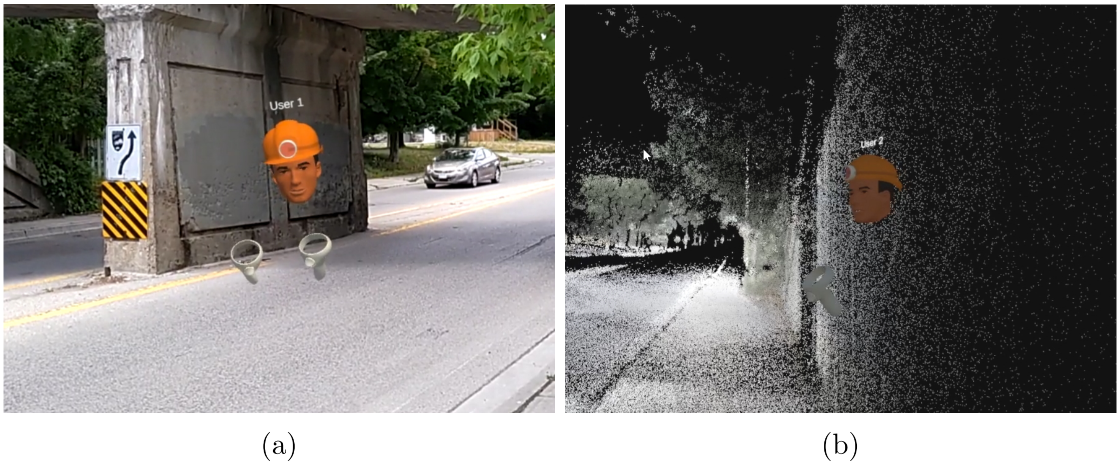
Figure 5.8: Hardware implementation for SIM: (a) on-site MR user equipped with panoramic camera and (b) remote VR user.

For the remote VR user, an Oculus Quest 2 was used as the VR headset for my implementation [158] in Fig. 5.8b. The Oculus Quest 2 was connected to a desktop computer equipped with an AMD Ryzen 5 3600 CPU and a NVIDIA GeForce GTX 1660 Ti GPU to run the VR application, which was necessary for rendering large point clouds such as the pre-built 3D map in SIM. The desktop computer then sends the rendered frames (scenes) to the Oculus Quest 2 through a USB 3.0 cable with minimal latency.

The server, which performed the localization and synchronization, was configured on a personal laptop. This laptop was an ROG Strix G17, equipped with an NVIDIA GeForce RTX 2060 GPU and an Intel Core i7 processor. The laptop was placed in the office next to the VR user, as depicted in Fig 8b. The server was wirelessly connected to the VR user's desktop through Wi-Fi, and the on-site MR users were remotely connected to the server through a 5G internet connection. The server used TCP sockets to establish persistent two-way connections to all MR/VR devices and the panoramic camera. The TCP sockets allow these devices to send and receive data (e.g., images, annotations, user poses, etc.) to and from the server.

User interaction

SIM facilitates the telepresence of MR/VR users, allowing them to visualize and engage with each other remotely. This interaction is achieved through the use of 3D avatars, enhancing spatially aware collaboration. The Oculus Quest 2 and HL2 support the function of tracking the users' head and hand movements. For the Oculus Quest 2, a controller was used to track hand positions and angles, while the HL2 can track the user's hands through its built-in cameras when they are visible to the camera. The head and hand positions of the MR/VR users were then spatially aligned and synchronized with other users through the server. The positions of other users' heads and hands were mapped onto avatar models. For example, in Fig. 5.9a the MR user is viewing a remote VR user (User 1), while in Fig. 5.9b the VR user is viewing an on-site MR user (User 2). Note that the hands are visualized with Oculus controller icons.



(a) (b)

Figure 5.9: Spatially aware collaboration in MR/VR using 3D avatars: Each user perceives other users as being present in their environment through avatars, allowing them to monitor hand and head positions. Screenshots in (a) and (b) illustrate the perspective of an on-site MR user and a remote VR user, respectively.

Implementation of MSL for panoramic image localization

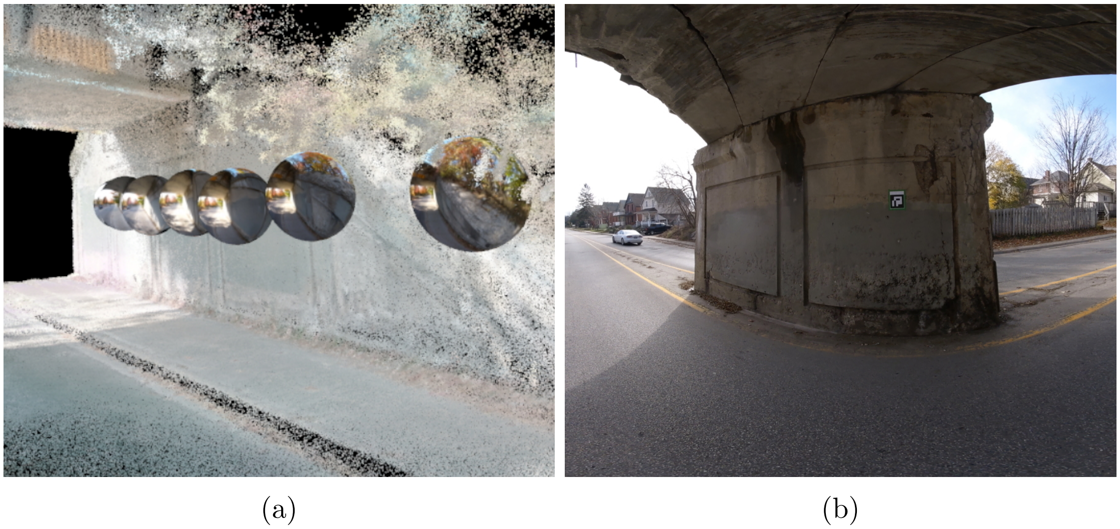
In the experiment, MSL was implemented for spatial alignment of panoramic images relative to the pre-built map. First, the handheld scanner was used to collect 2,890 images and 1,957 LiDAR scans (approximately 47 million points total) during the scanning procedure, while R3Live SLAM was used to obtain the camera poses for each image. For each RGB image, the previous 5 seconds of LiDAR scans were first concatenated and then projected onto the image plane to produce a corresponding sparse depth image. This is the same methodology for producing depth images as the KITTI dataset, which is the standard way for converting LiDAR scans to depth images [159]. The purpose of generating these depth images was to back-project the locations of 2D points in the RGB image onto 3D coordinates in the m_r frame. The RGB and depth images were then sampled uniformly to create 250 spatial anchors.

Once a panoramic image was captured, it was then converted to four perspective images using the cubemap projection using the OmniCV package [147]. MSL was used to localize the panoramic images into the prebuilt map using the procedure described in Fig. 5.3. In the MSL process, accurate image retrieval and precise feature detection and matching algorithms were deployed using the hierarchical localization toolbox, which is a library that implements all the feature detection, matching, and image retrieval algorithms used in my study [67].

For image retrieval, the learning-based place recognition algorithm, NetVLAD, was used to retrieve the top 5 most similar spatial anchors to each query image [116]. I then evaluated the accuracy of several feature detection algorithms including ORB, SIFT, and SuperPoint [139, 138, 141]. For feature matching, I tested Nearest Neighbor search (NN) with the number of classes (k) set to k=2 for SIFT and ORB, as well as dense feature matching algorithms including SuperGlue for SuperPoint features and detector-free matching with LoFTR [143, 160]. For SuperGlue and LoFTR, pre-trained models were provided by the original authors, which were trained on images in outdoors environments. These models were then deployed in my study for feature matching. For RANSAC, several parameters were selected to optimize the PnP solver's performance. For example, for each image, the allowable re-projection error was selected to be 10 pixels, the maximum number of iterations was selected to be 10,000, and the minimum number of inlier points for accepting localization results as valid was set to at least 50 points.

Panoramic image rendering

The set of panoramic images was overlaid as spheres on the 3D map in the virtual environment for remote VR users, in Fig. 5.10a. The location of each panoramic sphere is computed from the pose of the panoramic image obtained from the MSL. The VR user can navigate the map and select the sphere to see the surrounding view of the current state of the site on the selected location. Fig. 5.10b shows a sample panoramic image. The VR user is then able to add annotations to the panoramic image, which are then projected and anchored to the underlying 3D map. The annotations added by the VR user are then anchored to their corresponding locations in the real or virtual scenes and shared with other users in the session in real-time. In my implementation, the 3D pre-built map and the panoramic images were rendered using the Unity game engine [161].



(a)

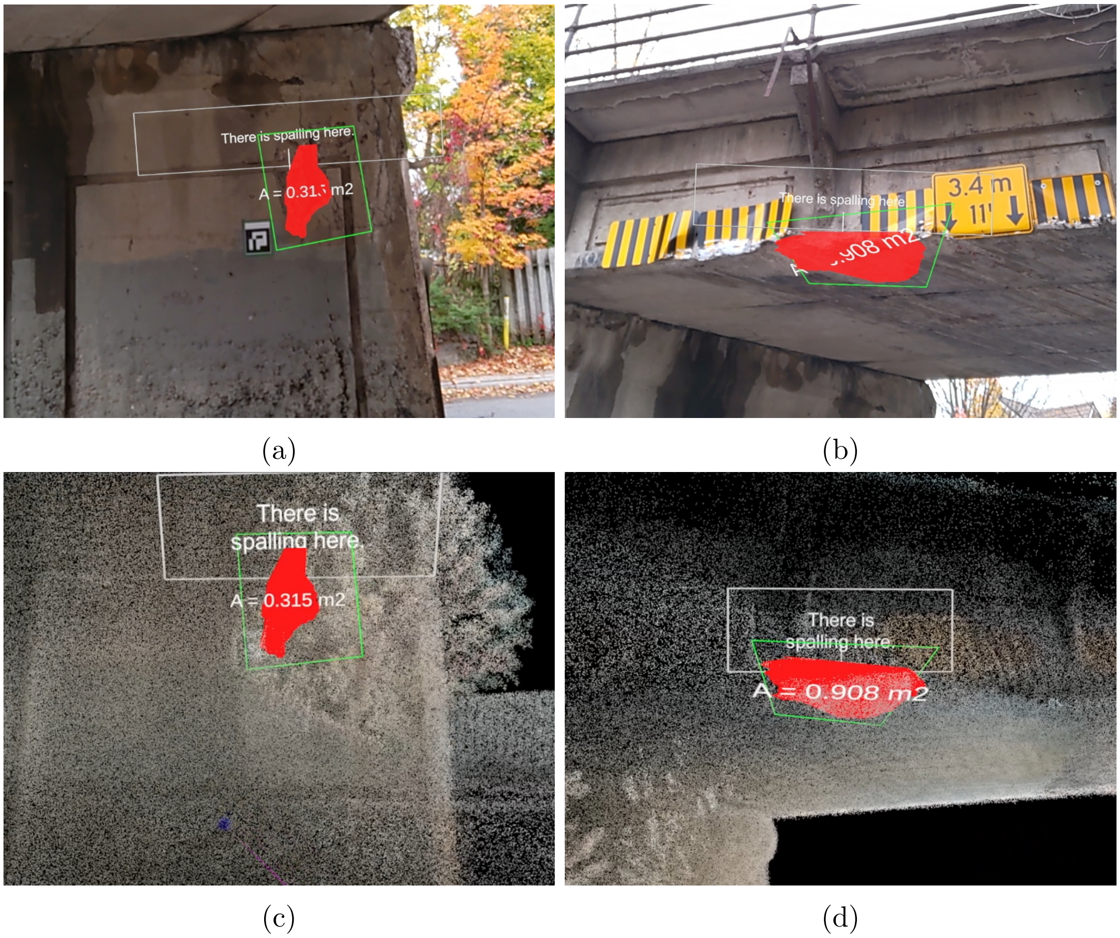
(b)

Figure 5.10: Panoramic image rendering in the VR environment: Panoramic images captured by on-site inspectors are spatially aligned with the pre-built map using MSL, allowing users to share annotations for inspection. (a) shows a 3D map in a virtual environment with localized panoramic images and if VR users click one of the panoramic images, the surrounding view on the selected location can be seen like the one in (b).

Spalling damage inspection

In this experiment, the goal is to inspect spalling damage on the bridge. During the inspection session, the VR user annotated both spalling regions 1 and 2 on the localized panoramic images and the annotated information includes the bounding box, defect type, and damage area measurement. The annotations were then anchored to the prebuilt maps and were shared with the virtual and the real scenes so that they were visible to the two MR users and one VR user. From the MR users' perspective, the annotations are projected as holograms that are anchored to the real scene, as seen in Fig. 5.11a. For the VR user, the annotations are shown to be anchored to the same location on the 3D map, as seen in Fig. 5.11c. The first and second images in Fig. 5.11 indicate the annotation of spalling regions 1 and 2, respectively.

Note that in this experiment, the accuracy of the area measurement for spalling regions was not measured because the performance of damage segmentation and quantification using the prebuilt maps have been evaluated in chapter 3. This experiment aims to demonstrate how interactive segmentation of damage regions developed by the myself or



(a)

(b)

(c)

(d)

Figure 5.11: Spalling region annotations including damage type, bounding box, segmented region, and area measurement: The annotations are spatially aligned to the 3D map and simultaneously shown to (a) on-site MR users on real-scenes and (b) remote VR users on virtual scenes.

others can be integrated into the SIM framework for collaboratively performing the inspection, and how spatial alignment enables annotations to be co-visible to MR/VR users simultaneously in a spatially aware context. However, in the next section, I will test the performance of MSL for panoramic image localization, which will affect annotation localization.

5.5.4 Spatial alignment evaluation

Most existing panoramic image localization algorithms use either the equirectangular or omnidirectional image projection models [162, 108, 163, 164, 165, 166]. These techniques perform feature detection on the entire image at once so that visual features can be extracted from images. However, this approach does not consider the high degree of distortion in panoramic images outside of the image center, which may negatively affect the quality of the features and cause failing or inadequate feature matching [167]. Some feature detectors such as SIFT have been modified to work for highly distorted images [168]. However, hand-crafted features such as SIFT and ORB have been shown to underperform when compared to state-of-the-art learning-based feature detectors and matchers such as SuperPoint, SuperGlue, R2D2, and LoFTR, in terms of both the quality and quantity of features extraction and matching [141, 143, 160]. This issue becomes apparent in civil engineering structures which often have low features textures (e.g., concrete, steel), or textures with repeating patterns (e.g., bricks), which makes detecting and matching features very difficult. Hand-crafted features have been shown to underperform for localization in civil engineering structures compared to learning-based features that are more robust in these environments [47].

However, learning-based features have still been only tested on perspective images with a low degree of distortion. This can be attributed to two main factors: the limited availability of adequate training data for different image projection models, such as equirectangular or omnidirectional images. Next, many DNNs rely on minimal distortion in local regions to effectively match corresponding features across multiple images. Thus, current approaches which only utilize hand-crafted features cannot be used to reliably localize panoramic images in civil engineering structures.

In this context, I evaluate the performance of the hand craft and learning-based feature extraction and matching methods in achieving spatial alignment. I gauge the spatial alignment accuracy of MSL for anchoring panoramic images by quantifying the spatial deviation between objects in the image and the 3D map. If the localization is accurate, then the offset would be zero and the panoramic images could be perfectly overlayed over the 3D map. To do this evaluation, a fiducial marker attached to the structure was used to measure the offset because it is visually distinctive, has visible corners, and has known dimensions. The marker has square sides of 20 cm and was taped to the bridge abutment near spalling region 1, shown in Fig. 5.12.

This method of measuring error is different than measuring the localization error of the estimated pose directly. However, both spatial alignment and localization error measure the absolute deviation between the images and 3D map. Unfortunately, it is not possible

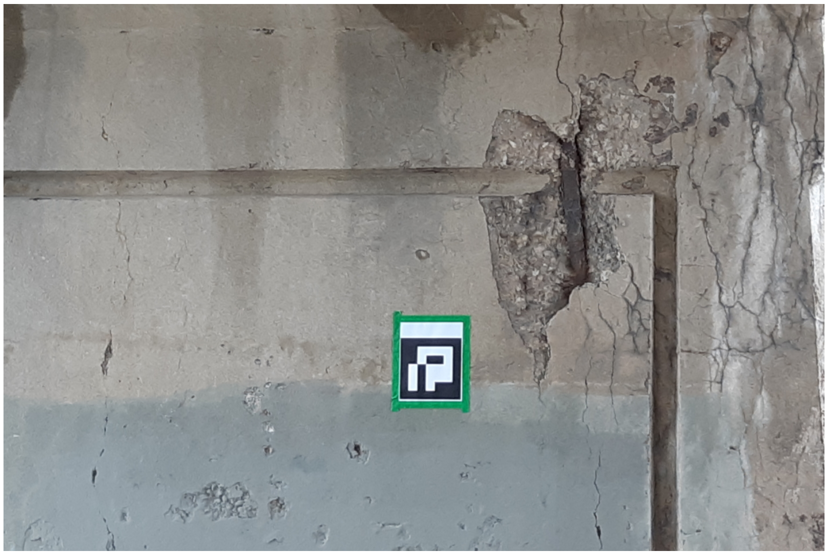


Figure 5.12: Fiducial marker taped to bridge's surface near spalling region 1 to quantify spatial alignment error in MSL for panoramic image localization.

to directly measure the precise locations of the 3D coordinates of the panoramic images. I assume that manual measurement from the 3D map is considered as the ground-truth, although, strictly speaking, this might not be entirely accurate. Without using beacon tracking system, the accurate location cannot be measured.

After attaching the fiducial marker, the structure was scanned to create the pre-built 3D map so that the four corners of the marker were visible in the 3D map. The corners were then manually selected using CloudCompare to obtain their 3D coordinates in m_r [169]. Once a panoramic image is captured and localized using MSL, the 3D corners are then projected onto the panoramic image to see where the marker would be in the image based on the localization result. Ideally, the projected corner points obtained from the 3D map would match the corners of the marker seen in the image.

The projected corners of the marker are then compared with the results of an ArUco marker detection algorithm implemented in OpenCV to automatically detect the corners of the marker in the panoramic images [128]. The pixel coordinates of the marker were then converted to physical coordinates using a four-point homography algorithm which also fixed perspective distortion [99]. The homography matrix was estimated by transforming the pixel coordinates of the corners to physical coordinates parallel to the marker's sides (20 cm × 20 cm). The spatial alignment error was then defined as the L2 distance in centimeters between the centers of the projected marker location and the marker location obtained from the detection algorithm.

For this experiment, 48 panoramic images were collected along the data collection path, and the marker was visible in each image. Two classic feature detectors (ORB, SIFT) and two learning-based feature detectors and matchers (LoFTR, SuperPoint+SuperGlue) were evaluated. After running the MSL panoramic image localization algorithm for each of the test images multiple times using different feature detectors and matchers, the results for panoramic image localization were tabulated in Table 5.1.

Table 5.1: Comparison of the spatial alignment errors using different feature detectors and matchers implemented in MSL for panoramic image localization.

Feature detector + matcher	Mean spatial alignment error ± standard deviation	Mean number of inlier matches / total matches
LoFTR	5.1 ± 2.9 cm	7416 / 18889
SuperPoint + SuperGlue	8.1 ± 7.6 cm	1230 / 4640
SIFT + NN	12.0 ± 12.3 cm	335 / 920
ORB + NN	920 ± 2,300 cm	25 / 358

The first column in Table 1 is presented as the mean error in centimeters, plus the standard deviation value for the images that were evaluated. The second column shows the mean number of inlier feature matches divided by the total number of feature matches per image. When the number of inlier-matched feature points is low (50 in this experiment), then I consider the result to be unreliable due to the low inlier ratio, so I excluded those cases from the first column.

Among the feature detectors and matchers that were evaluated, LoFTR produced the smallest mean error and standard deviation results. In comparison, the ORB did not produce enough inlier feature matches for a reliable result, so the mean error and standard deviation are very high. The difference in accuracy can be explained as follows: LoFTR produced the most inlier and total feature matches on average compared to SuperPoint+SuperGlue, SIFT, and ORB per panoramic image. Thus, MSL had a higher chance of producing a geometrically correct pose estimate when using LoFTR due to the large number of total and inlier feature matches that it produced. Another metric to consider is the ratio of inlier matches to total matches, which is indicative of the reliability of each feature detector and matcher. LoFTR (39.26%) has the highest inlier ratio, compared to SuperPoint+SuperGlue (26.52%), SIFT (36.39%), and ORB (7.06%), which makes it the most reliable feature matcher. SIFT also has a relatively high inlier ratio which indicates that it can detect and match features reliably. However, the low number of total features reduces the overall accuracy of MSL. This experiment also shows that learning-based feature detectors and matchers such as LoFTR and SuperPoint+SuperGlue significantly outperform hand-crafted methods such as SIFT and ORB in civil engineering structures with low brightness (underneath the bridge deck) and low-feature surfaces (concrete surfaces). Overall, LoFTR produced the best overall result for this study because it produced a large number of inlier feature matches, and it produced a spatial alignment accuracy of around 5 cm which is sufficient for bridge inspection applications.

It is important to highlight that some sources of error may have influenced the accuracy of the spatial alignment result. These sources of error are related to how the spatial anchors were created, so they do not affect the comparison between different feature detectors and matchers. First, there was some drift error from SLAM when constructing the pre-built 3D map. This drift error must have accumulated when creating the spatial anchors because they rely on the poses of the reference camera obtained from SLAM. Second, random noise error from the LiDAR sensor (1-2 cm) also influenced the creation of the spatial anchors because they rely on 3D data from the LiDAR. Thus, the combination of these two sources of errors when creating the spatial anchors can also affect the spatial alignment result, which is not related to either MSL or the feature detectors and matchers. Future studies should prioritize the reduction of the error sources associated with constructing the anchors. This could be achieved through the adoption of higher precision 3D sensors or the implementation of more robust SLAM algorithms.

Chapter 6

Summary and Conclusion

The research presented in this thesis addresses the critical need for enabling real-time remote collaborative structural inspections that can integrate human inspectors, advanced data collection platforms, and AI. The key motivation behind this work is to overcome the technical challenges that have hindered real-time collaboration between human users and machine agents so that human inspectors can make more informed decisions by leveraging precise inspection data and facilitating their real-time discussions. The SIM system proposed in this study facilitates seamless collaboration between human users and machine agents. New technological developments in the field of immersive environments enabled through MR/VR are adopted to support this type of collaboration and advance the field of vision-based inspections. MR/VR are utilized for aiding on-site inspectors by improving documentation, allowing communication with remote domain experts to improve the quality of data, and reducing the barrier of entry to interacting with robots and AI.

This concluding section summarizes the key findings and contributions of this work:

1. **Development of the SIM System:** The SIM system was created as a novel approach to infrastructure inspection. It integrates multiple components, including MR/VR-equipped inspectors, data collection robots, and AI algorithms, into a unified framework. SIM ensures that inspections are performed as a single synchronous process that combines data collection, analysis, and decision-making. Thus, ensuring that human users and machine agents collaborate to collect high quality data, process it, and make decisions on-site to support robust and efficient inspection.

2. **Algorithmic Contributions:** This thesis also introduces methodological contributions based on the algorithms that were developed as part of creating SIM. First, a

novel procedure was developed for refining real-time quantitative damage measurements from interactive segmentation. This is done by taking the 2D boundary of the binary mask produced by the interactive segmentation algorithm and spatially matching 2D image pixels to the 3D scene. Second, the SSL image-based localization algorithm was introduced to create spatial anchors for spatial alignment between robots and MR headsets, facilitating real-time supervision and control of robots by humans. Third, the MSL algorithm was developed to localize panoramic images and align them with a pre-built 3D map of the site, enabling seamless collaboration between MR and VR users in an immersive virtual environment.

3. **Human-AI Collaboration:** A novel damage quantification method called XRIV was developed to enhance the accuracy and robustness of automated vision-based visual inspection algorithms by incorporating human interactivity through MR. This approach improves the segmentation results and underscores the advantages of real-time interaction between expert users equipped with MR devices and AI. Experimental results showed that XRIV produces accurate spalling damage segmentation results (<10% error) that are suitable for vision-based inspection applications.

4. **Human-Robot Collaboration:** A system called HMCI was developed to enable on-site MR-equipped inspectors to collaborate with data collection robots, ensuring that defects are not missed during data collection. SSL was quantitatively tested in a lab environment using multiple feature detection and matching algorithms to achieve the best spatial alignment results. SuperPoint + SuperGlue were used to feature extraction and matching, which produced the lowest error ($\leq$1cm error). The experiment demonstrated how the MR-equipped inspector and robot are able to collaborate together to inspect the test environment and locate all sturctural defects, and that SSL achieves sufficient accuracy for real-time human-robot collaboration.

5. **Distributed Collaboration:** A distributed collaborative system was developed to enable multiple on-site MR-equipped inspectors to collaborate with remote VR-equipped experts. This approach enhances spatial awareness, co-presence, and synchronous decision-making between on-site and remote users. An experiment was conducted to demonstrate how multiple MR/VR inspectors collaborate remotely to perform a bridge inspection. The experiment also showed that MSL produced sufficient spatial alignment accuracy ($\leq$5cm error) for vision-based inspections when using the LoFTR feature matching algorithm.

These research outcomes can be integrated into the SIM system to transform traditional infrastructure inspection workflow that suffer from asynchronicity and lack of real-time collaboration, which can negatively affect data quality, accuracy, and may lead to costly re-inspections. The SIM system offers a solution that reduces spatial-temporal gaps in the inspection process, leading to more reliable and efficient inspections. Through the utilization of SIM, it is anticipated that engineers will be empowered to perform rapid and reliable real-time inspections across diverse scenarios and having no restriction on their locations.

It is worth noting that the SIM system has some limitations related to the proposed framework in this thesis. First, SIM requires a consistent internet connection to ensure that on-site and remote inspectors can always communicate with each other, the robots, and with the computational server to exchange data. Therefore, it is not possible to use SIM in remote regions with limited internet connectivity. Second, the entire structure must be scanned before the inspection session starts to ensure that MSL can localize panoramic images successfully and the digitized the models are shared with the remote VR users to be involved in the inspection session. Complete coverage of the structure's surface with enough overlap is recommended. However, SIM can still be utilized without a pre-built map if the configuration only involves the MR users, robots, and AI, and if the MR users are within close proximity to the robots. This is because SSL does not require a pre-built map to align multiple devices together locally. Third, the specific configuration of SIM (number of MR users, VR users, robots, and AI) would depend on many factors (e.g., domain, inspection application, structure size/complexity, etc.) that are not accounted for in this thesis. Although SIM is theoretically able to allow any number of MR/VR inspectors and robots to collaborate simultaneously, this thesis only tested some limited configurations designed to test the accuracy of specific components of the system (e.g., interactive segmentation, SSL, MSL). More research is needed to determine which configurations of SIM are practical for real inspection applications.

While this research has made significant strides in advancing infrastructure inspection, several avenues for future research remain open. These include further refinements and optimizations of the SIM system through the integration of emerging technologies for real-time data analysis and decision-making, and exploring additional interaction modes provided by MR/VR headsets. First, emerging technologies such as 5G connectivity and edge computing promise to reduce the latency of inference of AI models for data processing tasks. This is supported by increasing the bandwidth to send larger quantities of data for computing tasks and should allow utilizing AI for tasks such as defect detection with lower latency. Second, newer MR/VR headsets support novel modes of interaction such as eye-gaze and voice commands for interacting with robots and AI. Research should be conducted to determine how to utilize these modes of interaction to facilitate interaction and collaboration within the SIM system.

Emerging technological developments in generative AI such as Large Language Models (LLMs) allow for the potential of robot-AI collaboration, which is outside of the scope of this thesis. Potential uses could include robot path planning through generative AI and the analysis of text-based inspection reports. Aforementioned avenues of research such as 5G edge computing and MR/VR-based voice commands could support these efforts by allowing faster inference times and the issuing of speech-based commands by humans. This type of collaboration would imbue data collection robots with some autonomy to collect data with very limited supervision from human inspectors, allowing human inspectors to focus more on decision-making tasks.

In conclusion, this thesis has contributed to advancing the field of civil infrastructure inspection through the development of the SIM system to transform traditional inspection workflows. The contributions made in the areas of human-AI collaboration, human-robot collaboration, and distributed collaboration have the potential to transform inspection methodologies and enhance safety and quality in critical infrastructure management. This research not only showcases the power of collaboration between humans and machines but also paves the way for future innovations in the domain of inspection by embracing technological innovations such as the industrial metaverse. The SIM system represents a significant step forward in addressing the challenges posed by aging infrastructure and the need for more advanced inspection methodologies in an ever-evolving technological landscape.

References

[1] Chris Mcnally, Bill Ferreira, and David L. A. Gordon. The canadian infrastructure report card. *Canada The State of the Federation 2015*, pages 27–44, 2015.

[2] ASCE. America's infrastructure report card 2021, 2021.

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

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

[5] Office of the Auditor General of Ontario. *Value for Money Audit: Inspection and Maintenance of the Province's Bridges and Culverts.* Office of the Auditor General of Ontario, 2021.

[6] Mark Moore, Brent M Phares, Benjamin Graybeal, Dennis Rolander, Glenn Washer, Janney Wiss, et al. Reliability of visual inspection for highway bridges, volume i. Technical report, Turner-Fairbank Highway Research Center, 2001.

[7] Chul Min Yeum and Shirley J Dyke. Vision-based automated crack detection for bridge inspection. *Computer-Aided Civil and Infrastructure Engineering*, 30(10):759–770, 2015.

[8] Billie F Spencer Jr, Vedhus Hoskere, and Yasutaka Narazaki. Advances in computer vision-based civil infrastructure inspection and monitoring. *Engineering*, 5(2):199–222, 2019.

[9] X.W. Ye, T. Jin, and C.B. Yun. A review on deep learning-based structural health monitoring of civil infrastructures. *Smart Structures and Systems*, 24(5):567–585, Nov 2019.

[10] Mohsen Azimi, Armin Dadras Eslamlou, and Gokhan Pekcan. Data-driven structural health monitoring and damage detection through deep leaning: State-of-the-art review. *sensors*, 20, 2020.

[11] Chuan-Zhi Dong and F Necati Catbas. A review of computer vision–based structural health monitoring at local and global levels. *Structural Health Monitoring*, 20(2):692–743, Mar 2021.

[12] H Peel, S Luo, AG Cohn, and R Fuentes. Localisation of a mobile robot for bridge bearing inspection. *Automation in Construction*, 94:244–256, 2018.

[13] Nicholas Charron, Evan McLaughlin, Stephen Phillips, Kevin Goorts, Sriram Narasimhan, and Steven L Waslander. Automated bridge inspection using mobile ground robotics. *Journal of Structural Engineering*, 145(11):04019137, Nov 2019.

[14] Siyuan Chen, Debra F Laefer, Eleni Mangina, SM Iman Zolanvari, and Jonathan Byrne. Uav bridge inspection through evaluated 3d reconstructions. *Journal of Bridge Engineering*, 24(4):05019001, 2019.

[15] Vedhus Hoskere, Jong-Woong Park, Hyungchul Yoon, and Billie F Spencer Jr. Vision-based modal survey of civil infrastructure using unmanned aerial vehicles. *Journal of Structural Engineering*, 145(7):04019062, Jul 2019.

[16] Evan McLaughlin, Nicholas Charron, and Sriram Narasimhan. Automated defect quantification in concrete bridges using robotics and deep learning. *Journal of Computing in Civil Engineering*, 34(5):04020029, 2020.

[17] Yi-zhou Lin, Zhen-hua Nie, and Hong-wei Ma. Structural damage detection with automatic feature-extraction through deep learning. *Computer-Aided Civil and Infrastructure Engineering*, 32(12):1025–1046, 2017.

[18] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types: Autonomous shm using deep faster r-cnn. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, Sep 2018.

[19] Shengyuan Li, Xuefeng Zhao, and Guangyi Zhou. Automatic pixel-level multiple damage detection of concrete structure using fully convolutional network. *Computer-Aided Civil and Infrastructure Engineering*, 34(7):616–634, 2019.

[20] Chul Min Yeum, Jongseong Choi, and Shirley J Dyke. Automated region-of-interest localization and classification for vision-based visual assessment of civil infrastructure. *Structural Health Monitoring*, 18(3):675–689, May 2019.

[21] Byunghyun Kim and Soojin Cho. Automated multiple concrete damage detection using instance segmentation deep learning model. *Applied Sciences*, 10(22):8008, 2020.

[22] Dawei Li, Qian Xie, Xiaoxi Gong, Zhenghao Yu, Jinxuan Xu, Yangxing Sun, and Jun Wang. Automatic defect detection of metro tunnel surfaces using a vision-based inspection system. *Advanced Engineering Informatics*, 47:101206, 2021.

[23] Raza Ali, Joon Huang Chuah, Mohamad Sofian Abu Talip, Norrima Mokhtar, and Muhammad Ali Shoaib. Structural crack detection using deep convolutional neural networks. *Automation in Construction*, 133:103989, Jan 2022.

[24] Ayoung Kim and Ryan M Eustice. Real-time visual slam for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, 29(3):719–733, 2013.

[25] Young-Jin Cha, Wooram Choi, Gahyun Suh, Sadegh Mahmoudkhani, and Oral Büyüköztürk. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Computer-Aided Civil and Infrastructure Engineering*, 33(9):731–747, 2018.

[26] MJ Anitha, R Hemalatha, and S Radha. A survey on crack detection algorithms for concrete structures. In *Advances in Smart System Technologies*, pages 639–654. Springer, 2021.

[27] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, 2017.

[28] Maksim Filipenko and Ilya Afanasyev. Comparison of various slam systems for mobile robot in an indoor environment. In *2018 International Conference on Intelligent Systems (IS)*, pages 400–407. IEEE, 2018.

[29] Baichuan Huang, Jun Zhao, and Jingbin Liu. A survey of simultaneous localization and mapping. *arXiv preprint arXiv:1909.05214*, 2019.

[30] César Debeunne and Damien Vivet. A review of visual-lidar fusion based simultaneous localization and mapping. *Sensors*, 20(7):2068, 2020.

[31] Zhexiong Shang and Zhigang Shen. Real-time 3d reconstruction on construction site using visual slam and uav. *arXiv preprint arXiv:1712.07122*, 2017.

[32] Sungwook Jung, Duckyu Choi, Seungwon Song, and Hyun Myung. Bridge inspection using unmanned aerial vehicle based on hg-slam: Hierarchical graph-based slam. *Remote Sensing*, 12(18):3022, 2020.

[33] Xiaolong Chen, Jian Li, Shuowen Huang, Hao Cui, Peirong Liu, and Quan Sun. An automatic concrete crack-detection method fusing point clouds and images based on improved otsu's algorithm. *Sensors*, 21(5):1581, 2021.

[34] Sattar Dorafshan and Marc Maguire. Bridge inspection: human performance, unmanned aerial systems and automation. *Journal of Civil Structural Health Monitoring*, 8(3):443–476, Jul 2018.

[35] Tarek Rakha and Alice Gorodetsky. Review of unmanned aerial system (uas) applications in the built environment: Towards automated building inspection procedures using drones. *Automation in Construction*, 93:252–264, Sep 2018.

[36] Stephen Phillips and Sriram Narasimhan. Automating data collection for robotic bridge inspections. *Journal of Bridge Engineering*, 24(8):04019075, Aug 2019.

[37] Nadeem M. Shajahan, Thomas Kuruvila, Arjun S. Kumar, and Dhivin Davis. Automated inspection of monopole tower using drones and computer vision. In *2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS)*, page 187–192, Feb 2019.

[38] Abdellah Chehri and Ali Saeidi. Iot and deep learning solutions for an automated crack detection for the inspection of concrete bridge structures. In Alfred Zimmermann, Robert J. Howlett, Lakhmi C. Jain, and Rainer Schmidt, editors, *Human Centred Intelligent Systems*, Smart Innovation, Systems and Technologies, page 110–119. Springer, 2021.

[39] Davy Tsz Kit Ng. ”what is the metaverse? definitions, technologies and the community of inquiry”. *Australasian Journal of Educational Technology*, 38(44):190–205, Nov 2022.

[40] David V. Jáuregui and Kenneth R. White. ”implementation of virtual reality in routine bridge inspection”. *Transportation Research Record: Journal of the Transportation Research Board*, 1827(1):29–35, Jan 2003.

[41] Romina Druta, Cristian Druta, Paul Negirla, and Ioan Silea. ”a review on methods and systems for remote collaboration”. *Applied Sciences*, 11(21):10035, Oct 2021.

[42] Alexander Schäfer, Gerd Reis, and Didier Stricker. ”a survey on synchronous augmented, virtual and mixed reality remote collaboration systems”. (arXiv:2102.05998), Feb 2021. arXiv:2102.05998 [cs].

[43] Urs Riedlinger, Florian Klein, Marcos Hill, Christian Lambracht, Sonja Nieborowski, Ralph Holst, Sascha Bahlau, and Leif Oppermann. ”evaluation of mixed reality support for bridge inspectors using bim data: Digital prototype for a manual task with a long-lasting tradition”. *i-com*, 21(2):253–267, Aug 2022.

[44] Wen-Chung Wu and Van-Hoan Vu. ”application of virtual reality method in aircraft maintenance service—taking dornier 228 as an example”. *Applied Sciences*, 12(1414):7283, Jan 2022.

[45] Xifan Yao, Nanfeng Ma, Jianming Zhang, Kesai Wang, Erfu Yang, and Maurizio Faccio. ”enhancing wisdom manufacturing as industrial metaverse for industry and society 5.0”. *Journal of Intelligent Manufacturing*, Nov 2022.

[46] Zaid Abbas Al-Sabbag, Chul Min Yeum, and Sriram Narasimhan. Interactive defect quantification through extended reality. *Advanced Engineering Informatics*, 51:101473, Jan 2022.

[47] Zaid Abbas Al-Sabbag, Chul Min Yeum, and Sriram Narasimhan. Enabling human–machine collaboration in infrastructure inspections through mixed reality. *Advanced Engineering Informatics*, 53:101709, Aug 2022.

[48] Sainab Feroz and Saleh Abu Dabous. Uav-based remote sensing applications for bridge condition assessment. *Remote Sensing*, 13(9):1809, 2021.

[49] Juan A Besada, Luca Bergesio, Iván Campaña, Diego Vaquero-Melchor, Jaime López-Araquistain, Ana M Bernardos, and Jose R Casar. Drone mission definition and implementation for automated infrastructure inspection using airborne sensors. *Sensors*, 18(4):1170, 2018.

[50] Jacob J Lin, Amir Ibrahim, Shubham Sarwade, and Mani Golparvar-Fard. Bridge inspection with aerial robots: Automating the entire pipeline of visual data capture, 3d mapping, defect detection, analysis, and reporting. *Journal of Computing in Civil Engineering*, 35(2):04020064, 2021.

[51] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[52] Ramsundar Kalpagam Ganesan. *Mediating human-robot collaboration through mixed reality cues*. PhD thesis, Ph. D. Dissertation. Arizona State University, 2017.

[53] Maram Khatib, Khaled Al Khudir, and Alessandro De Luca. Human-robot contactless collaboration with mixed reality interface. *Robotics and Computer-Integrated Manufacturing*, 67:102030, 2021.

[54] Anthony Webster, Steven Feiner, Blair MacIntyre, William Massie, and Theodore Krueger. Augmented reality in architectural construction, inspection and renovation. In *Proc. ASCE Third Congress on Computing in Civil Engineering*, volume 1, page 996, 1996.

[55] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, 6(4):355–385, 1997.

[56] Julie Carmigniani, Borko Furht, Marco Anisetti, Paolo Ceravolo, Ernesto Damiani, and Misa Ivkovic. Augmented reality technologies, systems and applications. *Multimedia tools and applications*, 51(1):341–377, 2011.

[57] Amir H Behzadan, Suyang Dong, and Vineet R Kamat. Augmented reality visualization: A review of civil infrastructure system applications. *Advanced Engineering Informatics*, 29(2):252–267, 2015.

[58] Vineet R Kamat and Sherif El-Tawil. Evaluation of augmented reality for rapid assessment of earthquake-induced building damage. *Journal of computing in civil engineering*, 21(5):303–310, 2007.

[59] Rebecca Napolitano, Zachary Liu, Carl Sun, and Branko Glisic. Combination of image-based documentation and augmented reality for structural health monitoring and building pathology. *Frontiers in Built Environment*, 5:50, 2019.

[60] Fernando Moreu, Brian Bleck, Shreya Vemuganti, David Rogers, and David Mascarenas. Augmented reality tools for enhanced structural inspection. *Structural Health Monitoring*, 2, 2017.

[61] D Maharjan, M Agüero, D Mascarenas, R Fierro, and F Moreu. Enabling human-infrastructure interfaces for inspection using augmented reality. *Structural Health Monitoring*, page 1475921720977017, 2020.

[62] Enes Karaaslan, Ulas Bagci, and Fikret Necati Catbas. Artificial intelligence assisted infrastructure assessment using mixed reality systems. *Transportation Research Record*, 2673(12):413–424, 2019.

[63] Manash Pratim Das, Zhen Dong, and Sebastian Scherer. Joint point cloud and image based localization for efficient inspection in mixed reality. *arXiv:1811.02563 [cs]*, Nov 2018.

[64] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *2011 International Conference on Computer Vision*, page 667–674, Nov 2011.

[65] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. *arXiv:1803.10368 [cs]*, Apr 2018.

[66] Rih-Teng Wu and Mohammad Reza Jahanshahi. Data fusion approaches for structural health monitoring and system identification: Past, present, and future. *Structural Health Monitoring*, page 147592171879876, Sep 2018.

[67] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, page 12708–12717. IEEE, Jun 2019.

[68] Jeenal Vora, Santosh Nair, Anand K Gramopadhye, Andrew T Duchowski, Brian J Melloy, and Barbara Kanki. Using virtual reality technology for aircraft visual inspection training: presence and comparison studies. *Applied ergonomics*, 33(6):559–570, Nov 2002.

[69] Christian Linn, Simon Bender, Joshua Prosser, Kevin Schmitt, and Dirk Werth. Virtual remote inspection—a new concept for virtual reality enhanced real-time maintenance. In *2017 23rd International Conference on Virtual System & Multimedia (VSMM)*, pages 1–6, Dublin, Oct 2017. IEEE, IEEE.

[70] Jing Du, Yangming Shi, Zhengbo Zou, and Dong Zhao. Covr: Cloud-based multiuser virtual reality headset system for project communication of remote users. *Journal of Construction Engineering and Management*, 144(2):04017109, Feb 2018.

[71] Immanuel John Samuel, Ossama Salem, and Song He. Defect-oriented supportive bridge inspection system featuring building information modeling and augmented reality. *Innovative Infrastructure Solutions*, 7(4):1–17, Jun 2022.

[72] Urs Riedlinger, Florian Klein, Marcos Hill, Christian Lambracht, Sonja Nieborowski, Ralph Holst, Sascha Bahlau, and Leif Oppermann. Evaluation of mixed reality support for bridge inspectors using bim data-digital prototype for a manual task with a long-lasting tradition. *i-com: Vol. 21, No. 2*, 2022.

[73] Massimo Vaccarini, Alessandro Carbonari, Francesco Spegni, and Alberto Giretti. Enhancing bim through mixed reality for facility management. In *Building Information Modeling-A Sustainable Approach and Emerging Technologies*. IntechOpen, Sep 2022.

[74] Carmine Elvezio, Mengu Sukan, Ohan Oda, Steven Feiner, and Barbara Tversky. Remote collaboration in ar and vr using virtual replicas. In *ACM SIGGRAPH 2017 VR Village*, SIGGRAPH '17, pages 1–2. Association for Computing Machinery, New York, NY, USA, Jul 2017.

[75] Barrett Ens, Joel Lanir, Anthony Tang, Scott Bateman, Gun Lee, Thammathip Piumsomboon, and Mark Billinghurst. Revisiting collaboration through mixed reality: The evolution of groupware. *International Journal of Human-Computer Studies*, 131:81–98, Nov 2019.

[76] Peng Wang, Xiaoliang Bai, Mark Billinghurst, Shusheng Zhang, Xiangyu Zhang, Shuxia Wang, Weiping He, Yuxiang Yan, and Hongyu Ji. Ar/mr remote collaboration on physical tasks: A review. *Robotics and Computer-Integrated Manufacturing*, 72:102071, Dec 2021.

[77] Alexander Schäfer, Gerd Reis, and Didier Stricker. A survey on synchronous augmented, virtual and mixed reality remote collaboration systems. *ACM Computing Surveys (CSUR)*, 2021.

[78] Morteza Dianatfar, Jyrki Latokartano, and Minna Lanz. Review on existing vr/ar solutions in human–robot collaboration. *Procedia CIRP*, 97:407–411, Jan 2021.

[79] Eimei Oyama, Kohei Tokoi, Ryo Suzuki, Sousuke Nakamura, Naoji Shiroma, Norifumi Watanabe, Arvin Agah, Hiroyuki Okada, and Takashi Omori. Augmented reality and mixed reality behavior navigation system for telexistence remote assistance. *Advanced Robotics*, 35(20):1223–1241, Oct 2021.

[80] Mattias Seeman, Mathias Broxvall, and Alessandro Saffiotti. Virtual 360 panorama for remote inspection. In *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 1–5, Rome, Italy, Sep 2007. IEEE, IEEE.

[81] Gun A Lee, Theophilus Teo, Seungwon Kim, and Mark Billinghurst. Mixed reality collaboration through sharing a live panorama. In *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications*, pages 1–4. ACM Press, Bangkok, Thailand, 2017.

[82] Gun A Lee, Theophilus Teo, Seungwon Kim, and Mark Billinghurst. A user study on mr remote collaboration using live 360 video. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 153–164, Munich, Germany, Oct 2018. IEEE, IEEE.

[83] Theophilus Teo, Louise Lawrence, Gun A Lee, Mark Billinghurst, and Matt Adcock. Mixed reality remote collaboration combining 360 video and 3d reconstruction. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–14, 2019.

[84] Thammathip Piumsomboon, Gun A Lee, Andrew Irlitti, Barrett Ens, Bruce H Thomas, and Mark Billinghurst. On the shoulder of the giant: A multi-scale mixed reality collaboration with 360 video sharing and tangible interaction. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–17, 2019.

[85] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.

[86] Chunfeng Xu, Yange Wang, Wei Quan, and He Yang. Multi-person collaborative interaction algorithm and application based on hololens. In Vipul Jain, Srikanta Patnaik, Florin Popențiu Vlădicescu, and Ishwar K. Sethi, editors, *Recent Trends in Intelligent Computing, Communication and Devices*, Advances in Intelligent Systems and Computing, pages 303–315. Springer, Singapore, 2020.

[87] Patrick Hübner, Martin Weinmann, and Sven Wursthorn. Marker-based localization of the microsoft hololens in building models. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(1), 2018.

[88] Laura Pérez-Pachón, Parivrudh Sharma, Helena Brech, Jenny Gregory, Terry Lowe, Matthieu Poyade, and Flora Gröning. Effect of marker position and size on the registration accuracy of hololens in a non-clinical setting with implications for high-precision surgical tasks. *International journal of computer assisted radiology and surgery*, 16(6):955–966, 2021.

[89] Yihong Wu, Fulin Tang, and Heping Li. Image-based camera localization: an overview. *Visual Computing for Industry, Biomedicine, and Art*, 1(1):8, Sep 2018.

[90] Sean Ong and Varun Kumar Siddaraju. *"Azure Spatial Anchors"*, page 175–188. Apress, Berkeley, CA, 2021.

[91] Microsoft. "microsoft mesh", 2021.

[92] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[93] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8623–8632, 2020.

[94] Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, February 2020.

[95] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 373–381, 2016.

[96] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5306, 2019.

[97] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[98] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998. IEEE, 2011.

[99] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, Cambridge, UK; New York, 2nd ed edition, 2003.

[100] Christer Ericson. *Real-time collision detection*. CRC Press, 2004.

[101] Younhee Lee and Woong Lim. Shoelace formula: Connecting the area of a polygon and the vector cross product. *The Mathematics Teacher*, 110(8):631–636, 2017.

[102] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[103] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[104] Itseez. Open source computer vision library. https://github.com/itseez/opencv, 2020.

[105] Michael Riis Andersen, Thomas Jensen, Pavel Lisouski, Anders Krogh Mortensen, Mikkel Kragh Hansen, Torben Gregersen, and PJAU Ahrendt. Kinect depth sensor evaluation for computer vision applications. Technical report, Aarhus University, 2012.

[106] Trevor S Breusch and Adrian R Pagan. A simple test for heteroscedasticity and random coefficient variation. *Econometrica: Journal of the Econometric Society*, pages 1287–1294, 1979.

[107] Robert Castle, Georg Klein, and David W. Murray. Video-rate localization in multiple maps for wearable augmented reality. In *2008 12th IEEE International Symposium on Wearable Computers*, page 15–22, Sep 2008.

[108] Clemens Arth, Daniel Wagner, Manfred Klopschitz, Arnold Irschara, and Dieter Schmalstieg. *Wide Area Localization on Mobile Phones*. IEEE, Oct 2009.

[109] Jing Li, Chengyi Wang, Xuejie Kang, and Qiang Zhao. Camera localization for augmented reality and indoor positioning: a vision-based 3d feature database approach. *International Journal of Digital Earth*, 13(6):727–741, Jun 2020.

[110] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. *Proceedings of the IEEE international conference on computer vision*, May 2015.

[111] Meng Xu, Youchen Wang, Bin Xu, Jun Zhang, Jian Ren, Stefan Poslad, and Pengfei Xu. A critical analysis of image-based camera pose estimation techniques. *arXiv:2201.05816 [cs]*, Jan 2022.

[112] Grant Schindler, Matthew Brown, and Richard Szeliski. City-scale location recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, page 1–7, Jun 2007.

[113] Wei Zhang and Jana Kosecka. Image based localization in urban environments. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, page 33–40, Jun 2006.

[114] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, page 2599–2606, Jun 2009.

[115] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, page 1–22, 2004.

[116] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. *arXiv:1511.07247 [cs]*, May 2016.

[117] Peter Decker, Dietrich Paulus, and Tobias Feldmann. Dealing with degeneracy in essential matrix estimation. *2008 15th IEEE International Conference on Image Processing*, Jan 2008.

[118] R. Carceroni, A. Kumar, and K. Daniilidis. Structure from motion with known camera positions. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, page 477–484, Jun 2006.

[119] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 2100–2106, Nov 2013.

[120] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. Are large-scale 3d models really necessary for accurate visual localization? In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 6175–6184, Jul 2017.

[121] X. Gao, Xiaorong Hou, Jianliang Tang, and H. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2003.

[122] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, Feb 2009.

[123] Luis Ferraz, Xavier Binefa, and Francesc Moreno-Noguer. Very fast solution to the pnp problem with algebraic outlier rejection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 501–508, 2014.

[124] Mathieu Labbe and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[125] Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Apr 2021.

[126] Jiarong Lin and Fu Zhang. R3live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package. *arXiv:2109.07982 [cs]*, Sep 2021.

[127] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv:1511.00561 [cs]*, Oct 2016.

[128] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, Jun 2014.

[129] Microsoft. *HoloLens 2 hardware*. Microsoft, 2019.

[130] Livox avia specs, 2022.

[131] Turtlebot2, 2021.

[132] qm13. Azure kinect dk hardware specifications, 2021.

[133] Christopher Crick, Graylin Jay, Sarah Osentoski, Benjamin Pitzer, and Odest Chadwicke Jenkins. Rosbridge: Ros for non-ros users. In *Robotics Research*, pages 493–504. Springer, 2017.

[134] Unity game engine, 2022.

[135] Mixed reality toolkit, 2022.

[136] Magic Leap. Magic leap, 2019.

[137] Ros-sharp, Mar 2022.

[138] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, page 2564–2571, Nov 2011.

[139] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[140] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: Repeatable and reliable detector and descriptor. *arXiv:1906.06195 [cs]*, Jun 2019.

[141] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, page 224–236, 2018.

[142] Amila Jakubović and J. Velagic. Image feature matching and object detection using brute-force matchers. *2018 International Symposium ELMAR*, 2018.

[143] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. *arXiv:1911.11763 [cs]*, Mar 2020. arXiv: 1911.11763.

[144] Superglue pre-trained network, Mar 2022.

[145] Mixedreality-spectatorview, 2019.

[146] Zaid Abbas Al-Sabbag, Chul Min Yeum, and Sriram Narasimhan. Distributed collaborative inspections through smart infrastructure metaverse. *Automation in Construction*, (AUTCON-D-23-02332R1), Aug 2023.

[147] OmniCV. "omnicv", 2020.

[148] Long Quan and Zhongdan Lan. "linear n-point camera pose determination". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):774–780, Aug 1999.

[149] Martin A. Fischler and Robert C. Bolles. "random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Communications of the ACM*, 24(6):381–395, Jun 1981.

[150] Kyle Simek. "dissecting the camera matrix", 2013.

[151] Azra Fetić, Davor Jurić, and Dinko Osmanković. The procedure of a camera calibration using camera calibration toolbox for matlab. In *2012 Proceedings of the 35th International Convention MIPRO*, page 1752–1757, May 2012.

[152] Jorge Beltrán, Carlos Guindel, and Fernando García. Automatic extrinsic calibration method for lidar and camera sensor setups. *arXiv:2101.04431 [cs]*, Jan 2021.

[153] Chongjian Yuan, Xiyuan Liu, Xiaoping Hong, and Fu Zhang. Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments. *arXiv:2103.01627 [cs]*, Jun 2021.

[154] X.W. Ye, Jin Tao, and C.B. Yun. "a review on deep learning-based structural health monitoring of civil infrastructures". *Smart Structures and Systems*, 24:567–585, Nov 2019.

[155] Intel. "depth camera d455", 2021.

[156] Livox. "livox avia", 2021.

[157] Theta. "ricoh theta x", 2022.

[158] Meta. "oculus quest 2", 2022.

[159] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "are we ready for autonomous driving? the kitti vision benchmark suite". In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, page 3354–3361, Jun 2012.

[160] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. ”loftr: Detector-free local feature matching with transformers”. (arXiv:2104.00680), Apr 2021. arXiv:2104.00680 [cs].

[161] Unity. ”unity real-time development platform — 3d, 2d, vr and ar engine”, 2021.

[162] M. Jogan and A. Leonardis. ”robust localization using panoramic view-based recognition”. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, page 136–139 vol.4, Sep 2000.

[163] Clemens Arth, Manfred Klopschitz, Gerhard Reitmayr, and Dieter Schmalstieg. *”Real-time self-localization from panoramic images on mobile devices”*. Oct 2011.

[164] Clemens Arth, Alessandro Mulloni, and Dieter Schmalstieg. ”exploiting sensors on mobile phones to improve wide-area localization”. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, page 2152–2156, Nov 2012.

[165] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. ”global localization from monocular slam on a mobile phone”. *IEEE Transactions on Visualization and Computer Graphics*, 20(4):531–539, Apr 2014.

[166] Changhee Won, Hochang Seok, Zhaopeng Cui, Marc Pollefeys, and Jongwoo Lim. ”omnislam: Omnidirectional localization and dense mapping for wide-baseline multi-camera systems”. (arXiv:2003.08056), Mar 2020. arXiv:2003.08056 [cs].

[167] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. ”image matching using sift, surf, brief and orb: Performance comparison for distorted images”. (arXiv:1710.02726), Oct 2017. arXiv:1710.02726 [cs].

[168] Shuai Liu, Lingli Zhao, Junsheng Li, and Qun Cai. ”th sift features matching for spherical panoramic images”. In *11th IEEE International Conference on Control and Automation (ICCA)*, page 914–917, Jun 2014.

[169] Daniel Girardeau-Montaut. ”cloudcompare”, 2012.

[170] Microsoft. *MixedReality-SpectatorView*, 2019.

[171] Shen-En Chen, Wanqiu Liu, Haitao Bian, and Ben Smith. 3d lidar scans for bridge damage evaluations. *Sixth Congress on Forensic Engineering*, page 487–495, Jan 2013.

[172] Phillip S Dunston et al. Evaluation of augmented reality in steel column inspection. *Automation in Construction*, 18(2):118–129, 2009.

[173] Keang Ang Kouch, Kriengsak Panuwatwanich, and Pakawat Sancharoen. Application of wearable augmented reality system in expressway inspection. *EDUCATING BUILDING PROFESSIONALS FOR THE FUTURE IN THE GLOBALISED WORLD*, page 208, 2018.

[174] David DL Mascareñas, JoAnn P Ballor, Oscar L McClain, Miranda A Mellor, Chih-Yu Shen, Brian Bleck, John Morales, Li-Ming R Yeong, Benjamin Narushof, Philo Shelton, Eric Martinez, Yongchao Yang, Alessandro Cattaneo, Troy A Harden, and Fernando Moreu. Augmented reality for next generation infrastructure inspections. *Structural Health Monitoring*, 20(4):1957–1979, Jul 2021.

[175] Ramakrishna Perla, Gaurav Gupta, Ramya Hebbalaguppe, and Ehtesham Hassan. Inspectar: An augmented reality inspection framework for industry. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, page 355–356, Sep 2016.

[176] Hugo Silva, Ricardo Resende, and Maurício Breternitz. Mixed reality application to support infrastructure maintenance. In *2018 International Young Engineers Forum (YEF-ECE)*, page 50–54, May 2018.

[177] Dezhen Song, Qiang Hu, Ni Qin, and K. Goldberg. Automating inspection and documentation of remote building construction using a robotic camera. In *IEEE International Conference on Automation Science and Engineering, 2005.*, page 172–177, Aug 2005.

[178] Y. M. Wang, Y. Li, and J. B. Zheng. A camera calibration technique based on opencv. In *The 3rd International Conference on Information Sciences and Interaction Sciences*, page 403–406, Jun 2010.

[179] Chul Min Yeum and Shirley J. Dyke. Vision-based automated crack detection for bridge inspection: Vision-based automated crack detection for bridge inspection. *Computer-Aided Civil and Infrastructure Engineering*, 30(10):759–770, Oct 2015.

[180] Lipu Zhou, Zimo Li, and Michael Kaess. Automatic extrinsic calibration of a camera and a 3d lidar using line and plane correspondences. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 5562–5569. IEEE, Oct 2018.

[181] Nicola Mosca, Gaetano Pernisco, Maria Di Summa, Vito Renò, Massimiliano Nitti, and Ettore Stella. Virtual and augmented reality for quality control of aircraft interiors. In *International Conference on Image Analysis and Processing*, pages 225–234. Springer, 2022.

[182] Gareth Byers and SeyedReza Razavialavi. Virtual reality for the robot-enabled construction site inspection-a. *Transforming Construction with Reality Capture Technologies*, 2022.

[183] Jiale Shang, Huiqing Wang, Xunfu Liu, Yang Yu, and Qiuhua Guo. Vr+ ar industrial collaboration platform. In *2018 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 162–163. IEEE, 2018.

[184] Christopher Brown, Jamison Hicks, Christina H Rinaudo, and Reuben Burch. The use of augmented reality and virtual reality in ergonomic applications for education, aviation, and maintenance. *Ergonomics in Design*, page 10648046211003469, Mar 2021.

[185] Ministry of Transportation (MTO). *"Ontario structure inspection manual: OSIM."*. Ontario Ministry of Transportation, Bridge Office, St. Catharines, ON, 2000.

[186] Henrik Eschen, Tobias Kötter, Rebecca Rodeck, Martin Harnisch, and Thorsten Schüppstuhl. "augmented and virtual reality for inspection and maintenance processes in the aviation industry". *Procedia Manufacturing*, 19:156–163, 2018.

[187] Meta. "meta", 2023.

[188] Laura Pérez-Pachón, Parivrudh Sharma, Helena Brech, Jenny Gregory, Terry Lowe, Matthieu Poyade, and Flora Gröning. "effect of marker position and size on the registration accuracy of hololens in a non-clinical setting with implications for high-precision surgical tasks". *International Journal of Computer Assisted Radiology and Surgery*, 16(6):955–966, Jun 2021.

[189] P. Hübner, M. Weinmann, and S. Wursthorn. "marker-based localization of the microsoft hololens in building models". *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII–1:195–202, Sep 2018.

[190] Zaid Abbas Al-Sabbag. Easy camera lidar calibration. `https://github.com/zaalsabb/easy_camera_lidar_calibration`, 2023.

APPENDICES

Appendix A

Camera Calibration

Before the start of the inspection session, visual cameras used in the robot and the MR headset must be intrinsically calibrated using a checkerboard procedure unless calibration parameters are provided by the manufacturer [178, 151]. This procedure involves capturing many images (50-100 images) of the checkerboard in many different angles and distances. The 3×3 camera intrinsics matrix (K) and distortion parameters are then computed that minimize the re-projection error of the checkerboard corners. Fig. A.1 shows the checkerboard calibration board used for calibrating the robot's camera.

The camera intrinsics matrix for the robot's camera (K^r) and the MR headset's camera (K^q) must be computed or known in advance. If lens distortion is present, the images must be undistorted using the calibration parameters obtained from the checkerboard procedure.

The robot's LiDAR sensor is also calibrated extrinsically to its main visual camera (c_r) using a camera-LiDAR calibration process [180, 152, 153]. This procedure involves capturing at least one image and one LiDAR scan, registering the matching points (e.g., checkerboard corners) between the image and the 3D scan, and then using those 2D-3D correspondences to solve for the relative rotation matrix (R) and translation vector (t) using the PnP algorithm. Fig. A.2 shows an example of a LiDAR scan captured from the Livox LiDAR scanner for the calibration checkerboard.

Unfortunately, standard software packages such as MATLAB were not able to automatically register the corners of the checkerboard in the LiDAR scan due to the unique non-repeating scanning pattern of the Livox scanner. Therefore, I developed a new package called Easy Camera-LiDAR Calibration to allow users to manually select matching points between the image and 3D LiDAR scan, in cases where these points cannot be automatically detected by software. I published this package on GitHub [190].

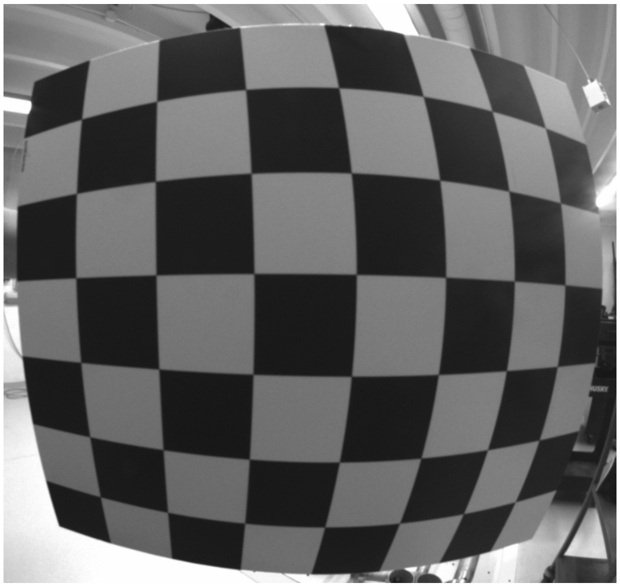


Figure A.1: Calibration checkerboard used for intrinsic camera calibration of robot's camera to obtain K^r.

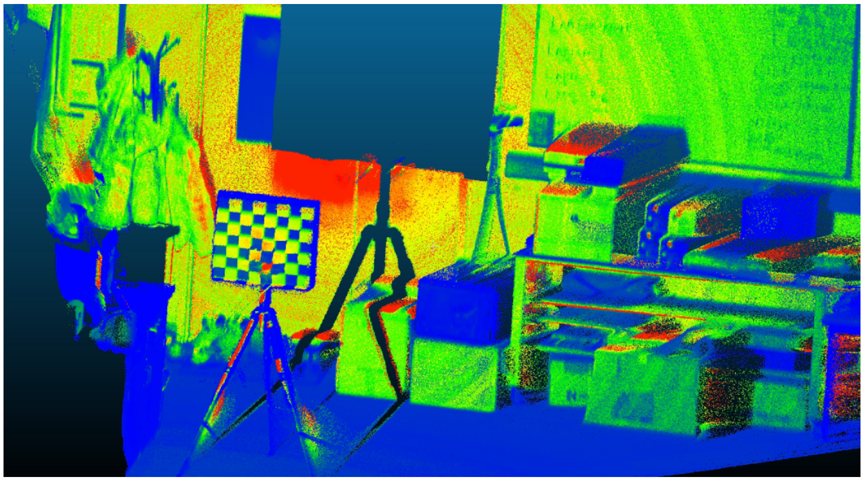


Figure A.2: LiDAR scan captured from Livox LiDAR scanner of calibration checkerboard. Overlapping regions between the image and LiDAR scan are used for extrinsic calibration for registering the camera and LiDAR sensors.