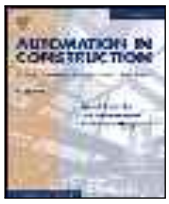
Automation in Construction 181 (2026) 106682

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon



SLAM-centric visual inspection of civil infrastructure

Nicholas Charron [a,*], Jake McLaughlin [a], Sriram Narasimhan [b]

[a] Civil & Environmental Engineering, University of Waterloo, 200 University Ave W, Waterloo, N2L 3G1, Ontario, Canada
[b] Civil & Environmental Engineering, University of California, Los Angeles, 405 Hilgard Avenue, Los Angeles, 90095, CA, USA

ARTICLE INFO

Dataset link: https://github.com/nickcharron/beam_inspection, https://beamrobotics.github.io/dataset.html

Keywords:
Inspection
Infrastructure
SLAM
Robotics
Defect tracking
Mapping

ABSTRACT

Existing robot-aided inspection methods suffer from inconsistent map accuracy, unreliable defect measurements, and platform-specific designs. This paper investigates whether a SLAM-centric framework can enable precise, repeatable, and platform-agnostic visual inspections. The framework integrates lidar–camera–inertial SLAM, offline trajectory refinement, inspection-map generation, defect extraction from imagery, and 3D ray-tracing to project defects into a unified map. The approach confirms that accurate defect localization, dimensional quantification, and dense inspection maps can be produced in real-world scenarios. This finding benefits infrastructure owners and inspectors by providing an end-to-end solution for robot-aided inspections that enable faster, safer, and more objective assessments compared to current qualitative workflows. The released datasets and software establish a foundation for future research on long-term defect monitoring and inspection automation.

1. Introduction

Over the last decade, there has been significant activity related to the inspection of critical infrastructure using mobile robotics. A wide range of custom or off-the-shelf mobile robotics platforms have been proposed for infrastructure inspection, such as climbing robots, ground robots, legged robots, aerial robots, surface vessels, and underwater robots [1]. Early work included platforms used only as a means to collect sensor data from hard-to-access locations, where data is analyzed on its own without any regard for where in the 3D world the data was collected. Over the last decade, the synergistic developments in the areas of robotics, especially simultaneous localization and mapping (SLAM), computer vision, and machine learning, have accelerated and advanced this research in new directions. Recent activities have shown how mobility and advanced onboard perception, such as lidar and high-resolution cameras, can be combined for unprecedented insights and quantification of the infrastructure condition.

While the developments in this area are impressive (we review pertinent literature in the next section), certain key challenges commonly encountered in infrastructure applications remain. For instance, the majority of current approaches rely on image-based 3D mapping, such as obtained using Structure from Motion (SfM). While visual information is important and intuitive in inspection applications, SfM is well-known to be unreliable and inaccurate in challenging inspection environments such as assets with large spatial extent or textureless surfaces (e.g., concrete). SLAM methodologies that integrate multiple sensors are known to address such limitations where complementary capabilities of lidar, cameras, and inertial measurement units (IMUs) can be leveraged for producing high-quality inspection maps. Few inspection systems in literature have implemented multi-sensor SLAM, and those that have, do not combine lidar, camera, and inertial measurements in a tightly-coupled way for robust SLAM performance even in degenerate environments. Aside from the lack of robust SLAM algorithm implementations in the domain of infrastructure inspection, the problem of sensor selection and mobility has not received the attention it deserves. Most works are constrained by a specific robotic platform which is not generalizable to various inspection types. Hence, there is an urgent need to develop an end-to-end(E2E) SLAM-centric inspection framework that is platform agnostic and can detect, quantify, and track defect (inspection objects of interest, e.g., spalls, cracks) changes in 3D maps of environments common to infrastructure inspections over multiple inspections.

The specific contributions of this paper include:

- An inspection map generation methodology that includes online SLAM, followed by offline trajectory refinement and de-coupled inspection map generation
- An image ray-tracing algorithm to map pixel data to a pre-generated unordered lidar pointcloud that is computationally efficient and precise without requiring meshing or other map alterations

* Corresponding author.
E-mail address: ncharron@uwaterloo.ca (N. Charron).

https://doi.org/10.1016/j.autcon.2025.106682
Received 16 March 2025; Received in revised form 18 November 2025; Accepted 19 November 2025
Available online 29 November 2025

- A versatile and configurable SLAM-centric inspection design that is platform-agnostic while enabling precise SLAM and high-quality inspection data
- We provide the first-of-its-kind datasets focused on inspection applications and open-source the datasets and software, enabling a baseline for future research in robot-based inspection of infrastructure

This paper is organized as follows. In Section 2, we present the relevant works that study the use of 3D map-based inspection of infrastructure using mobile platforms. In Section 3, we present the E2E inspection methodology including a description of the proposed map generation methodology and the defect ray-tracing methodology. In Section 4, we present the robotic system design, including sensor requirements and preferred configurations. In Section 5, we present the results, including a description of the datasets shown herein, and example results for all key steps of the inspection methodology.

2. Related work

The literature in the area of robotics and SLAM is vast and it is not practical to include an exhaustive review here. Of special mention are papers that have focused on robotics for construction planning and automation tasks (e.g., [2–5]). However, directly relevant to this work, we will only review literature that is focused on the inspection of civil infrastructure applications such as bridges and hence considered close to this work. More specifically, we will review works that have studied the use of SLAM for inspection. Early works that use SLAM for inspection generally focused on creating 2D maps for inspection of 2D surfaces such as bridge decks. Examples include the ROCIM system presented by Lim et al. [6], the RABIT robot by La et al. [7,8], the Uncrewed Ground Vehicle (UGV) by Gibb et al. [9], and the bearing inspection UGV by Peel, H. [10]. These works, while focused on 2D elements such as decks, provide impressive results and constitute early attempts to bring robotics to inspect civil infrastructure.

More recently, multi-sensor systems that employ 3D SLAM have been proposed for inspections. These works (e.g., [11,12]) are examples that utilize cameras and lidars to perform SLAM to detect defects of interest. However, they depend on tracking visual features, are limited in their flexibility on sensor and layout choice, or tied to a specific SLAM or meshing method e.g., (R^3LIVE [13]), which is challenging for mapping complex 3D geometry from noisy pointclouds. Most recently, Ge et al. [14] proposed an E2E inspection framework using SLAM. However, their SLAM does not use critical sensors such as IMUs or Cameras (cameras are only used for defect detection), which have been shown in SLAM literature to significantly improve reliability [15,16].

More advanced perception onboard Uncrewed Aerial Vehicles (UAVs) have been proposed. For example, [17,18] utilize GPS, 3D lidar, and multiple cameras for multi-sensor SLAM. Their work highlights robustness and synchronization challenges, underscoring the challenges in using SLAM for inspection. Importantly, studies on a comparison of sensor types and specifications, which led to the design of their platforms, and issues resulting from lack of synchronized data remain unaddressed. Other works utilizing UGVs also do not address issues related to sensor selection or synchronization (e.g., [19]), or do not include critical sensors such as IMUs ([14]) that are known to compensate for motion distortion and degenerate mapping scenarios [15]. Furthermore, these studies directly utilize maps created for SLAM to perform inspection tasks. Among other issues, SLAM maps are inherently sparse and do not contain the density necessary for quantification in inspection applications.

In inspection applications, the ability to overlay defects on a map with scale is crucial. While it is well-known that projection methods (3D points projected to 2D image pixels), ray-tracing (2D to 3D) methods are fraught with challenges. [Ray-tracingRaytracing is computationally expensive, even on large GPUs, especially for large pointclouds and high-resolution images needed for inspection. To accelerate ray-tracing, scenes are often structured as meshes or BVH [20], but these methods are ineffective for unorganized, noisy inspection maps. Attempts to address such issues such as [21] suffer from effects of noise, inconsistent density, and size effects.

In summary, of all the inspection literature that use multi-sensors SLAM for 3D map generation, there are three important areas of research that can be considered as research gaps. First, to our knowledge, no authors have demonstrated tightly-coupled Lidar–Camera–Inertial SLAM for inspection. Furthermore, inspection methodologies found in the literature still have significant limitations in mapping and map labeling that do not allow for reliable quantification and tracking of defects. Lastly, to our knowledge, an inspection system that is platform-independent and can be ported to any robotic or non-robotic system that suits the needs of the inspection is yet to be proposed. Furthermore, issues such as sensor selection and layout, and specifically how these choices affect SLAM and inspection results, have not received the attention they deserve in the literature.

3. Inspection methodology

Fig. 1 illustrates the overall inspection methodology or framework proposed in this paper. It will be shown that the steps in this method are general to most visual infrastructure inspection types and thus can be extended to many applications beyond bridge and parking garage inspection. The ultimate output of the methodology is to generate a high-quality, dense, and visually accurate 3D map of the structure, with defect information stored and easily viewable to inspectors.

From an implementation standpoint, the method starts with multi-modal data collected from an inspection platform, which could be a UAV, UGV, or a Uncrewed Surface Vessel (USV). We utilize the open-source set of software libraries and tools in ROS to collect and process data. Note that the steps are described as an offline process, however, due to the flexibility of ROS, many of these processes can be run in real-time on data being streamed by the device to a processing computer. Due to the nature of most inspection tasks, real-time processing aspects are not the focus here; these processes are not too computationally expensive to run in real-time should they be implemented and tuned for this purpose. The methodology can be divided into three major tasks: (1) trajectory and map generation, (2) defect detection using images, and (3) map labeling and defect quantification, which combines the defects generated in (1) with the 3D map generated in (2) and uses these results to generate the final inspection quantities that inspectors need. The following three subsections will describe the details of those steps. This work relies on qualitative assessment to validate the approach, as no benchmarks exist to evaluate inspection maps.

3.1. Trajectory & Map generation

Fig. 2 shows the full mapping workflow, including inputs and outputs of each step in the pipeline. The trajectory and map generation component first starts through implementation of online (processed in real-time) SLAM using a combination of lidar, camera, and inertial data. The online SLAM was designed to be similar to the state-of-the-art LVI-SAM [16], and we do not claim novelty in the specific online SLAM methodology used herein. We re-implemented the LVI-SAM architecture since the open source tooling made available by the authors is unmaintained and not designed to be versatile to different sensor systems. This process generates an initial estimate of the trajectory, with a low-rate pose synced to camera images and/or lidar scans (depending on the SLAM implementation used) and a high-rate pose that uses the IMU data to estimate the trajectory between the low-rate poses (as is standard in most modern SLAM implementations). Not only does the online SLAM provide initialization for the remaining two steps, but it can also be run in real-time during data collection so that inspectors can see coverage and ensure proper tracking of the

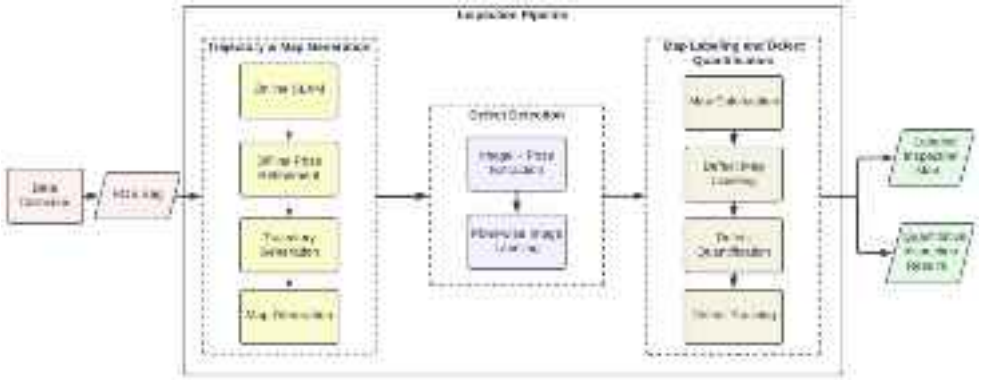


Fig. 1. Overall inspection workflow.

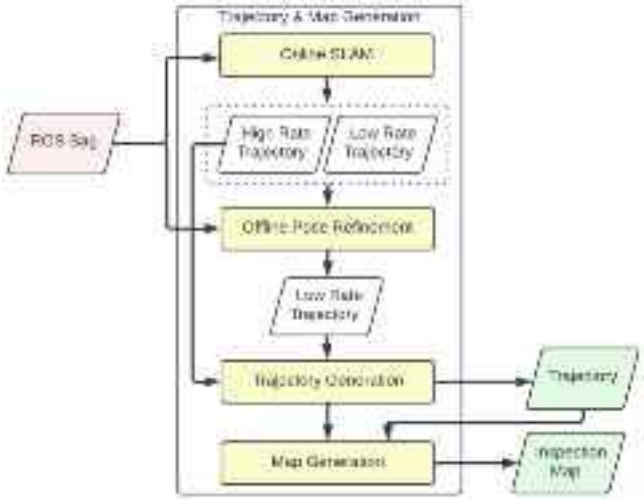


Fig. 2. Mapping component of the inspection workflow.

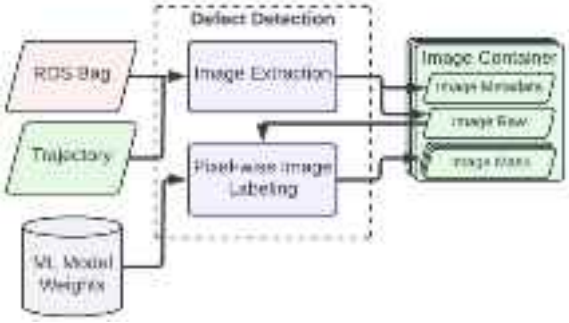


Fig. 3. Defect detection component of the inspection workflow.

inspection device on site. It should also be noted that this framework is not tied to a specific SLAM implementation. Instead, the SLAM used in this work is designed to resemble classic state-of-the-art SLAM methods, and the rest of the inspection framework is designed around the typical outputs from any online SLAM. The second step in the methodology is offline pose refinement which is aimed at optimizing for the best possible trajectory given more compute resources and no restrictions on compute time.

The offline refinement performs a batch optimizer given a scan-to-map registration approach similar to the online SLAM, but with a much larger map size and more computationally expensive scan-matching. Furthermore, loop closure constraints are added for each scan that revisits a similar location by registering against N previous scans, resulting in many registration constraints for every loop closure location. For loop closure detection, we use Euclidean distance and ScanContext [22] and aggregate many neighboring scans to get a dense map for comparisons. Next, the high-rate poses from online SLAM are combined with the refined low-rate poses, and a high-rate refined pose trajectory is generated. This approach for generating a high-rate and globally consistent trajectory was first introduced in our work in [23]. This trajectory is then fed to map generation, which produces an inspection map. This de-coupled map generation approach was first introduced in our work in [23], in the context of integrating sonar, lidar, camera and IMU measurements from an USV.

3.2. Defect detection

Fig. 3 shows the full defect detection workflow, including inputs and outputs of each step in the pipeline. There are two major components to this step. The first is extracting images and metadata from various camera streams. The second is labeling these images with defects of interest in such a way that they can be used for map labeling and quantification (see Section 3.3). The next sections will describe the proposed solution for accomplishing both these tasks.

3.2.1. Image and pose extraction

The goal of this step is to extract image information that will be useful for gathering inspection data from several cameras on a mobile platform. Ideally, high-quality images of defects of interest are desired, as well as full image coverage of the environment so maps can be completely colorized and visually realistic 3D maps can be generated for inspectors. Redundant images showing similar views should also be minimized since this will result in excessive data processing and storage requirements. This, therefore, creates a trade-off between the quantity of data and view coverage.

The image extraction process proposed here simply uses the trajectory to extract images given a user-specified minimum rotation or translation in between each consecutively selected image. Furthermore, the trajectory can be used to filter out images that may have high distortion by calculating the rotation rates.

3.2.2. Pixel-wise image labeling

This section refers to defect labeling as the process of labeling defects in images extracted during the image extraction step. Although this step is a key component in the inspection pipeline, this component is designed to be relatively general in order to ensure the pipeline can be generalized to various inspection tasks. For example, defect

labeling for bridge and parking garage inspection would mostly consist of detecting cracks, spalls, and delamination of concrete. For new construction projects, perhaps the defect labeling would entail quantifying the number of piping in order to calculate the percent complete for assembling piping. The only requirement for the labeling is that pixels be labeled on the image to quantify some class type.

First, for a more general and automatic defect extraction technique that can run on the datasets used in this work, Defect Instance Segmentation YOLO, or DIS-YOLO [24] is used. DIS-YOLO is a fully convolutional model that uses the YOLOv3 [25] backbone and is trained to simultaneously detect and segment three concrete surface defects: cracks, spalls, and exposed rebar. The model was trained on the combination of three open datasets for bridge inspection and one dataset from the Hong Kong Highways department, resulting in 1433 labeled images.

Second, for a more human-in-the-loop approach, some of the images are also processed with the popular Segment Anything Model (SAM) [26]. SAM is the largest segmentation model to date, trained with over 1 billion masks on 11 million images. SAM is also designed to work using a prompt, making this model extremely versatile for any sort of human-in-the-loop segmentation task. With a small amount of human work, various sorts of defects or objects in an image can be quickly segmented. In our experience, large distinct area defects were easy to label with SAM, including spalls, and repair patches. On the other hand, linear defects such as cracks, or non-distinct defects such as discoloration, were harder to label with SAM and we had to resort to DIS-YOLO which was specifically trained for such defects.

Regardless of the methods used, all defect labeling methods must generate one or more segmentation maps. Once all segmentation masks are generated, the pixel labels can be back-projected for each mask, as well as the raw images, to the 3D map as presented in Section 3.3.

3.3. Map labeling and defect quantification

Given a 3D pointcloud map made specifically for inspection, a precise and dense trajectory, and a set of labeled images, such information can be combined into a single data structure that enables the following key features: (1) A to-scale map allowing all data to be geographically cataloged for easy localization relative to the structure, (2) an easy means to visualize all inspection data in one place from a computer or in 3D using a VR device, (3) a way to combine several observations of the same objects from different viewpoints, (4) a way to quantify defects that may have been partly observed from different images, and (5) a way to identify specific defects between consecutive inspections. This section describes all the sub-components that allow such data to be combined into a map in an efficient manner and how to extract important inspection data from such a map, including defect/object sizes and changes in size over time.

Fig. 4 shows the full defect detection workflow, including inputs and outputs of each step in the pipeline. The first two steps, which can run in parallel, are Map Colorization and Defect Map labeling. Since defects are stored as image masks, the same algorithms can be used to transfer defect information to the map as regular RGB information. Both these tasks are grouped into Section 3.3.1. The output of these two subcomponents is a 3D inspection map where points have their own labels describing their color and whether they belong to some defect or other class from the input masks. Given a labeled map, points can be grouped into objects/defects, and their sizes can be quantified (Section 3.3.2). Finally, if previous inspection maps exist of that same structure, these maps can be aligned, and objects can be tracked over time to detect changes in sizes or to quantify new objects not existing in previous inspections (Section 3.3.2).

3.3.1. 3D map labeling from images

Map labeling is defined here as the process of taking images or image masks with known poses and transferring the information from the image pixels to the 3D pointcloud map. For regular images, transferring RGB image fields to the map provides visual cues for the inspectors or even intensity fields from an infrared image that may be used to visually locate subsurface defects. For image masks, this includes transferring any defect or object classification field added in the defect labeling described in Section 3.2.2.

As discussed in Section 2, there are two general approaches to solve this: projection and ray-tracing. Since the inspection approach proposed in this work decouples the map generation from the remainder of the inspection pipeline, occlusions become a significant issue since the map points may have been generated from a wide range of viewpoints that differ from the image's viewpoint. These map points also may have been altered during the map creation, such as averaging points in each voxel during down-sampling. This means that it cannot be known which pixel captured which point during data collection since a point may be a combination of many other points captured at different times.

This work proposes a highly efficient custom ray-tracing approach. The approach uses a few key concepts to drastically increase computation time without degrading accuracy. These concepts are illustrated in Fig. 5. First, for each image, all points in the map are projected into the image. All points that do land on the image plane are removed from the map, resulting in a map that can be orders of magnitude smaller (Fig. 5(a)). This decreased map size significantly decreases the time for each volume search performed during [ray-tracingraytracing. Second, a hit map is created with equal dimensions to the image. This is a binary image where each pixel is true if at least one map point has been projected into that pixel (Fig. 5(b)). [Ray-tracingRaytracing then only needs to be performed for all pixels in the hit-map labeled as true. Raytracing is then performed normally, where a ray is extended from the camera center, passing through each pixel in the hit-map and extended until contacting the map (Fig. 5(c)). The final improvement which increases speed without degrading accuracy is how the ray is extended. In each increment, the ray is extended by a length equal to the distance to the next closest point in the map. This is shown in Fig. 5(c) by the dotted circles where the yellow points show each ray extension endpoint. This ensures the ray does not pass by any point during increments while also not performing unnecessarily small increments when there are no map points nearby.

3.3.2. Defect quantification and tracking

Given a labeled map, it is possible to quantify the sizes of objects or defects belonging to each class. This implementation was first introduced by our work in [27]. The approach can be summarized as follows. First, all points in the 3D map not belonging to the class of interest are removed. Next, points are separated into groups (or objects) using Euclidean distance segmentation. Clustering thresholds are manually set given the expected spacing of the defects. Given the dense maps in our approach, defect segmentation using clustering is not very sensitive to this threshold. Assuming each defect sits along a single plane (which is the case for the defects detected in this work), the total area is quantified for each object by projecting points to the best-fitting plane and calculating the convex hull, and using this to estimate the area. For a detailed explanation of this process, refer to [27]. This method makes the assumption that features are on a plane to simplify the area and length calculations. However, the defects are labeled in 3D, therefore, methods for quantification in 3D can be derived for quantifying defects on curved surfaces such as columns. We leave this exercise for future work.

Having a labeled map with defects and sizes allows for automatic and precise tracking of defect growth over time if defects can be associated between maps generated using different inspections. For a description of how lidar–camera inspection maps can be aligned and how the resulting defects can be tracked, refer to our work in [28]. Note

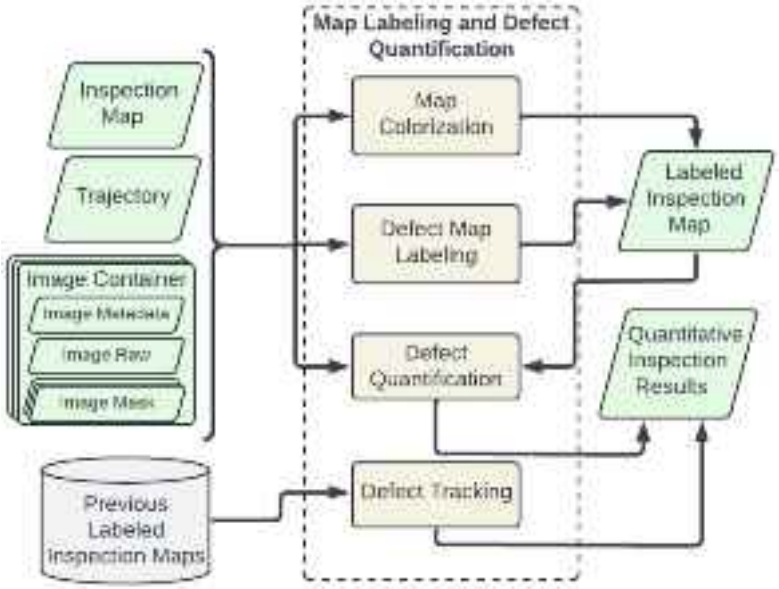
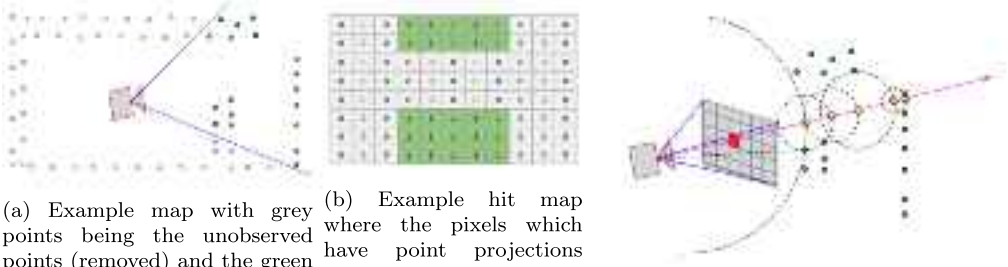


Fig. 4. Map labeling component of the inspection workflow.



(a) Example map with grey points being the unobserved points (removed) and the green points being the observed points (kept)

(b) Example hit map where the pixels which have point projections are labeled as 1 and the others are labeled with 0.

(c) Illustration of the ray-tracing methodology.

Fig. 5. Computationally efficient map labeling methodology illustration.

that the work in [28] discusses how to align maps from different inspections; however, the accuracy of defect tracking once the maps have been aligned is not assessed. Defect tracking merits comprehensive treatment and is reserved for a future study.

4. Platform-agnostic system design

To unlock robust and precise SLAM-centric inspection in a platform-agnostic formulation, we must address three issues. First is the selection of sensor specifications and configurations to support state-of-the-art SLAM systems requirements. This is an important problem that has been largely overlooked in SLAM or inspection literature, even though the results that are obtained are intricately linked to this exercise. We analyze published work in this area to determine the commonalities between best-performing hardware systems. We also discuss a selection of sensors that can be leveraged both for SLAM and inspection, while meeting the requirements for both. Second, we address the question of mobility. When discussing mobility, we will present the overall size and weight of the proposed sensor system and discuss its applicability to the common robotic platforms. Third, we discuss important additional sensors that can be added for specific inspections to improve results when applicable.

4.1. Base module sensor selection

The main idea behind our proposed platform-agnostic inspection system is to design a self-sufficient base module that performs all the core tasks required for the proposed SLAM-centric inspection approach. Our proposed system requires at least the following base sensors: 3D lidar, SLAM camera, IMU, and high-resolution camera. As discussed in Sections 1 & 2, to achieve reliable SLAM in any inspection environment, the combination of lidar, camera, and IMU data is essential. We will discuss each of these sensors individually, including their criteria, required configurations, and the specific sensor models used in this work. Table 1 shows the final sensor selection for our proposed hardware system, including the model, mass, and relevant specifications. Table 2 presents some hand-selected state-of-the-art SLAM works and their selected sensors.

First, let us discuss lidar selection. This sensor is not only critical to ensure robust SLAM, especially in mapping environments that lack sufficient visual saliency, lidar is also indispensable for the remainder of the proposed inspection pipeline, downstream from SLAM. Since we require a dense 3D map of the environment, a lidar enables consistently high density in the final map for all environmental features. This is because lidars do not require high visual texture to produce dense maps, as is the case with image-based reconstruction. We select a 3D lidar to ensure the lidar measurements for SLAM have visibility over all 6DOF of the sensor motion. A 2D lidar sensor could be used as an alternative should the sensor be mechanically rotated, such as Kim et at. [33,34] or Zhang et al. [35]. We also select a rotating lidar beam based lidar design over a solid state lidar for full 360° coverage to optimize lidar-SLAM measurements, and because the majority of lidar-SLAM literature has been centered around these types of 3D lidar

Table 1
Base module sensors.

Sensor type	Model	Mass (g)	Relevant specifications
Lidar	Velodyne VLP16 Lite	590	Vertical FOV: 30°, 300,000 points per second, 100 m range
Camera 1 (SLAM)	Flir BlackflyS	36	3.2 MP resolution, RGB, 55 frames per second
Lens 1	Fujinon	160	2.7 mm focal length, 2/3 inch image circle, C mount, FOV: 185° × 140°
Camera 2 (Inspection)	Flir BlackflyS	36	12.3 MP resolution, RGB, 23 frames per second
Lens 2	Flir Tamron	44	8 mm focal length, 1/1.8 inch image circle, C mount, FOV: 50.8° × 38.6°
IMU	Xsens MTi-30	72	Frequency: 400 Hz, Data types: xyz acceleration, xyz angular velocity, absolute heading, east-north-up orientation
Micro-controller	Teensy 3.6	20	

Table 2
Popular visual and or lidar SLAM hardware selection details.

SLAM method	Lidar	Camera rate/resolution	IMU rate
LVI-SAM [16]	Velodyne VLP16	30 Hz/0.4 MP	500 Hz
SuperOdometry [15]	Velodyne VLP16	NA/NA	NA
UMMLT LVIO [29]	Ouster OS1-64 & Velodyne VLP16	15 Hz/1.6 MP	100 Hz
VLOAM [30]	Velodyne VLP16 & Velodyne HDL32E	60 Hz/0.4 MP	400 Hz
VINS-MONO [31]	NA	30 Hz/0.4 MP & 20 Hz/0.4 MP	100 Hz & 200 Hz
LIO-SAM [32]	Velodyne VLP16	NA/NA	500 Hz

sensors. For specific 3D lidar selection, we chose the Velodyne VLP16 for its low weight and due to its ubiquitous use in lidar-SLAM literature as shown in Table 2 (e,g, [15,16,32]).

Next, we will discuss camera selection. The requirement to use cameras for SLAM-centric visual inspection is more intuitive. While vision-only SLAM systems that perform well enough in specific environments (e.g., indoors) have been demonstrated, camera-based SLAM often performs well in degenerate lidar-based SLAM environments such as environments with low structure (e.g., outdoors in open fields or roadways). On the inspection side, images are a fundamental necessity for manually or automatically retrieving inspection results, as we will see in Section 3. Specifications may be different for the two use cases, however, to keep weight low for mobility reasons discussed further in the next section, we can select camera specifications that are well suited for both cases. As shown in Table 2, image rates typically span between 15 and 60 Hz, with the average being around 20–30 Hz, which we set as a minimum for our camera requirements. In terms of resolution, LVI SLAM systems use somewhere in the range of 0.4 MP to 1.6 MP. Lens selection is another important detail that is rarely discussed in inspection or SLAM literature. A detailed study of FOV selection for optimal visual SLAM was conducted by Zhang et al. [36]. In this work, they conclude that larger FOV lenses provide increased performance in indoor environments since the high FOV allows features to be tracked for a longer time and provides more overlap in consecutive frames for more robust feature tracking. On the other hand, for outdoor scenes where features are typically at greater distances, lower FOV lenses provide better results as the increased angular resolution drastically improves feature measurements. For infrastructure inspections, we will often have a mix of feature ranges depending on the application. For example, the indoor inspection of parking garages or industrial facilities would benefit from the larger FOV lenses. Outdoor inspections such as bridges or building facades could benefit from more narrow FOV lenses.

Table 3
Example GSD and defect detection sizes.

Case	Distance to surface (m)	Vert. GSD (pixels/cm)	Hor. GSD (pixels/cm)	Min. Crack Width (mm)
Parking garage	3	14.3	14.4	2.1
Highway bridge	5	8.6	8.6	3.5

Given the need to generate high-quality inspection maps, we recognize the practical advantages of selecting two cameras: one low-resolution with a high FOV (3.2 MP and 50.8° × 38.6° FOV), and one high-resolution camera with a low FOV (12.6 MP and 185° by 140° FOV). Since image resolution can be manually downsampled prior to running SLAM, this gives us the advantages for both SLAM and inspection. For SLAM, we utilize the high FOV camera when inspecting close-range infrastructure and use the low FOV camera when inspecting longer-range infrastructure. For inspection, we can use the high FOV with a low-resolution camera to generate colorized maps of a piece of infrastructure. For defect detection, we can leverage the high-resolution camera to ensure we can reliably detect small defects such as cracks and exposed re-enforcing bars.

It is worth analyzing ground sampling density (GSD) for the selected cameras as a measure of the scale of defects we could detect. GSD is the distance between two adjacent pixel centers measured on the ground, indicating the spatial resolution of an image captured by a camera. It is calculated based on the camera's sensor size, focal length, and the distance from the ground. Given the above sensor specification for the Inspection Camera, i.e., 38.6 × 50.8 FOV with 3000 × 4095 pixel resolution, we will calculate the vertical and horizontal GSD for two cases: (1) a typical parking garage inspection where we are inspecting the soffit of the garage from a ground vehicle, and (2) a typical highway overpass bridge inspection from a ground vehicle. To be conservative, we will assume the ground vehicle's height to be 0 m above the ground. Given a minimum parking garage height in Ontario of 2.1 m, we will estimate the GSD for garages with a height of 3 m. For highway bridges, given a minimum bridge clearance of 4.15 m, we will assume a clearance of 5 m.

Table 3 summarizes the results for these two cases. The GSD for the parking garage case is 14.3 pixels/cm (vertical), whereas the GSD for the bridge case is 8.6 pixels/cm. To illustrate this with an example involving cracks, assume we need a minimum of 3 pixels per crack width for accurate detection. This means the current specifications will allow us to detect cracks that are at least 2.1 mm wide in the parking garage and 3.5 mm wide in the bridge. These requirements were deemed suitable for the inspection types targeted in this study, that is, detecting larger cracks that may need repair. However, some inspections may have more stringent requirements which may require closer access to the structure, or larger resolution cameras. Even in such cases, this approach to sensor selection can be followed to design the robotic inspection system in order to meet such application-specific needs.

For IMU selection, the main selection criteria are the data rate and the type of data that is available. Table 2 shows that state-of-the-art SLAM methods use IMUs that output data in the 100 Hz to 500 Hz. We select the Xsens MTi-30, which outputs measurements at 400 Hz and provides accelerations and rotation rates about three axes while also providing an absolute orientation estimate which can also be used for SLAM. Utilizing IMUs for integrating high-rate measurements into a SLAM system requires special treatment, we refer readers to our earlier work [23] for details.

4.2. Mobility

For mobility, we would ideally like a system that can be mounted to common robotic platforms discussed in Section 2, the most common being: UAVs, USVs, and UGVs. In addition to these mechanized

Fig. 6. Inspection platforms. Left: handheld, middle: UGV, right: USV.

mobility platforms, a hand-held platform can also be suitable for inspection applications, as a human inspector is expected to be involved in these tasks. This observation is also supported by our discussions with practicing inspectors and our own experimentation. Importantly, a handheld device reduces the overall complexity of the system, thus providing the lowest entry barrier among all mobility platforms for adoption.

Four essential sensors, which include the Velodyne Puck Lite lidar, two cameras with lenses, the IMU, and the micro-controller needed to synchronize the sensors (see Section 4.3) have a combined mass of approximately 958 g. The breakdown of mass for each sensor on our system is shown in Table 1. Of the typical mobile robotic platforms used for inspection, UAVs tend to have the smallest payloads. To serve as a reference, DJI Matrice 200 has a maximum payload of 2.34 kg (leaving 1.38 kg for computer and cables), which is a common inspection platform, and can be outfitted with the system designed here. In the form of a hand-held rig, the weight is not expected to pose significant payload limitations, with the option of carrying the non-sensor mass (e.g., computer, batteries, etc.) in a convenient form (e.g., shoulder or inside a backpack).

Fig. 6 illustrates the 3 platforms that have been tested for this work. The left image shows the handheld device which includes all the base sensors shown in Table 1, with an additional low-resolution camera with high FOV pointing backward to ensure full colorization of the lidar pointcloud. This handheld device also includes a backpack for storing the onboard computer and battery. The center image shows the UGV device. The UGV is a Husky robot from Clearpath Robotics. The sensor suite includes the same base module that is shown on the handheld device, with three additional sensors mounted upwards to capture high-quality data of the undersides of infrastructure. These sensors include: (1) a high-resolution RGB camera with low FOV, (2) Velodyne VLP-16, (3) the Flir ADK infrared camera. The figure on the far right is the USV platform, which consists of a Huron from Clearpath Robotics, with all the same sensors as the UGV platform, minus the infrared camera.

4.3. Synchronization and calibration

Precise time synchronization is critical to building a SLAM-centric inspection system. This is also an area that is seldom discussed in inspection literature, or even in SLAM literature, but is a necessary requirement for either of their success. A robot collecting data from different sensors should synchronize the data from all sensors using a single reference clock.

Fig. 7 illustrates the proposed inspection system hardware synchronization strategy. The synchronization strategy involves a microcontroller that synchronizes multi-modal data and a separate computer that stores the sensor data with pre-estimated timestamps. The Xsens IMU has the option of outputting a sync signal, which can be used to inform the receiving device when each data was captured. The microcontroller reads this signal and records each of the timestamps for the corresponding IMU. The Velodyne lidar does its own time-stamping using the controller on the sensor. To ensure the timestamps are in the same time domain as the microcontroller (and therefore all other sensor data), it can use a combination of a PPS (one pulse per second) trigger signal with corresponding timestamps in an NMEA string format. The lidar then generates its own ROS message, which is sent to the data collection computer, where all lidar points are time-stamped with a consistent clock. Lastly, the FLIR Blackfly cameras can be triggered by an external PWM (pulse with modulation) trigger signal. Since there is a time lag between when the camera receives the trigger and when the center capture of the exposure occurs, the camera returns a sync signal back to the microcontroller, which is then used to log the precise timestamp of the image. Therefore, the microcontroller has access to all timestamps associated with the cameras and IMUs, and those timestamps are packaged into ROS messages and sent to the data collection computer for storage or real-time processing. The sensor data itself always gets sent directly to the data collection computer from the sensors, and this computer finds the corresponding timestamp messages from the microcontroller.

Precise sensor synchronization is also equally as important as time synchronization. This includes both intrinsics and extrinsics calibration. Intrinsics include the parameters used to model the sensors themselves, such as focal lengths for cameras and distortion parameters. Extrinsics refers to estimating the rigid transformation between all sensors. For intrinsic calibration of the cameras and the camera-IMU extrinsic calibration, we use the open source Kalibr [37] tooling. For extrinsics calibration of other sensors, we use the camera–lidar calibration tool presented in [38], which specifically enables precise target-based calibration of non-overlapping field of view (FOV) sensors.

5. Results

This section presents experimental validation of the proposed SLAM-centric inspection framework on bridge and parking-garage datasets. The datasets, including the publicly released collection, are first described, followed by mapping results, defect detection and quantification, and quantitative evaluations of map quality, ray-tracing efficiency, and measurement accuracy.

5.1. Datasets

Two datasets are used to demonstrate the proposed SLAM-centric inspection system. In addition to publicly releasing these datasets, we also release a larger dataset with a variety of locations and platform configurations. These datasets show two different inspection applications well suited to the SLAM-centric inspection approach: (1) parking garage inspection and (2) bridge inspection. Both datasets were collected with the UGV and sensor suite discussed in Section 4.

5.1.1. Park Street Bridge

The Park Street Bridge is located at the intersection of Park Street and Cherry Street in Kitchener, Ontario, Canada. The bridge is a concrete box girder bridge with one center concrete pier and two concrete abutment walls. The bridge spans a two-lane public road with a clearance of 3.4 m, and it supports an active railway track. The bridge is in very poor structural condition, with significant concrete spalling and cracking throughout the structure. There is also significant discoloration, some paint or patching, as well as some graffiti. The

Fig. 7. Time synchronization diagram.

Fig. 8. Park Street bridge dataset images.

amount of deterioration makes this a good candidate for this research work. Furthermore, there is one sidewalk on each side of the road, making access easy for data collection. Fig. 8 shows different views of the Park Street Bridge, including some of the deterioration. For this dataset, several recordings were made where the UGV was driven around the bridge following a few different trajectories, including some with large loop closures for improved SLAM, as well as some straight line trajectories with no loops. The data was collected mid-day on a clear day, resulting in high visibility.

5.1.2. Duke Street Parking Garage

The Duke Street Parking Garage is located at the intersection of Duke Street West and Ontario Street North in Kitchener, Ontario, Canada. This is a public multi-floor parking garage with concrete slabs, girders, and columns providing all the support. At the time of inspection, there was some spalling and cracking, with some orange paint highlighting some of the spalling. The garage also had several parked vehicles at inspection time. The concrete slab is sloped in part of the garage, however, the part that was inspected which had the deterioration, was flat. Fig. 9 shows four photos of the parking garage, including some cracking on a column and spalling/cracking on the slab soffit. For this dataset, two recordings were made using the UGV as it drove straight down one of the bays, turned around, and returned. Since there were only a few significant localized defects, only that small section of the garage was mapped. This approach contrasts with the bridge dataset, where the entire bridge was covered. This dataset was also collected mid-day on a clear day for good lighting conditions.

5.2. Public dataset

The public dataset being released alongside this work was collected at 8 different locations. The dataset contains multiple examples of

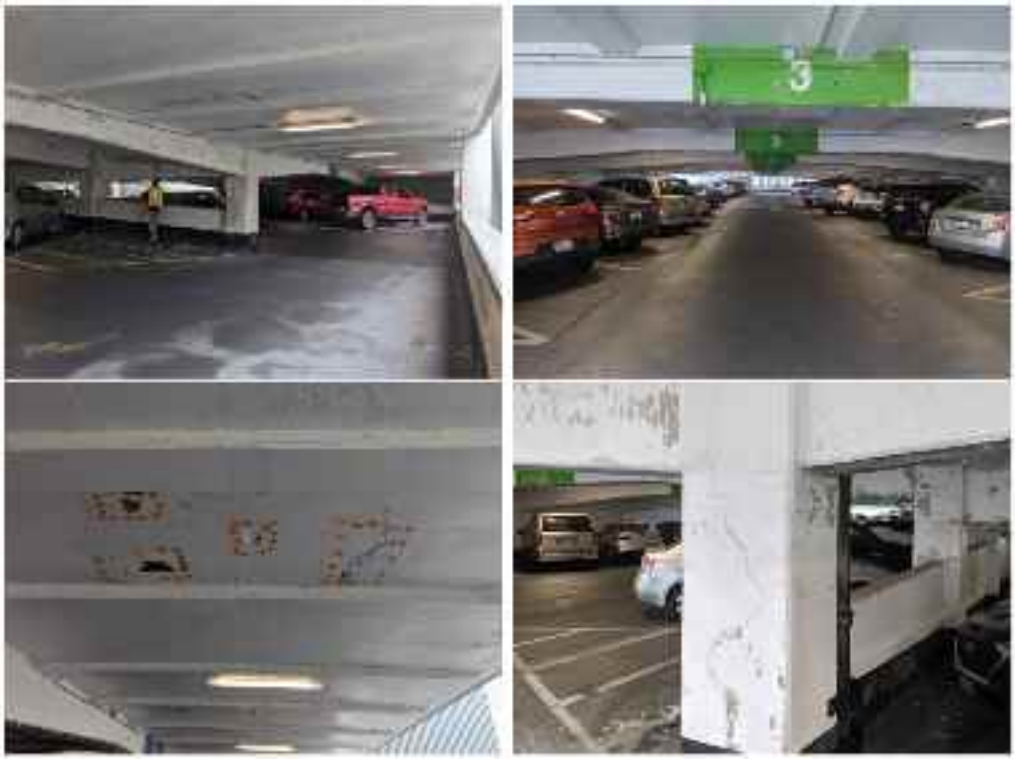


Fig. 9. Duke Street parking garage dataset images.

Table 4
Summary of the dataset locations, number of recordings (count), combined durations, combined sizes and platforms recorded on.

Location	Count	Duration (s)	Size (Gb)	Images	Platforms
Park St. bridge	4	442	57.39	11k	UGV
Conestogo bridge	5	851	89.08	21k	UGV + Handheld
Parking garage	2	339	44.51	9k	UGV
Hydrology lab	2	656	44.03	17k	UGV
Structures lab	2	559	56.99	13k	Handheld
Rod coutts hall	2	282	22.27	7k	UGV
Student design center	1	165	12.85	4k	UGV
Total	**18**	**3294**	**327.12**	**84k**	–

typical inspection environments, both indoor and outdoor. There are 4 indoor university labs, 2 bridge undersides, 1 parking garage, and 1 wide open outdoor environment. Additionally, the recordings vary by which platform they were recorded on, either being recorded using the proposed device 4.1 as a handheld unit or mounted on a UGV. Details of all recordings in the dataset are outlined in Table 4. More details of the dataset are available at https://beamrobotics.github.io/dataset.html.

5.3. Mapping results

Fig. 10 illustrates the final maps for both datasets after completing all mapping and trajectory generation steps. These are the direct output maps from the inspection map generation process.

5.4. Defect detection and quantification

Figs. 11 and 12 shows example defects extracted from the bridge dataset. In this dataset, SAM was used to semi-automatically select defect areas to be extracted and quantified. This is a good example of an inspection where it is beneficial for labeling to be done with a human in the loop. Since the bridge is in poor condition, explicit defects are not obvious and will be left to an inspector and the owner to decide what exactly needs to be repaired. In this dataset, SAM is used with a few manual positive and negative labels input by the inspector. This is used to segment spall areas, as well as a repaired section which is being called a delamination. In Figs. 11 and 12, the left column shows the raw image, and the right shows the raw image with the defect labeled in red, which will then be used in the defect map labeling and quantification discussed next.

Figs. 14, 15 and 16 show the results after the map labeling process on the defects from the bridge and garage datasets, respectively. These defects correspond to the same ones shown in Figs. 14, 13 and 15 using the same inspection maps shown above. The left column on these figures shows the map labeled using the RGB images, and the right column shows the defect maps labeled on the map. In the left column, only the points that are seen by the camera at that image time are labeled, and all other points are left black (note that these can later be combined to color the entire map). The right column shows the points labeled as defects in red, and all other points are in blue. In the bottom row of Fig. 16, the remainder of the map is also shown in gray to get a perspective as to where that section of the map is on the entire structure.

Fig. 17 shows the segmented defects for the bridge dataset, while Fig. 18 shows these defects situated on the entire pointcloud map. Having the ability to display defects on a complete 3D map can be helpful for inspectors to visualize and get a perspective of all defects, including their sizes and locations relative to each other. Table 5 presents all defect quantification results for the bridge dataset. In this dataset, all defects are marked as either spalls or delaminations. Quantification

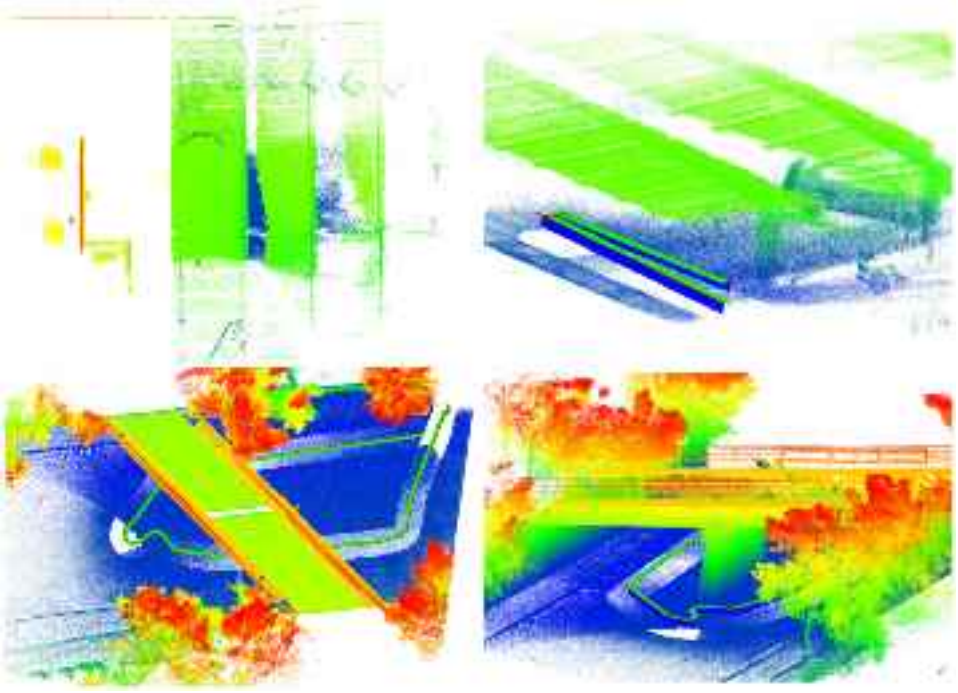


Fig. 10. Lidar inspection map generation results using the vertical lidar. Top row: parking garage dataset, bottom row: bridge dataset, left column: plan view, right column: oblique view.

Fig. 11. Bridge dataset example defect labels from the front-facing high field-of-view camera. Image IDs from top to bottom: C1I10, C1I26. The left side shows the raw images, and the right shows the image with the defect labeled in red.

metrics in Table 5 include the concave hull area and 2D bounding box dimensions (width, height, area), which can be useful to try to estimate overall patch size at repair time.

Fig. 19 and Table 6 below show the same information as above but for the parking garage. For this dataset, the defects labeled were cracks. For crack quantification, the bounding box area is also quantified, but crack length is also crudely estimated by simply taking the largest distance between any two points in the defect cloud. For more analysis on quantification accuracy for such an approach, refer to our previous work in [39].

Table 5
Defect quantification summary for bridge dataset.

Image label	Defect type	Hull area (m^2)	BBox width (m)	BBox length (m)	BBox area (m^2)	Color
C1110	Spall	1.95	1.39	2.45	3.40	Red
C1126	Spall	0.39	0.73	0.99	0.72	Yellow
C2120	Delam	2.61	1.35	2.26	3.05	Green
C3127	Spall	7.91	2.51	6.88	17.26	Blue

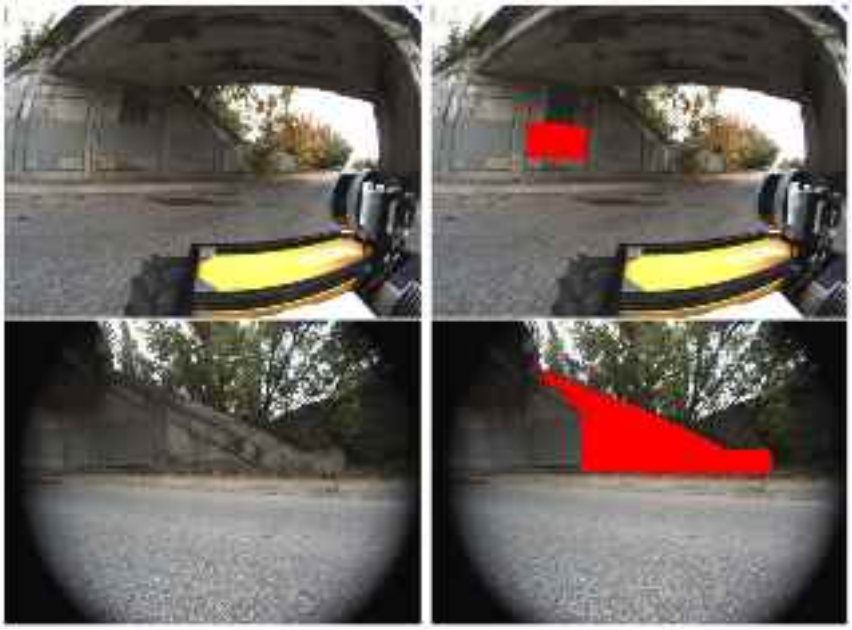


Fig. 12. Bridge dataset example defect labels from the rear-facing camera (top) and the front-facing high-resolution camera (bottom). IDs: C2I20 top, C3I27 bottom. The left side shows the raw images, and the right shows the defects overlayed in red.

Fig. 13. Garage dataset example defect labels. Top: F1I2, bottom: F4I4.

Table 6
Defect quantification summary for garage dataset.

Image label	Defect type	Length (m)	BBox width (m)	BBox length (m)	BBox area (m^2)	Color
C112	Crack	1.39	0.32	1.35	0.43	Red
C112	Crack	1.4	0.33	1.36	0.44	Green
C414	Crack	0.65	0.41	0.5	0.20	Green
C414	Crack	0.38	0.22	0.3	0.06	Yellow
C414	Crack	0.26	0.09	0.24	0.02	Blue
C414	Crack	0.37	0.09	0.36	0.03	Purple
C414	Crack	0.19	0.08	0.17	0.01	Brown

5.5. Quantitative evaluation

This section presents some quantitative evaluation methods used to validate three main contributions of this work. We first evaluate the map quality which essentially validates the end-to-end approach for map generation, including online SLAM, offline refinement, and the decoupled mapping. Since our datasets were collected in uncontrolled environments without proper ground truthing, we could not perform more standard localization and mapping metrics such as drift or absolute pose error. Instead, we trust that the end-to-end quantitative errors presented herein adequately capture such errors. Second, we evaluate the speed-up in the proposed efficient ray-tracing methodology. Finally, we evaluate overall measurement accuracy by taking point measurements in the final map and comparing to on-site measurements.

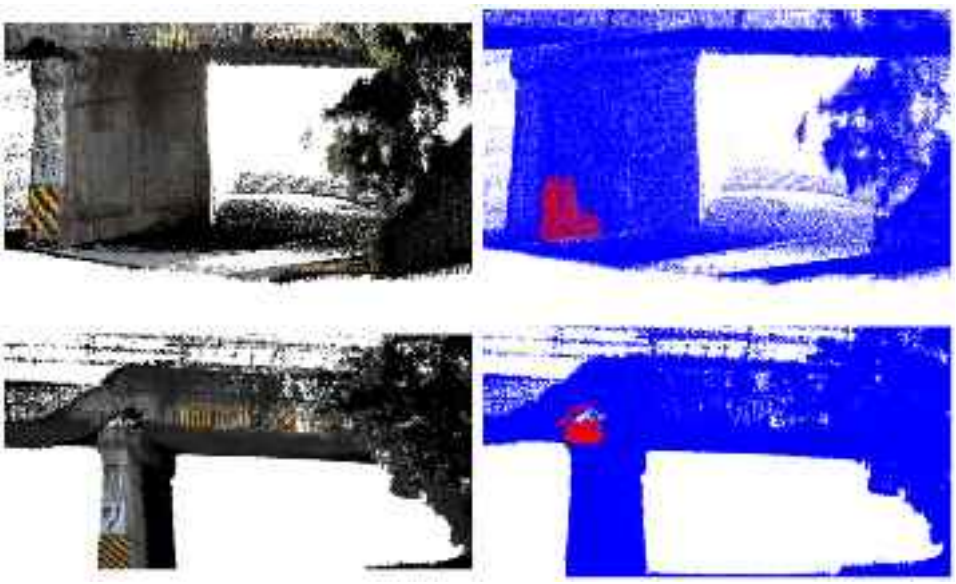


Fig. 14. Bridge map labeling results from the front-facing high field-of-view camera. Image IDs from top to bottom: C1I10, C1I26. The left column shows the map colorized with the RGB camera, and the right column shows the map colorized with the defects.

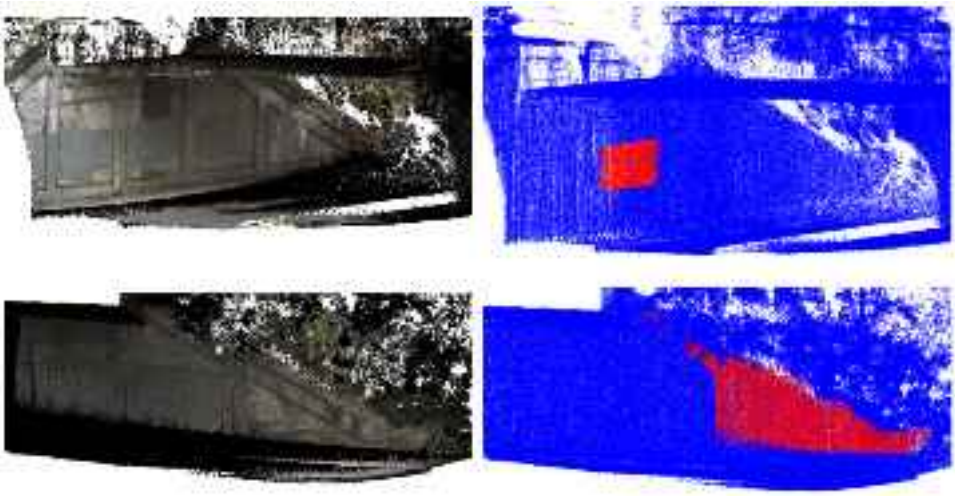


Fig. 15. Bridge map labeling results from the rear-facing camera (top) and the front-facing high-resolution camera (bottom). Image IDs from top to bottom: C2I20, C3I27. The left column shows the map colorized with the RGB camera, and the right column shows the map colorized with the defects.

5.5.1. Map evaluation

To support the development of high-quality inspection maps, a reliable evaluation metric is needed—one that reflects structural accuracy rather than trajectory precision. Traditional SLAM metrics, such as trajectory error or optimization residuals, are inadequate for this purpose and often rely on ground truth data, which is impractical to obtain in real-world inspection scenarios. Existing map quality metrics typically assume access to reference maps, limiting their applicability. This work proposes a new, self-contained metric that evaluates map quality directly from the pointcloud, enabling practical assessment and parameter tuning without the need for external references.

This section introduces a point-to-plane distance metric designed to reflect how humans qualitatively assess pointcloud map quality—by inspecting surface clarity and consistency. Rather than evaluating the entire map, the approach focuses on the most reliable geometric structures: large planar surfaces, which are common in inspection datasets and can be robustly identified. The method extracts the top horizontal and vertical planes using geometric and clustering constraints, filters outliers, and then computes the average distance of inlier points to their respective planes. Fig. 20 shows the plan extraction results for the parking garage dataset. Calculating the "surface thickness" of these planes effectively captures the noise and drift present in the map, providing a quantitative proxy for visual quality—cleaner, thinner surfaces correspond to higher map accuracy.

As additional evidence to our map quality metric, we also provide *Density* metrics from the popular open-source CloudCompare [40] software. The *Density* metric is computed by iterating over all points and calculating the number of neighbors found within a fixed search radius and dividing by the area to approximate a surface density. The idea behind this metric is that higher quality maps should be denser as points are more concentrated along the surfaces.

Table 7 shows the results of running the proposed map quality metrics on both datasets presented in this work. We run these metrics using three different versions of the inspection map. The first has no refinement (online SLAM only), the second has light refinement, which only performs light loop closure, and the third is the full refinement using the strategy discussed above. In all cases, we can see that the map quality metrics increase substantially with the proposed map refinement methodology.

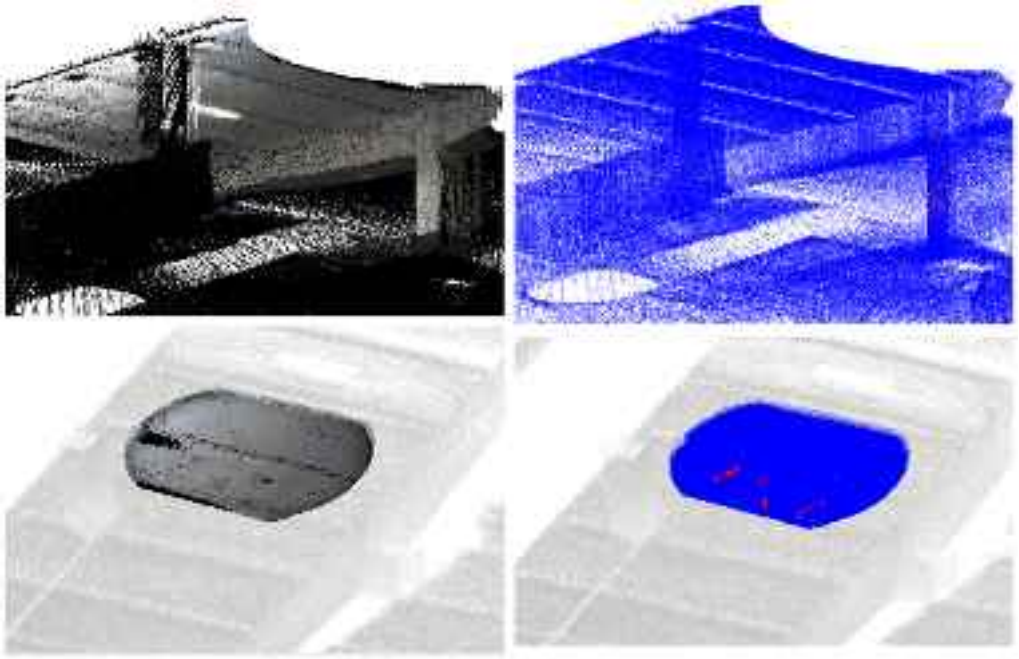


Fig. 16. Garage map labeling results. Image IDs from top to bottom: F1I2, F4I4.

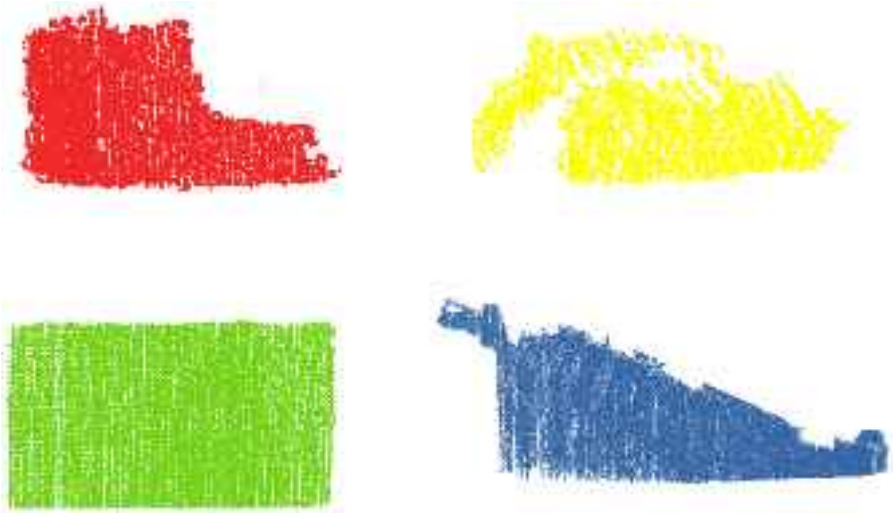


Fig. 17. Bridge dataset segmented defects from each image shown in Fig. 13. See Table 5 for color labels.

Fig. 18. Visualization of all bridge defect labels on the map using different colors.

Fig. 19. Garage dataset segmented defects from each image shown in Fig. 13. See Table 6 for color labels.

Table 7
Comparison of point-cloud map quality metrics for bridge and parking-garage datasets.

Dataset	Refinement level	Point-to-plane (cm) (proposed)	Density ($\frac{pt}{cm^2}$) (CloudCompare)
Garage	None	4.164	0.493
	light	3.904	0.512
	max	3.647	0.543
Bridge	None	6.175	0.301
	light	5.183	0.367
	max	4.231	0.396

Table 8
Ablation study results on ray-tracing computation time.

Camera	Cropped map size	Time (s)				
		Ours	No map crop	No hit map	Fixed ray Ex	No speed ups
SLAM	539k	4.7	6.2	22	192	1819
Insp. Fwd	227k	3.1	3.6	88	98	5641
Insp. Frnt	104k	0.4	1.8	48	99	2815

5.5.2. Ray-tracing ablation study

The main advantage of the proposed ray-tracing methodology is that it achieves the same level of precision as standard image-to-pointcloud ray-tracing while being significantly more computationally efficient. This is possible because all final pixels are rendered directly onto the unaltered map using standard ray-tracing, without any meshing or restructuring. Accordingly, this section focuses on demonstrating the improvements in computational efficiency via an ablation study.

Table 8 shows the results of the ablation study on the proposed efficiency improvements to our ray-tracing. In this table, no map crop means that the lidar map was not cropped to only the points projecting into the image plane. No hit map means we ray-trace all pixels in the image, fixed ray extension means that we extend the ray each iteration by a fixed size equal to the hit threshold, and no speed ups means none of the improvements we propose to the ray-tracing are applied. We take 3 images in total, all with the same timestamp near the start of the parking garage dataset. The first is an image from the SLAM camera, which contains many points and both close and far distances due to its high FOV lens. The second image is a high-resolution forward-facing inspection camera containing mostly points at a distance, but fewer points than the SLAM camera due to the narrow FOV. The third image is the high-resolution upwards-pointed inspection camera viewing the soffit which is in close proximity to the camera. We demonstrate that turning off each of the three improvements on its own has a significant slowdown on computation time, and also show the significant computation time when all improvements are turned off at once. The tests were performed on a ThinkPad laptop with a 12th Gen Intel Core i7-12700Hx20 processor with 16 GB of RAM. When comparing our method to the standard ray-tracing with no speed ups, we achieve an average speed increase of 9000x.

Table 9
Comparison of on-site measurements versus measurements of the same features from the colored map.

#	Site (m)	Map (m)	Abs error (m)	Error (%)
1	2.26	2.28	0.02	0.9
2	1.35	1.32	0.03	2.2
3	2.41	2.68	0.08	3.3
4	0.75	0.77	0.02	2.6
5	0.75	0.76	0.01	1.3
6	0.65	0.67	0.02	3.0
7	0.51	0.54	0.03	5.4

5.5.3. Measurement validation

As with many real-world defects, the defects measured in this study are fairly subjective as to their boundaries and hence can be ambiguous. Such boundaries are often based on inspector's opinion or based on the repair plan. Hence, in this study, a more objective approach is taken to validate the accuracy of our defect measurements; we measure distinct and unambiguous features on the infrastructure to serve as a proxy to defect measurement accuracy. These proxy-measurements should properly represent the measurement error in the presented system, however, true in-field measurements of defects may be higher. Table 9 presents the results from these tests. The second column shows the length measurement captured during the on-site visit, whereas the third column shows the result for that same feature measured using the colored map generated with our proposed approach. Fig. 21 illustrates all the measurements taken on the physical structure. We took a total of 7 measurements; Measurements 1 to 5 were taken on the Bridge dataset, whereas measurements 6 and 7 were taken on the parking garage dataset. Measurement 1 is the width of the rectangular section on the Southwest face of the middle abutment wall. Measurements 2 and 3 are the height and width of the defect shown in green above, respectively. Measurements 4 and 5 are the heights of the yellow and white signs on the East end of the middle abutment, respectively. Measurement 6 is the width of the column that contains the crack shown above, and measurement 7 is the height of the black paint on that same column.

An average of error of 3 cm or 2.7% was obtained over all 7 validation measurements, with a standard deviation of 2.3 cm. Note that the automated defect measurement shown in Table 5 also had similar dimensions to the results shown in Table 9 at measurements 4 and 5, with no error in the measured height and 6% error in the measured width.

6. Conclusions and limitations

This paper presented an end-to-end visual inspection task using a SLAM-centric approach. Physical systems are implemented to achieve effective SLAM and inspection data collection, while providing the versatility needed for common infrastructure inspection tasks. The proposed approach demonstrates a reliable method for generating precise trajectories and 3D maps specific for inspection, which can be integrated with ML models and images to extract, quantify, and track defect changes over time. The methodology is validated in real-world scenarios, including inspection of a concrete bridge and a concrete parking garage. Due to its versatile design, the framework can be applied to additional inspection tasks such as tunnels, culverts, and building facades.

The major limitations and recommended areas for future study are discussed next. First, reliable SLAM remains a challenge despite proper system design and sensor integration/synchronization. In practice, designing the software to enable such a system still remains complex, and there is a lack of open-source resources to do so. This remains a high engineering burden that is often too onerous for typical academic research labs. Second, the 3D colored maps produced by overlaying images and 3D pointclouds remain far from visually realistic renderings. Clearly, a

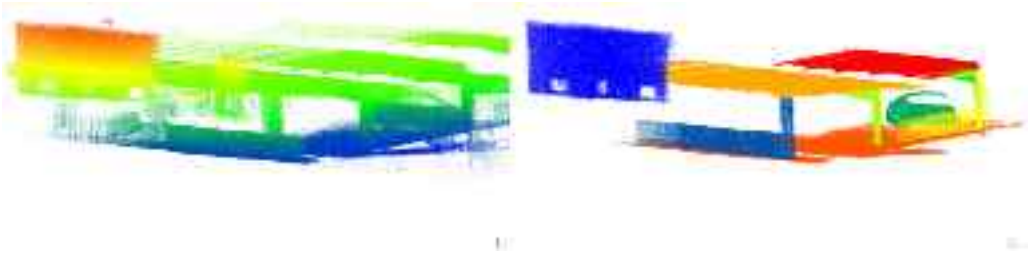


Fig. 20. Parking garage dataset plane extraction results. Left: raw pointcloud, right: final planes.

Fig. 21. Images of each of the 7 locations used to validate defect measurements precision.

visually realistic digital twin of the asset would be very beneficial for inspectors. Recent advancements in neural rendering techniques such as Nerf [41] and Gaussian Splatting [42] show tremendous potential to solve this problem. These solutions are also capable of using the data provided by the inspection system presented in this paper (images and pointclouds), and therefore the exploration of this space is highly recommended for future work. In terms of datasets made available publicly, the lack of ground truths for the defects and trajectories is a limitation. Despite this limitation, we think that these extensive datasets taken together with the baseline results provided here from state-of-the-art SLAM constitutes an important body of scientific data and results for researchers to develop their own systems for inspections.

This research addresses pressing societal needs by reducing traditional inspection methods' cost, time, and variability. The proposed SLAM-centric framework empowers infrastructure managers with a quantitative, repeatable, and scalable approach to asset assessment, enabling improved resource allocation and prioritization of maintenance tasks. The ability to track changes in defects over time fosters a proactive rather than reactive maintenance culture, ultimately enhancing the safety and longevity of critical infrastructure components like bridges and parking garages.

CRediT authorship contribution statement

Nicholas Charron: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Jake McLaughlin:** Writing – review & editing, Software, Methodology, Formal analysis, Data curation. **Sriram Narasimhan:** Writing – review & editing, Supervision, Resources, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We would like to thank Dr. Chul Min Yeum for his invaluable assistance in collecting measurements for our validation section. We also gratefully acknowledge Gabe Earle, Alex Thoms and Dirk Friesen for their support in integrating sensors for the robotic platforms presented herein, and for help collecting data released as part of this paper.

Data availability

All datasets and source code used in this paper are publicly available. The inspection software is released as open-source at https://github.com/nickcharron/beam_inspection, and the associated multimodal inspection datasets can be accessed at https://beamrobotics.github.io/dataset.html.

References

[1] H. Ahmed, H.M. La, N. Gucunski, Review of non-destructive civil infrastructure evaluation for bridges: State-of-the-art robotic platforms, sensors and algorithms, Sensors 20 (14) (2020) 3954, http://dx.doi.org/10.3390/s20143954.

[2] M. Trzeciak, K. Pluta, Y. Fathy, L. Alcalde, S. Chee, A. Bromley, I. Brilakis, P. Alliez, Conslam: Periodically collected real-world construction dataset for SLAM and progress monitoring, in: European Conference on Computer Vision, Springer, 2022, pp. 317–331, http://dx.doi.org/10.1007/978-3-031-25082-8_21.

[3] A.D. Nair, J. Kindle, P. Levchev, D. Scaramuzza, Hilti SLAM challenge 2023: Benchmarking single+ multi-session SLAM across sensor constellations in construction, IEEE Robot. Autom. Lett. 9 (8) (2024) 7286–7293, http://dx.doi.org/10.1109/LRA.2024.3421791.

[4] L. Lu, F. Dai, Accurate road user localization in aerial images captured by unmanned aerial vehicles, Autom. Constr. 158 (2024) 105257, http://dx.doi.org/10.1016/j.autcon.2023.105257.

[5] M. Kopsida, I. Brilakis, P. Vela, A review of automated construction progress monitoring methods, in: Proceedings of the 32nd CIB W78 Conference on Construction IT, Apollo - University of Cambridge Repository, 2023, http://dx.doi.org/10.17863/CAM.92941, URL: https://www.repository.cam.ac.uk/handle/1810/345518.

[6] R.S. Lim, H.M. La, W. Sheng, A robotic crack inspection and mapping system for bridge deck maintenance, IEEE Trans. Autom. Sci. Eng. 11 (2) (2014) 367–378, http://dx.doi.org/10.1109/TASE.2013.2294687.

[7] H.M. La, R.S. Lim, B.B. Basily, N. Gucunski, J.G. Yi, A. Maher, F.A. Romero, H. Parvardeh, Mechatronic systems design for an autonomous robotic system for high-efficiency bridge deck inspection and evaluation, IEEE-ASME Trans. Mechatron. (ISSN: 1083-4435) 18 (6) (2013) 1655–1664, http://dx.doi.org/10.1109/Tmech.2013.2279751,

[8] H.M. La, N. Gucunski, S.-H. Kee, J. Yi, T. Senlet, L. Nguyen, Autonomous robotic system for bridge deck data collection and analysis, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 1950–1955, http://dx.doi.org/10.1109/IROS.2014.6942821.

[9] S. Gibb, H.M. La, T. Le, L. Nguyen, R. Schmid, H. Pham, Nondestructive evaluation sensor fusion with autonomous robotic system for civil infrastructure inspection, J. Field Robot. 35 (6) (2018) 988–1004, http://dx.doi.org/10.1002/rob.21791.

[10] H.A. Peel, Robotic Navigation and Inspection of Bridge Bearings (Ph.D. thesis), University of Leeds, Woodhouse, Leeds LS2 9JT, UK, 2019.

[11] L. Yang, B. Li, W. Li, H. Brand, B. Jiang, J. Xiao, Concrete defects inspection and 3D mapping using CityFlyer quadrotor robot, IEEE/CAA J. Autom. Sin. 7 (4) (2020) 991–1002, http://dx.doi.org/10.1109/JAS.2020.1003234.

[12] C.-Q. Feng, B.-L. Li, Y.-F. Liu, F. Zhang, Y. Yue, J.-S. Fan, Crack assessment using multi-sensor fusion simultaneous localization and mapping (SLAM) and image super-resolution for bridge inspection, Autom. Constr. 155 (2023) 105047, http://dx.doi.org/10.1016/j.autcon.2023.105047.

[13] J. Lin, F. Zhang, R3LIVE: A robust, real-time, RGB-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package, in: 2022 International Conference on Robotics and Automation, ICRA, IEEE, 2022, pp. 10672–10678, http://dx.doi.org/10.1109/ICRA46639.2022.9811935.

[14] L. Ge, A. Sadhu, Deep learning-enhanced smart ground robotic system for automated structural damage inspection and mapping, Autom. Constr. 170 (2025) 105951, http://dx.doi.org/10.1016/j.autcon.2024.105951.

[15] S. Zhao, H. Zhang, P. Wang, L. Nogueira, S. Scherer, Super odometry: IMU-centric lidar-visual-inertial estimator for challenging environments, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2021, pp. 8729–8736, http://dx.doi.org/10.1109/IROS51168.2021.9635862.

[16] T. Shan, B. Englot, C. Ratti, D. Rus, Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 5692–5698, http://dx.doi.org/10.1109/ICRA48506.2021.9561996.

[17] S. Jung, S. Song, S. Kim, J. Park, J. Her, K. Roh, H. Myung, Toward autonomous bridge inspection: A framework and experimental results, in: 2019 16th International Conference on Ubiquitous Robots, UR, IEEE, 2019, pp. 208–211, http://dx.doi.org/10.1109/URAI.2019.8768677.

[18] S. Jung, D. Choi, S. Song, H. Myung, Bridge inspection using unmanned aerial vehicle based on HG-SLAM: Hierarchical graph-based SLAM, Remote. Sens. 12 (18) (2020) 3022, http://dx.doi.org/10.3390/rs12183022.

[19] Z. Ameli, Y. Aremanda, W.A. Friess, E.N. Landis, Impact of UAV hardware options on bridge inspection mission capabilities, Drones 6 (3) (2022) 64, http://dx.doi.org/10.3390/drones6030064.

[20] D. Meister, S. Ogaki, C. Benthin, M.J. Doyle, M. Guthe, J. Bittner, A survey on bounding volume hierarchies for ray tracing, in: Computer Graphics Forum, vol. 40, Wiley Online Library, 2021, pp. 683–712, http://dx.doi.org/10.1111/cgf.142662.

[21] Y. Yan, Z. Mao, J. Wu, T. Padir, J.F. Hajjar, Towards automated detection and quantification of concrete cracks using integrated images and lidar data from unmanned aerial vehicles, Struct. Control. Health Monit. 28 (8) (2021) e2757, http://dx.doi.org/10.1002/stc.2757.

[22] G. Kim, S. Choi, A. Kim, Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments, IEEE Trans. Robot. 38 (3) (2021) 1856–1874, http://dx.doi.org/10.1109/TRO.2021.3116424.

[23] A. Thoms, G. Earle, N. Charron, S. Narasimhan, Tightly coupled, graph-based DVL/IMU fusion and decoupled mapping for SLAM-centric maritime infrastructure inspection, IEEE J. Ocean. Eng. (2023) 663–676, http://dx.doi.org/10.1109/JOE.2023.3265742.

[24] C. Zhang, C.-c. Chang, M. Jamshidi, Simultaneous pixel-level concrete defect detection and grouping using a fully convolutional model, Struct. Health Monit. 20 (4) (2021) 2199–2215, http://dx.doi.org/10.1177/1475921720985437.

[25] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, 2018, http://dx.doi.org/10.48550/arXiv.1804.02767, arXiv preprint arXiv:1804.02767.

[26] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A.C. Berg, W.-Y. Lo, P. Dollar, R. Girshick, Segment anything, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 4015–4026, http://dx.doi.org/10.48550/arXiv.2304.02643.

[27] E. McLaughlin, N. Charron, S. Narasimhan, Automated defect quantification in concrete bridges using robotics and deep learning, J. Comput. Civ. Eng. 34 (5) (2020) 04020029, http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0000915.

[28] J. McLaughlin, N. Charron, S. Narasimham, Visual-lidar map alignment for infrastructure inspections, 2024, http://dx.doi.org/10.48550/arXiv.2501.14486, ArXiv Preprint.

[29] D. Wisth, M. Camurri, S. Das, M. Fallon, Unified multi-modal landmark tracking for tightly coupled lidar-visual-inertial odometry, IEEE Robot. Autom. Lett. 6 (2) (2021) 1004–1011, http://dx.doi.org/10.1109/LRA.2021.3056380.

[30] J. Zhang, S. Singh, Visual-lidar odometry and mapping: Low-drift, robust, and fast, IEEE Int. Conf. Robot. Autom. (2015) 2174–2181, http://dx.doi.org/10.1109/ICRA.2015.7139486,

[31] T. Qin, P. Li, S. Shen, Vins-mono: A robust and versatile monocular visual-inertial state estimator, IEEE Trans. Robot. 34 (4) (2018) 1004–1020, http://dx.doi.org/10.1109/TRO.2018.2853729.

[32] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, D. Rus, Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 5135–5142, http://dx.doi.org/10.1109/IROS45743.2020.9341176.

[33] P. Kim, J. Chen, Y.K. Cho, Robotic sensing and object recognition from thermal-mapped point clouds, Int. J. Intell. Robot. Appl. 1 (3) (2017) 243–254, http://dx.doi.org/10.1007/s41315-017-0023-9.

[34] C. Wang, C. Harper, Y. Lee, R. Harris, C. Berryman, Autonomous mobile robot localization and mapping for unknown construction environments, in: Construction Research Congress, ISBN: 9780784481264, 2018, pp. 491–500, http://dx.doi.org/10.1061/9780784481264.015, URL: https://ascelibrary.org/doi/pdf/10.1061/9780784481301.

[35] J. Zhang, S. Singh, et al., LOAM: Lidar odometry and mapping in real-time, in: Robotics: Science and Systems, vol. 2, Berkeley, CA, 2014, pp. 1–9, http://dx.doi.org/10.1007/s10514-016-9548-2.

[36] Z. Zhang, H. Rebecq, C. Forster, D. Scaramuzza, Benefit of large field-of-view cameras for visual odometry, Proc. - IEEE Int. Conf. Robot. Autom. (ISSN: 10504729) 2016-June (2016) 801–808, http://dx.doi.org/10.1109/ICRA.2016.7487210,

[37] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, R. Siegwart, Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes, in: 2016 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2016, pp. 4304–4311, http://dx.doi.org/10.1109/ICRA.2016.7487628.

[38] N. Charron, H. Weng, S.L. Waslander, S. Narasimhan, A target-based extrinsic calibration framework for non-overlapping camera-lidar systems using a motion capture system, Proc. Conf. Robot. Vis. (2025) http://dx.doi.org/10.21428/d82e957c.58a61852, https://crv.pubpub.org/pub/jd9ql2lj.

[39] N. Charron, E. McLaughlin, S. Phillips, K. Goorts, S. Narasimhan, S.L. Waslander, Automated bridge inspection using mobile ground robotics, J. Struct. Eng. 145 (11) (2019) 04019137, http://dx.doi.org/10.1061/(ASCE)ST.1943-541X.0002404.

[40] CloudCompare, (Version 2.11.1) [GPL Software], Technical Report, GPL software, 2016, URL: http://www.cloudcompare.org/.

[41] B. Mildenhall, P.P. Srinivasan, M. Tancik, J.T. Barron, R. Ramamoorthi, R. Ng, Nerf: Representing scenes as neural radiance fields for view synthesis, Commun. ACM 65 (1) (2021) 99–106, http://dx.doi.org/10.1145/3503250.

[42] B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, 3D Gaussian splatting for real-time radiance field rendering, ACM Trans. Graph. 42 (4) (2023) 139, http://dx.doi.org/10.48550/arXiv.2308.04079, URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.