TRANSPORTATION POOLED FUND PROGRAM
QUARTERLY PROGRESS REPORT

Lead Agency (FHWA or State DOT): Kansas DOT

INSTRUCTIONS:

Lead Agency contacts should complete a quarterly progress report for each calendar quarter during which the projects are active. Please provide a project schedule status of the research activities tied to each task that is defined in the proposal; a percentage completion of each task; a concise discussion (2 or 3 sentences) of the current status, including accomplishments and problems encountered, if any. List all tasks, even if no work was done during this period.

Transportation Pooled Fund Program Project # *(i.e, SPR-2(XXX), SPR-3(XXX) or TPF-5(XXX)* **TPF-5(535)**	**Transportation Pooled Fund Program - Report Period:** ☐Quarter 1 (January 1 – March 31) ☒Quarter 2 (April 1 – June 30) ☐Quarter 3 (July 1 – September 30) ☐Quarter 4 (October 1 – December 31)
TPF Study Number and Title: **TPF-5(535): Human-centered Steel Bridge Inspection enabled by Augmented Reality and Artificial Intelligence**	

Lead Agency Contact: **David Behzadpour**	**Lead Agency Phone Number:** **785-291-3847**	**Lead Agency E-Mail** David.Behzadpour@ks.gov
Lead Agency Project ID: Click or tap here to enter text.	**Other Project ID (i.e., contract #):** Click or tap here to enter text.	**Project Start Date:** 10/1/2024
Original Project Start Date: **10/1/2024**	**Original Project End Date:** **9/30/2027**	**If Extension has been requested, updated project End Date:** Click or tap to enter a date.

Project schedule status:

☐On schedule	☐On revised schedule	☐Ahead of schedule	☒ Behind schedule

Overall Project Statistics:

Total Project Budget	Total Funds Expended This Quarter	Percentage of Work Completed to Date
$600,000	$23,898	42%

Project Description:

The main objective of this proposed research is to provide state DOTs practical tools for supporting human-centered steel bridge inspection with real-time defect (e.g., fatigue cracks and corrosion) detection, documentation, tracking, and decision making. The proposed research will not only bridge the gaps identified in the IDEA project, but also expand the existing capability by developing AI algorithms for crack and corrosion detection. In addition to AR headsets, the project will also develop AR-based inspection capability using tablet devices. The tablet device can be used to perform AR-based inspection directly in a similar way to the AR headset. It can also leverage Unmanned Aerial Vehicles (UAV) for remote image and video acquisition during inspections, enabling bridge inspections from a distance in a human-centered manner.

Progress this Quarter
(includes meetings, work plan status, contract status, significant progress, etc.):

1. **Task 1: CV and AI algorithms for crack and corrosion inspection**

The proposed MetaCNN-based stacking ensemble achieved the highest overall segmentation performance, with a Global Damage IoU of 92.5%, outperforming all individual YOLOv8 base learners.

The MetaCNN improved class-wise segmentation performance, achieving the highest IoU for Fair corrosion (82.8%) and Poor corrosion (44.4%), showing its ability to better fuse complementary predictions from multiple YOLOv8 models.

2. **Task 2: AR-based software for human-centered bridge inspection**

Subtask 2.3: AR software environment for AR headset

In this quarter, spatial meshing performance within the Magic Leap 2 headset application was significantly improved by integrating an updated spatial meshing implementation provided by Magic Leap developer resources. The improved mesh was tested in the field, and more accurately aligned with the physical bridge infrastructure, resulting in improved projection accuracy and more reliable spatial anchoring during field demonstrations.

Subtask 2.4: AR software environment for tablet device and UAV

A real-time RGB-D SLAM system has been developed and evaluated using the Intel RealSense D455 depth camera deployed on a NVIDIA Jetson Orin NX embedded platform. The system integrates depth and color streams to simultaneously estimate camera pose and reconstruct a dense 3D map of the environment. Running on the Jetson Orin NX's ARM-based CPU, the pipeline achieves an average processing rate of approximately 4 frames per second, demonstrating the feasibility of onboard real-time 3D mapping on resource-constrained edge hardware.

Anticipated work next quarter:

1. **Task 1: CV and AI algorithms for crack and corrosion inspection**
   - Investigate alternative base learners and fusion strategies, including U-Net, DeepLabV3+, SegFormer, Mask2Former, transformer-based models, and attention-free or attention-based MetaCNN fusion
   - Explore advanced learning strategies such as curriculum learning, semi-supervised learning, active learning, zero-shot learning, and self-supervised learning to reduce dependence on extensive manual annotations.
   - Use self-supervised and foundation-models for pseudo-labeling and label refinement, including DINO/DINOv2 feature clustering and SAM-assisted boundary refinement.

- Investigate generative models such as Pix2Pix or GAN-based refinement methods to improve initial masks, enhance corrosion boundaries, and reduce uncertainty in ambiguous severity regions.

Task 2: AR-based software for human-centered bridge inspection

- Continue to validate AR headset inspection application in the field
- Implement new database features for inspection storage, allow inspectors to add notes to inspections
- Continue to improve the performance of the real SLAM algorithm by using GPU
- Display acquired images and generated models using Unity.

Significant Results:

1. **Task 1: CV and AI algorithms for crack and corrosion inspection**

   The proposed methodology aims to develop a robust and reliable framework for automated corrosion segmentation in steel bridges using a stacking-based meta-ensemble learning strategy. The framework integrates multiple YOLOv8 segmentation models and applies pixel-level probability fusion to overcome the limitations of individual model predictions commonly encountered in vision-based structural health monitoring applications. As illustrated in Figure 1, the proposed framework consists of two main stages. In the first stage, multiple YOLOv8 segmentation models operate in parallel to generate independent class-wise probability maps for each input image. In the second stage, these probability maps are concatenated along the channel dimension and passed to a convolutional meta-learner, referred to as MetaCNN. The MetaCNN learns nonlinear spatial relationships among the base-model outputs and produces a refined final corrosion segmentation map. By combining complementary information from different base learners, the proposed stacking strategy improves prediction consistency, reduces the influence of model-specific errors, and enhances the robustness of corrosion segmentation across varying surface conditions and severity levels.

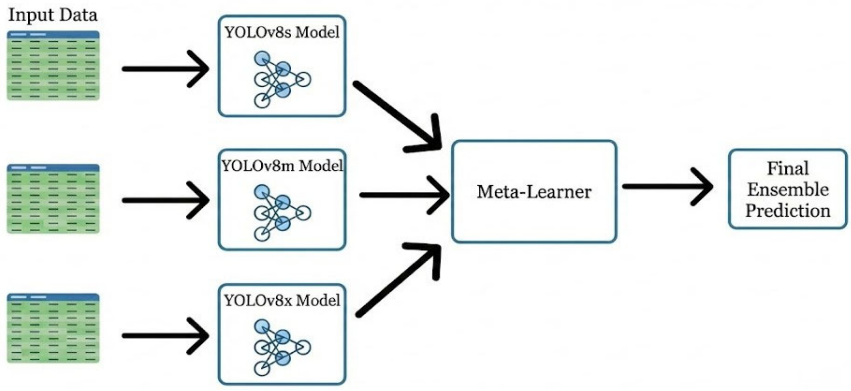


Figure 1. Stacking Ensemble Learning Framework

The dataset used in this study is steel bridge corrosion dataset obtained from the Kansas Department of Transportation (KDOT) and the Texas Department of Transportation (TxDOT). It consists of 639 labeled images, including 379 images from TxDOT and 260 images from KDOT. Corrosion damage is categorized into three severity classes based on visual characteristics and structural implications. Class 1 (Fair) represents mild surface corrosion or discoloration, Class 2 (Poor) corresponds to deeper corrosion with visible disintegration or pack rust, and Class 3 (Severe) denotes advanced corrosion resulting in section loss, holes, or direct impacts on structural integrity.

To improve model generalization and compensate for limited data availability on-the-fly data augmentation was applied during YOLO training, generating augmented image variants dynamically at each training iteration. The augmented dataset was divided into training, validation, and testing subsets using an 80%–10%–10% split to ensure unbiased performance and enable reliable assessment of both learning convergence and generalization capability on unseen bridge imagery.

Three variants of the YOLOv8 architecture, namely YOLOv8s, YOLOv8m, and YOLOv8x were selected as Level-0 base learners to introduce architectural diversity within the ensemble. These models differ in depth, width, and parameter count, allowing the ensemble to balance inference speed, detection precision, and sensitivity to subtle corrosion features across multiple spatial scales. Each YOLOv8 base learner was trained independently on the augmented training dataset using the standard YOLOv8 loss formulation. Training was conducted for 200 epochs with early stopping (50 epoch) to terminate training upon convergence, using a batch size of 4 and an input image resolution of 512×512 pixels. To mitigate overfitting, a dropout rate of 0.05 was applied. For the secondary stage, the MetaCNN was trained using AdamW optimizer with learning rate of 0.001 for 150 epochs and with a batch size of 2. YOLOv8 utilizes a decoupled, anchor-free detection head trained with a multi-part loss function. Complete Intersection over Union (CIoU) and Distribution Focal Loss (DFL) are combined to optimize bounding-box regression accuracy, while Binary Cross-Entropy (BCE) loss is used to maximize classification accuracy. Independent training of base learners promotes complementary error patterns and prevents bias propagation across models.

To enable stacking, the final detection layers of each YOLOv8 model were adapted to output dense pixel-wise probability maps rather than discrete predictions. For each input image, each base learner generates a three-channel probability tensor corresponding to the Fair, Poor, and Severe corrosion classes. This representation preserves detailed spatial confidence information that is critical for ensemble fusion. The stacking framework concatenates the probability maps from all three base learners along the channel dimension, forming a nine-channel input tensor for the meta-learner. This stacked representation explicitly captures agreement and disagreement regions among the base models, enabling the meta-learner to perform fine-grained pixel-level reasoning and targeted correction of systematic prediction errors.

The MetaCNN was implemented as a lightweight convolutional Level-1 learner for stacking-based segmentation fusion. The input consists of stacked probability maps generated by the YOLOv8 base learners. For the full ensemble, the input tensor has $C_{in} = 9$channels, corresponding to three YOLOv8 models and three corrosion severity classes. The network contains three 3×3convolutional blocks, each followed by Batch Normalization and ReLU activation. These layers learn nonlinear spatial relationships among the base learners' probability maps. The final prediction is produced using a 1×1convolution followed by sigmoid activation, generating class-wise probability maps for Fair, Poor, and Severe corrosion. Compared to previous quarterly reports, no CBAM or explicit attention mechanism is used in this final architecture; therefore, the fusion is performed entirely through convolutional feature learning. The network is trained using a combined BCE and Dice loss to balance pixel-wise classification accuracy and region-level spatial overlap.

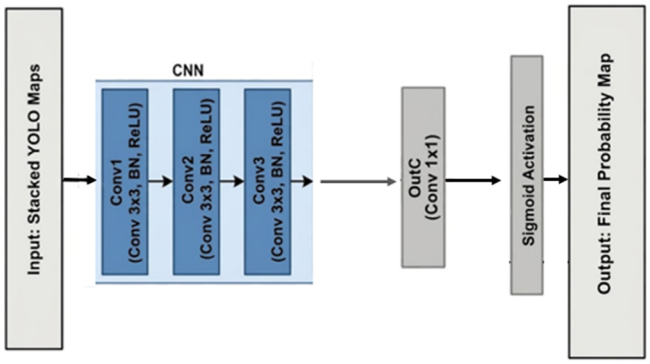


Figure 2. Revised MetaCNN Architecture

1.1 Results

To evaluate segmentation performance, Intersection over Union (IoU), also known as the Jaccard Index, was used as the primary metric. IoU measures the degree of overlap between the predicted corrosion mask and the corresponding ground-truth mask. It is defined as the ratio between the intersection and the union of the predicted and ground-truth regions:

$$IoU = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

where TP represents the number of correctly predicted corrosion pixels, FP represents pixels incorrectly predicted as corrosion, and FN represents corrosion pixels missed by the model. In this study, IoU was computed at the pixel level. For overall corrosion detection, all severity classes were merged into a single damage class before calculating Global IoU. In addition, class-wise IoU was calculated separately for the Fair, Poor, and Severe corrosion categories to evaluate the model's ability to distinguish between different damage severity levels. Higher IoU values indicate better agreement between the predicted masks and the ground-truth annotations.

Figure 3 presents the pixel-level IoU results for the proposed MetaCNN framework and the three YOLOv8 base learners on the same test dataset. The Global IoU was computed by merging all corrosion severity categories into a single damage class, while the Fair, Poor, and Severe columns report class-wise IoU values.

The results show that MetaCNN achieved the best overall segmentation performance, with the highest Global Damage IoU of 92.5%, outperforming all individual YOLOv8 models. MetaCNN also obtained the best class-wise IoU for Fair corrosion (82.8%) and Poor corrosion (44.4%), indicating that the stacking-based fusion improved both overall damage localization and class-level discrimination.

Among the individual YOLO models, YOLOv8m achieved the second-best global damage IoU at 83.1%, followed by YOLOv8x (81.9%) and YOLOv8s (81.2%). Interestingly, YOLOv8s obtained the highest Severe-class IoU of 61.4%, suggesting that the smaller model was more effective for detecting severe corrosion regions in this dataset. However, MetaCNN provided the most balanced and robust performance across the global and class-wise metrics, demonstrating the benefit of integrating multiple base learners through the proposed meta-ensemble framework.

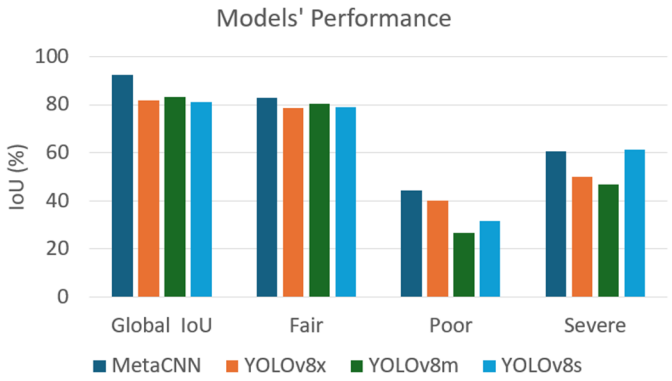
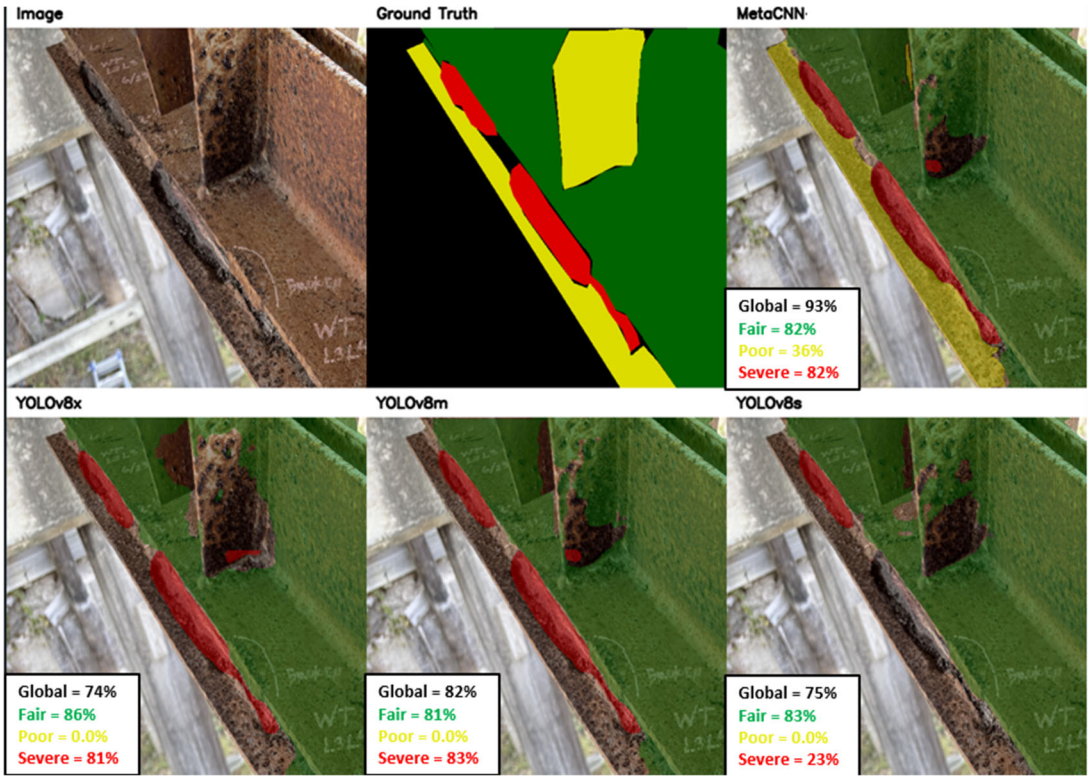


Figure 3. IoU (%) for global damage and class-wise corrosion segmentation

1.2 Sample inference results

In this section, the qualitative results are presented. Figure 4 illustrates representative qualitative results produced by the proposed MetaCNN-based ensemble framework. The figure presents the original bridge inspection image, the corresponding ground-truth annotation, and the predicted segmentation outputs for the three corrosion severity classes: Fair in green, Poor in yellow, and Severe in red. The results show that the proposed framework can effectively localize corrosion regions with complex morphologies and generate spatially consistent predictions across different severity levels. However, some localized misclassifications remain in regions where visual characteristics are shared between adjacent severity classes. These errors may result from ambiguous corrosion textures, gradual transitions between severity levels, or inconsistencies in the ground-truth annotations, suggesting that further refinement of the labeling protocol could improve model training and evaluation reliability.



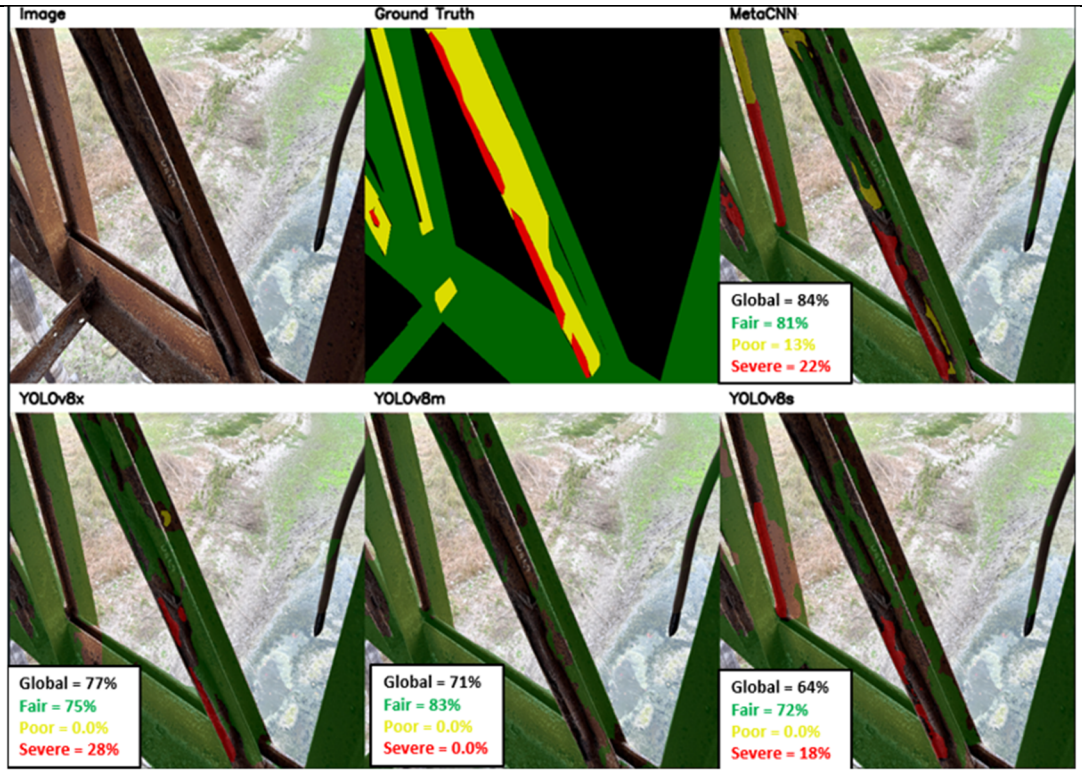


Figure 4. Inferences of Models on DOT dataset

2. Task 2: AR-based software for human-centered bridge inspection

Subtask 2.3: AR software environment for AR headset

This quarter, development focused on improving the accuracy and reliability of the spatial meshing within the headset application. Previous validation and field demonstrations revealed significant discrepancies between physical bridge components and the spatial mesh generated by the headset, as illustrated in Figure 5. This negatively affected the projection-based visualization system, as the projection process is reliant on a consistent spatial mesh alignment. When the spatial mesh did not accurately align with physical structures, projected inspection results appeared out of place on physical structures, reducing visualization accuracy and degrading the reliability of the spatial anchoring.

To improve the quality of the spatial mesh generated and address this issue. Initial efforts focused on modifying the headset tracking configuration in unity, adjusting project settings, and tuning parameters within the spatial meshing script. Numerous configurations were tested, built and deployed to the headset in an attempt to improve mesh fidelity and alignment. However, these modifications did not produce noticeable improvements and spatial mesh generation remained inconsistent and accuracy was unreliable.

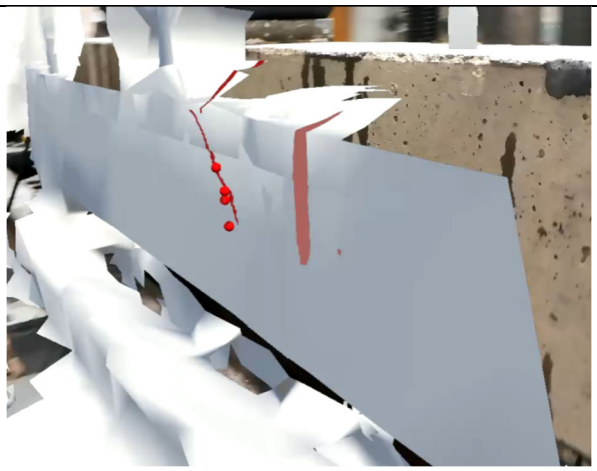


Figure 5. Previous spatial mesh implementation showing misalignment between mesh and physical structure

Following these optimization attempts, we consulted the magic leap developer forums to identify alternative approaches. Based on recommendations from Magic Leap developers, we explored a Unity sample project distributed through Magic Leap Hub 3. The sample project implemented a newer spatial meshing system than the one we were currently integrating into the inspection application. After building and deploying the sample application to the headset, testing consistently demonstrated a noticeably improved spatial mesh quality and alignment with the surrounding physical structures.

The updated spatial mesh implementation was then copied into the bridge inspection application by replacing the previous meshing components with newer scripts provided in the sample project. Unity project settings were also altered to match those of the example project. Following the integration, the application was rebuilt and deployed to the Magic Leap 2 headset for validation. As shown in Figure 6, testing showed that the new implementation substantially improved the accuracy of the generated spatial mesh, which maintained better alignment between virtual and inspected structures. This improvement directly enhanced the accuracy of the projection-based visualization system.

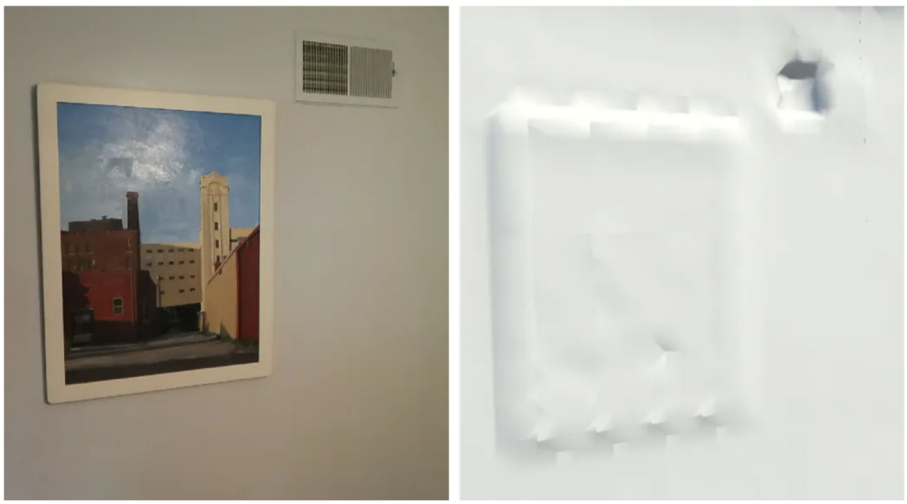


Figure 6. Physical structure at left and the visual mesh generated in the scene

To validate the updated implementation under realistic operating conditions, the updated application was deployed and used in a field demonstration on bridge infrastructure. As shown in Figure 7, the improved spatial mesh maintained substantially better alignment with the steel bridge throughout the inspection, allowing

projected inspection results to remain accurately projected onto the surface. Compared to previous demonstrations, the updated implementation exhibited noticeably improved spatial mesh accuracy.

The improved spatial meshing implementation represents a significant advancement in the overall AR inspection pipeline. By providing a more accurate digital representation of the physical environment, the updated application improves the fidelity of projection-based visualization and establishes a more reliable foundation for persistent spatial anchorage and temporal visualization capabilities.

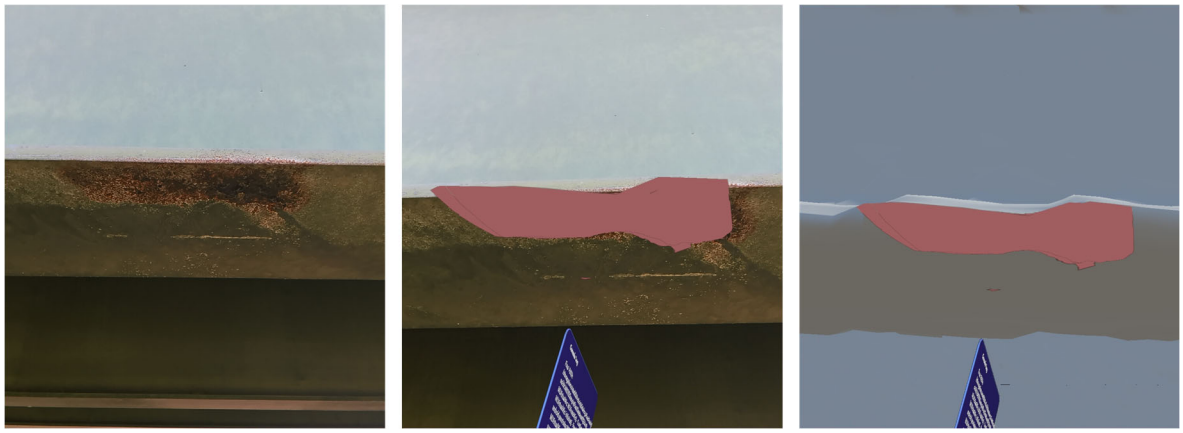


Figure 7. Field demonstration of the updated spatial meshing implementation showing physical structure, inference projection on spatial mesh, and spatial mesh generation, respectively.

Subtask 2.4: AR software environment for tablet device and UAV

In parallel, we use an RGB-D camera to capture synchronized color and depth streams of the construction environment and apply a Simultaneous Localization and Mapping (SLAM) algorithm to incrementally reconstruct a dense 3D point cloud in real time.

The Intel RealSense camera operates as a stereo depth camera, using two infrared imagers separated by a baseline of approximately 5 cm to compute per-pixel depth via stereo matching. The camera simultaneously outputs a color (RGB) image at 1280×720 pixels and a depth image at 848×480 pixels, both at up to 30 frames per second. Because the depth sensor and color sensor are at different physical positions on the device, the raw depth map is expressed in the depth camera's coordinate frame. Before any further processing, each depth frame is reprojected into the color camera's coordinate frame using the factory-calibrated extrinsic rigid transform. This alignment is performed by the RealSense SDK's rs.align() function, producing a depth image that shares the same resolution and field of view as the color image and enabling direct pixel-level correspondence between color values and metric depth values.

To estimate the motion of the camera between consecutive frames, we detect and match local visual features using the Scale-Invariant Feature Transform (SIFT). SIFT keypoints are detected at multiple scales by finding local extrema in a Difference-of-Gaussian scale-space pyramid, and each keypoint is assigned a dominant gradient orientation to achieve rotation invariance. A 128-dimensional descriptor is then computed from the gradient statistics in a 4×4 neighborhood grid around each keypoint, making it robust to moderate illumination and viewpoint changes. Descriptor matching between consecutive frames is performed using the Fast Library for Approximate Nearest Neighbors with a KD-tree index. Ambiguous matches are filtered by Lowe's ratio test, which retains a match only when the nearest-neighbor descriptor distance is less than 80% of the second-nearest-neighbor distance. Given a set of matched feature pairs, we recover the relative camera pose using the Perspective-n-Point (PnP) algorithm. RANSAC runs for 300 iterations at a confidence level of 0.999, ensuring a

very high probability of selecting an outlier-free sample even when 50% of matches are incorrect. After RANSAC selects the best hypothesis and its inlier set, the pose is refined by minimizing the total squared reprojection error over all inliers using the Levenberg–Marquardt (LM) non-linear least-squares algorithm.

Each processed frame is converted to a colored 3D point cloud by back-projecting every valid depth pixel through the pinhole model, yielding 500,000-900,000 raw points per frame at 1280×720 resolution. To control memory and computational load, a voxel grid filter with a cell side length of 5 mm is applied immediately after back-projection: all points within the same voxel are replaced by their centroid, reducing the per-frame point count to approximately 30,000-80,000 points while preserving millimeter-scale geometric detail. Frames are processed in non-overlapping sliding windows of size W=5. Within each window, SIFT+PnP estimates the incremental relative poses between consecutive frame pairs and accumulates them into a window-level local pose.

Two strategies bound memory consumption during prolonged scanning sessions. First, a static detection heuristic compares the current global camera position with the position at the last map update. If the displacement is less than 5 mm, the window's point cloud is discarded without accumulation, avoiding redundant points when the operator holds the camera still. Second, the total number of points in the global map is capped at 2,000,000. When this limit is exceeded, a random subset of that size is retained, preserving the overall spatial distribution of the map while bounding memory consumption to approximately 200-400 MB.

Figure 8 shows the real time SLAM window. The system adopts a three-thread architecture to decouple sensor capture, SLAM computation, and visualization. The capture thread continuously reads synchronized color and depth frames from the RealSense SDK, performs depth-to-color alignment, and pushes frames into a bounded queue of maximum size 2. When the queue is full, the oldest frame is discarded and replaced with the newest, ensuring that the SLAM thread always processes the most temporally recent observations rather than falling progressively further behind real time. The main thread runs the Open3D interactive 3D viewer and an OpenCV live preview window, and monitors keyboard input. This separation ensures that pressing Q to terminate the session is registered within milliseconds regardless of the current SLAM computation load.

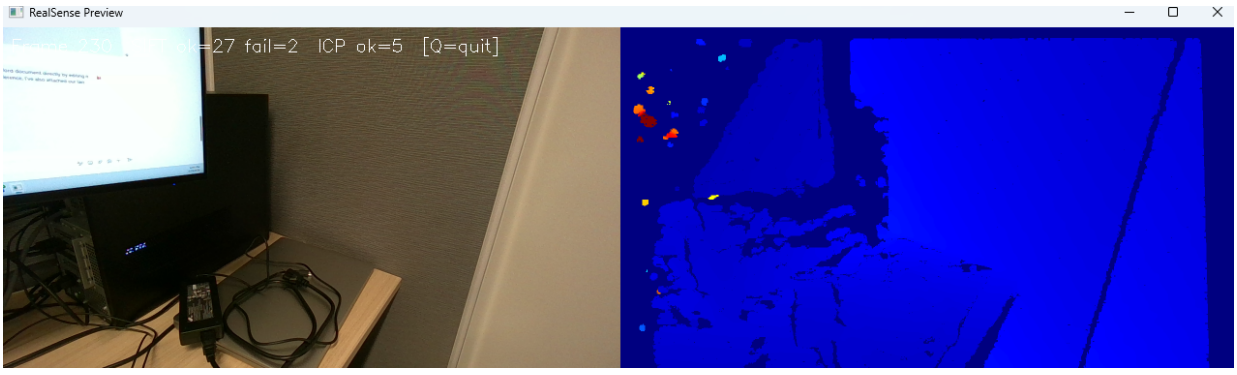


Figure 8. Real time window to demonstrate the acquired RBG and depth images.

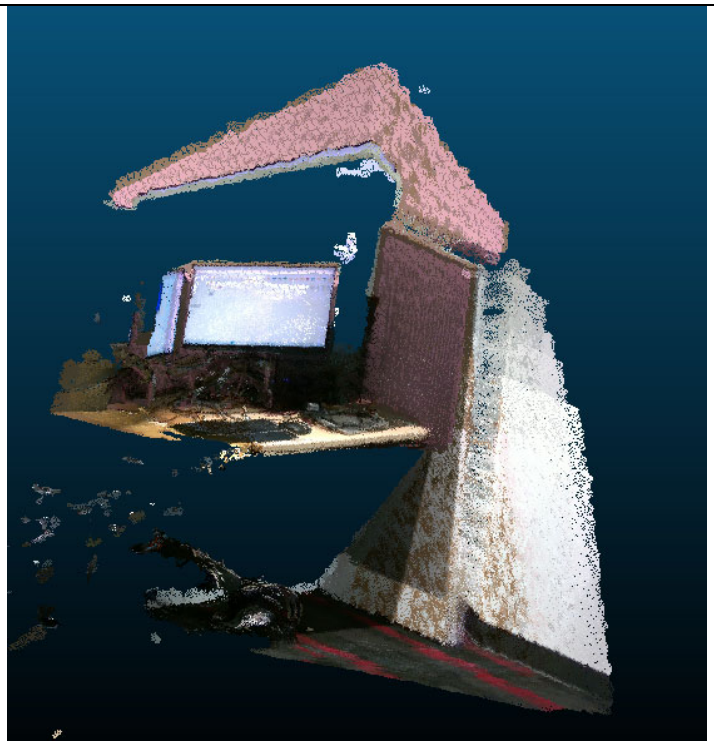


Figure 9. Reconstructed point cloud

Figure 9 shows the reconstructed point cloud using the real-time Intel RealSense camera feed. It is noted that the false color of the reconstructed point cloud is affected by ambient lighting conditions during capture; however, the underlying geometric structure and spatial layout of the scanned components are relatively accurate, as the depth measurements are based on active infrared stereo sensing and are therefore largely independent of visible-light illumination.

The current run processed 51 frames over 22.7 seconds wall time, achieving an effective throughput of approximately 2.7 frames per second. As shown in Figure 10, SIFT+PnP feature extraction and pose estimation dominate at 200 ms per frame pair, followed by voxel grid downsampling at 87 ms and point cloud back-projection at 51 ms. ICP inter-window alignment contributes only 20 ms per window.

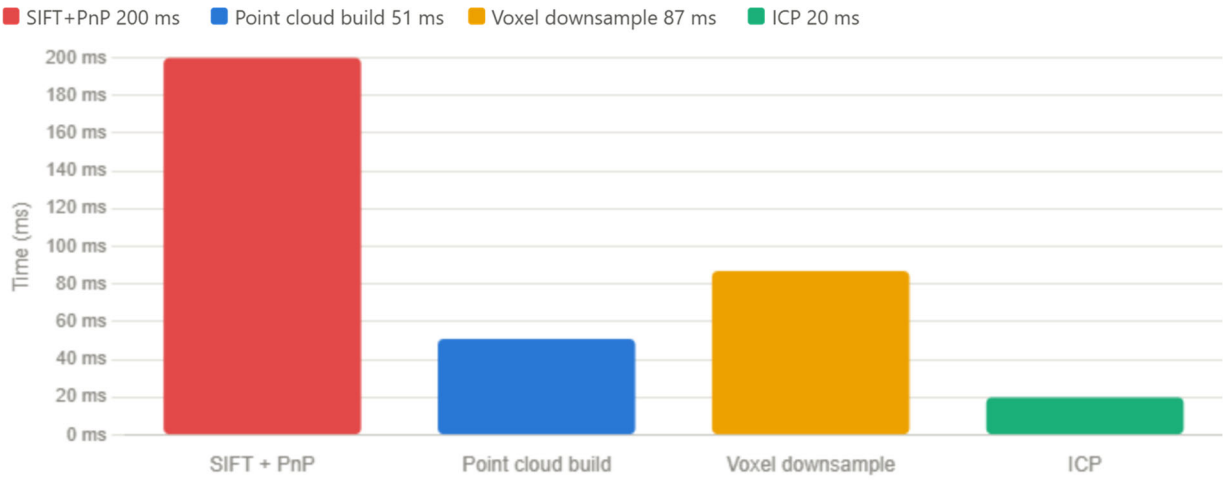


Figure 10. Time performance of the real time slam algorithm

We then evaluated the computational performance of the proposed real-time RGB-D SLAM pipeline on an NVIDIA Jetson Orin NX embedded platform and compared the results against the laptop CPU baseline. The performance advantage of the Jetson Orin NX can be attributed to three hardware characteristics. First, the Jetson adopts a unified memory architecture in which the CPU, GPU, and image signal processor share a single physical memory pool with an aggregate bandwidth of 204 GB/s, compared to approximately 50 GB/s on a typical laptop. Point cloud operations such as voxel grid downsampling and ICP nearest-neighbor search are inherently memory-bandwidth-bound, and the substantially higher bandwidth directly reduces their execution time. Second, the Jetson Orin NX is designed for sustained embedded workloads and does not exhibit the thermal throttling that reduces clock speeds on laptop processors under prolonged high-load conditions, ensuring consistent frame-to-frame timing throughout the scanning session. Third, the ARM Cortex-A78AE cores on the Jetson are optimized for energy-efficient continuous operation rather than peak burst performance, making them better suited to the steady, repetitive nature of the SLAM pipeline than general-purpose laptop processors.

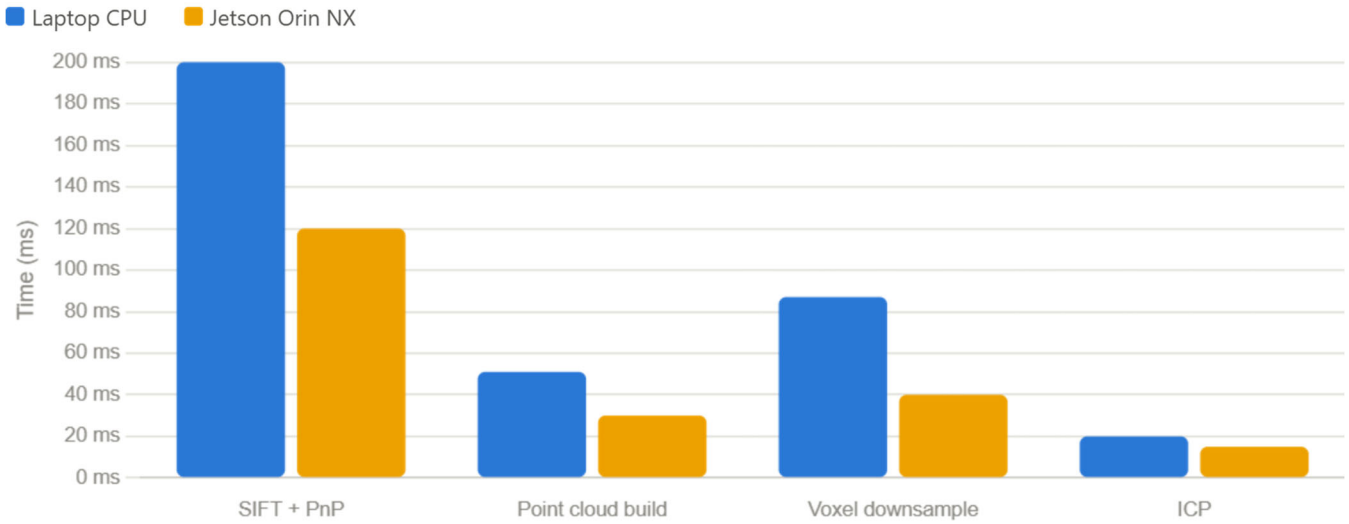


Figure 11. The comparison of time performance between the laptop and Jetson

Circumstance affecting project or budget. (Please describe any challenges encountered or anticipated that might affect the completion of the project within the time, scope and fiscal constraints set forth in the agreement, along with recommended solutions to those problems).

Due to delays in staffing, a one-year no-cost extension may be required to complete the project scope.

Potential Implementation: