

Article

Memory-Efficient 3D LiDAR Graph SLAM for Ballast Water Tank Inspection Robots Using Robust Hierarchical Bundle Adjustment and a Kaczmarz Backend

Sanghyun Cha ¹ , Wonchul Yoo ²  and Tae-wan Kim ^{1,3,4,*}

¹ Deep Learning Laboratory, Department of Naval Architecture and Ocean Engineering, College of Engineering, Seoul National University, Seoul 08826, Republic of Korea; kunja4618@snu.ac.kr

² Avikus Co., Ltd., 11F, 70 Nonhyeon-ro 85-gil, Gangnam-gu, Seoul 06234, Republic of Korea; wchyoo@snu.ac.kr

³ Research Institute of Marine Systems Engineering, Seoul National University, Seoul 08826, Republic of Korea

⁴ Institute of Engineering Research, Seoul National University, Seoul 08826, Republic of Korea

* Correspondence: taewan@snu.ac.kr

Abstract

Autonomous inspection of ballast water tanks requires three-dimensional (3D) LiDAR-based simultaneous localization and mapping (SLAM) in Global Positioning System (GPS)-denied, geometrically repetitive interiors, where sensing, mapping, and control modules share a limited onboard memory budget. Graph SLAM backends that rely on sparse factorization can incur fill-in, increasing peak memory and limiting deployment on edge computers. The proposed architecture couples a robust hierarchical bundle adjustment frontend with a factorization-free Kaczmarz backend. The frontend combines residual-adaptive weighting, damped and bounded pose updates, soft fallback, local-map compression, and memory-aware keyframe control. The backend stores the whitened Jacobian in compressed sparse row (CSR) format and performs row-wise projections without explicitly forming the normal equations, a Cholesky factor, or a transpose cache. Evaluation was conducted on Norwegian University of Science and Technology (NTNU) Ballast Water Tank missions 1–3, containing 851, 1202, and 1084 LiDAR frames. Following robust local bundle adjustment and verified similarity alignment, translational root-mean-square errors were 0.080, 0.110, and 0.127 m, corresponding to 0.137%, 0.143%, and 0.122% of the reference path lengths; archived baseline ratios ranged from 0.281% to 0.372%. These results support a numerical architecture that combines frontend stabilization, row-wise optimization, and memory-aware policies for resource-constrained marine inspection robots.

Keywords: ballast water tank inspection; 3D LiDAR SLAM; hierarchical bundle adjustment; Kaczmarz method; memory-efficient SLAM



Academic Editor: Vincenzo Crupi

Received: 23 June 2026

Revised: 8 July 2026

Accepted: 9 July 2026

Published: 13 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Ship ballast water tanks, double-bottom compartments, cofferdams, and similar enclosed structures are among the most demanding locations for structural inspection. They are typically dark, Global Positioning System (GPS)-denied, geometrically repetitive, and accessible only through narrow hatches or manholes. Corrosion products, deteriorated coatings, poor ventilation, limited communication, and slippery or contaminated surfaces further increase the risk and duration of manual inspection. Autonomous or remotely supervised robots can reduce human exposure while producing spatially registered point clouds, images, and condition-monitoring records for engineering assessment [1–3].

Reliable navigation in these environments cannot rely solely on nominal design information. Even when computer-aided design drawings and construction records are available, the as-built or in-service geometry may differ because of fabrication tolerances, welding distortion, thermal deformation, coating accumulation or loss, corrosion, repair, retrofitting, temporary equipment, and undocumented obstacles. The robot must therefore update its spatial representation from onboard observations rather than treat the design model as an exact navigation map. Three-dimensional (3D) LiDAR-based simultaneous localization and mapping (SLAM) is well suited to this task because it jointly estimates the sensor trajectory and the surrounding structural geometry [4,5].

Modern LiDAR SLAM systems commonly separate high-rate odometry or scan registration from lower-rate mapping and graph optimization. LiDAR Odometry and Mapping (LOAM) established an influential odometry-and-mapping decomposition [6], while LiDAR-inertial odometry via smoothing and mapping (LIO-SAM) incorporated inertial preintegration, keyframes, and loop-closure factors within a smoothing framework [7]. Hierarchical LiDAR bundle adjustment improves large-scale consistency by decomposing the mapping problem into local optimization windows and higher-level pose constraints [8]. Ballast water tanks, however, present failure modes that are less prominent in general environments: long planar surfaces weakly constrain motion in particular directions, repetitive stiffeners create ambiguous correspondences, narrow passages cause abrupt changes in visibility, and vibration or contact can introduce transient motion distortion. A single erroneous local registration can therefore produce a large pose update that contaminates later keyframes and global constraints.

Graph SLAM backends formulate poses and measurements as a nonlinear least-squares problem. Square Root SAM, iSAM, iSAM2, and g2o show that sparse factorization and incremental graphical inference can solve these problems efficiently [9–12]. Elimination-based solvers must nevertheless retain factorization structures. As graph connectivity and loop-closure density increase, elimination can introduce fill-in: entries that are zero in the original sparse Jacobian become nonzero in the resulting factors [13–15]. Reordering can reduce fill-in, but it does not eliminate the need to construct and store factorization-related data.

This memory growth is operationally significant on compact inspection robots. The NVIDIA Jetson Orin Nano Super Developer Kit, for example, provides 8 GB of low-power double data rate 5 (LPDDR5) memory and a nominal 7–25 W power range [16]. The same physical memory is shared by the operating system, robotics middleware, LiDAR packet buffers, sensor drivers, point-cloud filtering, scan registration, local-map storage, inspection perception, logging, and graphics processing unit (GPU) workloads. A backend that is practical on a workstation can therefore become the peak-memory bottleneck in a sealed, thermally constrained robotic platform.

Iterative sparse solvers provide alternatives to direct factorization. In this paper, Krylov-family methods refer to iterative solvers such as conjugate gradient, preconditioned conjugate gradient (PCG), least-squares QR (LSQR), and least-squares minimal residual (LSMR), which construct successive search subspaces through matrix-vector products rather than explicitly factorizing the system matrix. PCG can be efficient when the linear system and preconditioner are favorable, whereas LSQR and LSMR operate directly on sparse least-squares systems [17–20]. Kaczmarz methods form a distinct row-action family in which each iteration projects the current estimate toward the hyperplane defined by a selected row [21]. Randomized row selection offers expected convergence guarantees for suitable consistent systems, and weighted or block variants extend the design space [22–24]. The present study does not assume that Kaczmarz is universally faster or more accurate than Cholesky, PCG, LSQR, or LSMR. Its relevance here is the row-wise access pattern:

an update can be executed from a compressed sparse row (CSR) representation without explicitly forming the normal equations, a Cholesky factor, or a transpose cache.

Accordingly, this study develops a system-level SLAM architecture for resource-constrained marine inspection rather than a new mathematical variant of the Kaczmarz method. The architecture combines robust local hierarchical bundle adjustment (HBA), compressed local maps, row-wise global optimization, and memory-aware operating policies. An 8 GB-class edge computer is used as the deployment reference; however, the reported experiments establish architectural feasibility rather than a completed real-time Jetson implementation. The principal contributions are as follows:

1. A robust local HBA frontend is developed for repetitive and locally degenerate ship-interior geometry by combining residual-adaptive weighting, pose-step clamping, damping, and soft fallback.
2. A factorization-free global backend is formulated directly on the whitened sparse least-squares system and solved through CSR-based row projections without explicitly forming the approximate Hessian or a Cholesky factor.
3. Local-map compression, keyframe regulation, and optimization-range control are co-designed with the numerical backend to bound the problem presented to the solver.
4. The complete architecture is evaluated on NTNU Ballast Water Tank missions 1–3 using absolute and path-normalized trajectory errors.

2. Materials and Methods

2.1. System Architecture and Onboard Design Objective

The pipeline comprises six stages: (i) LiDAR acquisition and geometric feature extraction; (ii) initial LiDAR odometry; (iii) robust local HBA over overlapping windows; (iv) keyframe and local-map compression; (v) factor-graph assembly and CSR linearization; and (vi) row-wise Kaczmarz optimization under memory-aware control. The frontend produces an initial pose sequence and geometric correspondences. Consecutive scans are grouped into overlapping local windows, and their poses are refined jointly. The optimized local poses and compressed submaps then provide odometry, submap, and loop-closure constraints for the global graph. Before each global solve, the memory controller evaluates the keyframe and factor counts, the number of nonzero Jacobian entries, point-cloud storage, and current process memory. The backend subsequently executes a prescribed budget of Kaczmarz row projections. Figure 1 summarizes the complete architecture.

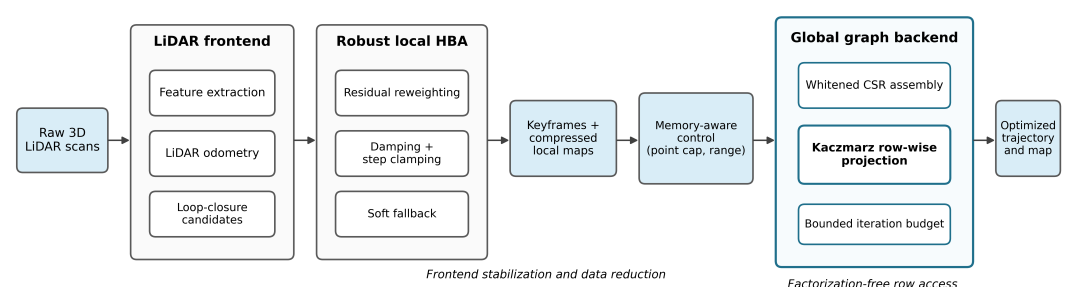


Figure 1. Proposed marine-inspection simultaneous localization and mapping (SLAM) architecture. The frontend combines light detection and ranging (LiDAR) odometry, robust local hierarchical bundle adjustment (HBA), soft fallback, and local-map compression. The backend assembles weighted factor rows in compressed sparse row (CSR) format and applies Kaczmarz projections subject to keyframe, point-cap, and optimization-range controls.

The design does not assume that the full physical memory of the edge computer is available to graph optimization. A safety margin must remain after accounting for the operating system, middleware, sensor and frontend buffers, inspection and perception

modules, map storage, and temporary allocations. Memory efficiency is therefore treated as an end-to-end operating constraint rather than solely as a solver-level metric.

2.2. Robust Hierarchical Bundle Adjustment Frontend

2.2.1. Local Window Formulation

Let the ordered keyframe poses belong to the special Euclidean group $SE(3)$:

$$\mathcal{T} = \{\mathbf{T}_1, \mathbf{T}_2, \dots, \mathbf{T}_N\}, \quad \mathbf{T}_i \in SE(3). \quad (1)$$

The trajectory is divided into overlapping local windows

$$\mathcal{W}_q = \{i_q, i_q + 1, \dots, i_q + n_w - 1\}, \quad (2)$$

where n_w is the local window size. The overlap preserves geometric continuity between adjacent submaps and enables constraints to propagate through the hierarchy.

For a geometric correspondence k in window $\mathcal{W}_q$, let $\mathbf{r}_k$ denote a point-to-plane, point-to-line, or feature residual, and let $\mathbf{\Omega}_k$ be its information matrix. The robust local objective is

$$\arg \min_{\{\mathbf{T}_i\}_{i \in \mathcal{W}_q}} \sum_{k \in \mathcal{C}_q} w_k \mathbf{r}_k^\top \mathbf{\Omega}_k \mathbf{r}_k, \quad (3)$$

where $\mathcal{C}_q$ is the correspondence set and $w_k \in [w_{\min}, 1]$ is a dynamic reliability weight.

2.2.2. Residual-Dependent Weight Reduction

Ballast-tank scans can contain large residuals because of repeated stiffeners, partially observed surfaces, abrupt visibility changes, or motion distortion. Rejecting every large residual can also discard temporarily informative constraints. The frontend therefore reduces residual influence progressively. Let

$$s_k = \sqrt{\mathbf{r}_k^\top \mathbf{\Omega}_k \mathbf{r}_k} \quad (4)$$

be the normalized residual magnitude. In the archived implementation used for the reported runs, the residual-dependent weight was controlled by a residual threshold δ_r and a minimum weight $w_{\min}$:

$$w_k = \begin{cases} 1, & s_k \leq \delta_r, \\ \max\left(w_{\min}, \frac{\delta_r}{s_k}\right), & s_k > \delta_r. \end{cases} \quad (5)$$

The threshold δ_r controls the onset of down-weighting, and $w_{\min}$ prevents a large residual from being assigned zero influence. Smaller δ_r or larger suppression through $w_{\min}$ yields more aggressive outlier attenuation but can also reduce useful geometric constraints in planar or partially observed tank sections. Conversely, larger thresholds preserve more correspondences but allow ambiguous matches to retain greater influence. The archived Park and NTNU recovery configurations used $\delta_r = 0.6$ and $w_{\min} = 0.05$; these values were fixed across the reported recovery runs rather than re-tuned per mission. Additional exploratory NTNU robust-local-BA logs with lower residual thresholds and tighter step clamps were available in the archive, but they did not constitute a complete full-factorial sensitivity study.

Thus, nominal constraints retain full influence, while large residuals are progressively down-weighted with a nonzero lower bound. After linearization, the weighted local system becomes

$$\min_{\delta\boldsymbol{\xi}} \left\| \mathbf{W}^{1/2} (\mathbf{J}\delta\boldsymbol{\xi} + \mathbf{r}) \right\|_2^2, \quad (6)$$

where $\delta\boldsymbol{\xi}$ stacks the local pose increments. The archived recovery constants used in the reported Park and NTNU runs are summarized in Table 1.

Table 1. Archived robust local-BA and memory-policy settings used in the reported Park and NTNU recovery runs. These operating settings were recovered from archived source/configuration logs and are not claimed to be globally optimized universal parameters.

Parameter	Value	Unit	Role
Residual threshold δ_r	0.6	residual units	Onset of residual down-weighting in Equation (5).
Minimum residual weight $w_{\min}$	0.05	–	Lower bound preventing full removal of high-residual constraints.
Local update damping μ	10^{-6}	–	Numerical stabilization of the local normal equations.
Translation clamp $t_{\max}$	0.75	m	Maximum translational pose update applied in one local iteration.
Rotation clamp $\theta_{\max}$	5.0	deg	Maximum rotational pose update applied in one local iteration.
Fallback blend α	0.0	–	Conservative fallback setting retained in the recovery logs.
Processing stride	5	frames	Input-frame stride for the retained recovery runs.
Keyframe voxel leaf	0.8	m	Local-map/keyframe compression setting.
Keyframe point cap	50,000	points/cloud	Upper bound on stored keyframe point count.
Global-BA pose guard	20	poses	Range guard; global BA is skipped when the pose count exceeds this limit.

2.2.3. Damping and Pose-Step Clamping

Repeated planar geometry can leave some motion directions weakly observable and make the local system ill-conditioned. A damped update is therefore computed as

$$(\mathbf{J}^\top \mathbf{W} \mathbf{J} + \mu \mathbf{I}) \delta\boldsymbol{\xi} = -\mathbf{J}^\top \mathbf{W} \mathbf{r}, \quad (7)$$

where $\mu > 0$ is increased when the objective fails to decrease or when an excessive update is detected.

For each pose increment

$$\delta\boldsymbol{\xi}_i = \begin{bmatrix} \delta\mathbf{t}_i^\top & \delta\boldsymbol{\theta}_i^\top \end{bmatrix}^\top, \quad (8)$$

the translational and rotational components are clamped independently:

$$\delta\mathbf{t}_i^c = \delta\mathbf{t}_i \min\left(1, \frac{t_{\max}}{\|\delta\mathbf{t}_i\|_2 + \epsilon}\right), \quad (9)$$

$$\delta\boldsymbol{\theta}_i^c = \delta\boldsymbol{\theta}_i \min\left(1, \frac{\theta_{\max}}{\|\delta\boldsymbol{\theta}_i\|_2 + \epsilon}\right). \quad (10)$$

This clamping limits the influence of an unstable linearization and keeps each update within a scale compatible with physically plausible inter-frame motion.

2.2.4. Soft Fallback

A local solution is flagged as unreliable when the objective increases, the inlier ratio falls below a threshold, scan overlap becomes insufficient, or repeated clamping is required. An abrupt return to the pre-optimization pose can create a trajectory discontinuity. The soft fallback instead interpolates between the predicted pose $\mathbf{T}_{\text{pred}}$ and the locally optimized pose $\mathbf{T}_{\text{opt}}$:

$$\mathbf{T}_{\text{app}} = \mathbf{T}_{\text{pred}} \text{Exp}\left(\alpha \text{Log}\left(\mathbf{T}_{\text{pred}}^{-1} \mathbf{T}_{\text{opt}}\right)\right), \quad 0 \leq \alpha \leq 1. \quad (11)$$

A high-confidence solution uses $\alpha \approx 1$, whereas a low-confidence solution is applied conservatively using a smaller value of α .

2.2.5. Local-Map Compression

After local optimization, the point clouds in window $\mathcal{W}_q$ are transformed into a common coordinate frame:

$$\mathcal{M}_q = \bigcup_{i \in \mathcal{W}_q} \mathbf{T}_i \mathcal{P}_i, \quad (12)$$

where $\mathcal{P}_i$ is the point cloud associated with pose i . A compression operator $\mathcal{C}(\cdot)$ produces

$$\widetilde{\mathcal{M}}_q = \mathcal{C}(\mathcal{M}_q; v_q, n_{\text{max}}, \mathcal{F}_q), \quad (13)$$

where v_q is the voxel resolution, n_{max} is the point cap, and $\mathcal{F}_q$ denotes geometry-preserving selection criteria. Broad planar regions are downsampled, whereas edges, stiffener boundaries, brackets, and other geometrically informative structures are preferentially retained.

2.3. Global Graph SLAM Formulation

Let the stacked global pose state be

$$\mathbf{x} = [\boldsymbol{\xi}_1^\top, \boldsymbol{\xi}_2^\top, \dots, \boldsymbol{\xi}_N^\top]^\top. \quad (14)$$

For a relative-pose measurement $\mathbf{Z}_{ij}$ between poses $\mathbf{T}_i$ and $\mathbf{T}_j$, the residual is defined on SE(3) as

$$\mathbf{r}_{ij}(\mathbf{x}) = \text{Log}\left(\mathbf{Z}_{ij}^{-1} \left(\mathbf{T}_i^{-1} \mathbf{T}_j\right)\right). \quad (15)$$

The nonlinear graph objective is

$$\min_{\mathbf{x}} \sum_{(i,j) \in \mathcal{E}} \rho\left(\mathbf{r}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{r}_{ij}\right) + \sum_{p \in \mathcal{P}} \rho\left(\mathbf{r}_p^\top \boldsymbol{\Omega}_p \mathbf{r}_p\right), \quad (16)$$

where $\mathcal{E}$ is the set of relative-pose factors and $\mathcal{P}$ contains prior factors.

At the current estimate, each factor is linearized:

$$\mathbf{r}_k(\mathbf{x} \oplus \Delta \mathbf{x}) \approx \mathbf{r}_k(\mathbf{x}) + \mathbf{J}_k \Delta \mathbf{x}. \quad (17)$$

Let $\boldsymbol{\Lambda}_k$ satisfy

$$\boldsymbol{\Lambda}_k^\top \boldsymbol{\Lambda}_k = \boldsymbol{\Omega}_k. \quad (18)$$

With robust factor weight w_k , the whitened block row and right-hand side are

$$\mathbf{A}_k = \sqrt{w_k} \boldsymbol{\Lambda}_k \mathbf{J}_k, \quad \mathbf{b}_k = -\sqrt{w_k} \boldsymbol{\Lambda}_k \mathbf{r}_k. \quad (19)$$

Stacking the factor rows yields

$$\min_{\Delta \mathbf{x}} \|\mathbf{A} \Delta \mathbf{x} - \mathbf{b}\|_2^2. \quad (20)$$

A conventional normal-equation backend forms

$$\mathbf{H}\Delta\mathbf{x} = \mathbf{g}, \quad \mathbf{H} = \mathbf{A}^\top \mathbf{A}, \quad \mathbf{g} = \mathbf{A}^\top \mathbf{b}, \quad (21)$$

and then factorizes $\mathbf{H}$. The proposed backend instead operates directly on the rows of $\mathbf{A}$. Figure 2 traces a relative-pose factor through linearization, whitening, sparse row assembly, and a subsequent row-wise Kaczmarz update.

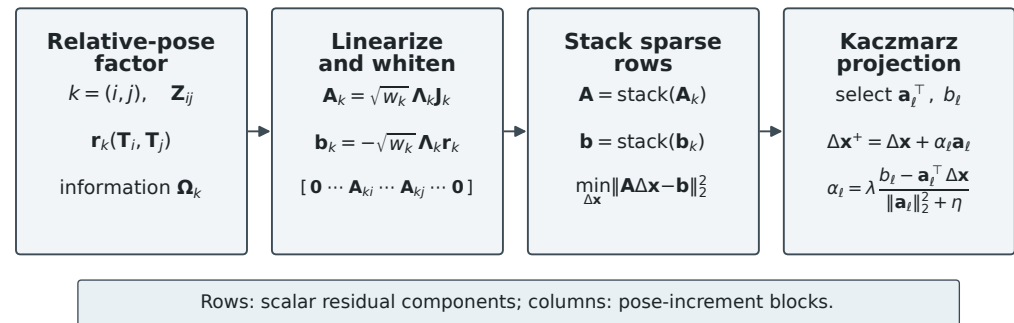


Figure 2. Transformation of a relative-pose factor into a row-wise Kaczmarz update. Linearization and whitening of factor k produce a sparse block row whose nonzero column blocks correspond only to the connected pose increments. Stacking the factor rows yields the least-squares system $(\mathbf{A}, \mathbf{b})$; each Kaczmarz step then selects one scalar row $\mathbf{a}_\ell^\top$ and its right-hand-side element b_ℓ .

2.4. CSR-Based Kaczmarz Backend

Let $\mathbf{a}_i^\top$ denote row i of $\mathbf{A}$, and let b_i be the corresponding right-hand-side element. Given the current increment estimate $\Delta\mathbf{x}^{(\ell)}$, a relaxed Kaczmarz update with a numerical safeguard is

$$\Delta\mathbf{x}^{(\ell+1)} = \Delta\mathbf{x}^{(\ell)} + \lambda \frac{b_i - \mathbf{a}_i^\top \Delta\mathbf{x}^{(\ell)}}{\|\mathbf{a}_i\|_2^2 + \eta} \mathbf{a}_i, \quad (22)$$

where λ is the relaxation factor and $\eta > 0$ prevents excessive updates when a selected row has a very small norm.

For row-norm sampling,

$$p_i = \frac{\|\mathbf{a}_i\|_2^2}{\|\mathbf{A}\|_F^2}. \quad (23)$$

When frontend confidence scores or additional robust weights are retained separately from the whitening in $\mathbf{A}$, the sampling probability can be generalized to

$$p_i = \frac{q_i \|\mathbf{a}_i\|_2^2}{\sum_j q_j \|\mathbf{a}_j\|_2^2}, \quad 0 \leq q_i \leq 1. \quad (24)$$

This rule reduces the selection frequency of low-confidence rows while retaining their bounded contribution to the solution.

The matrix is stored in CSR arrays containing nonzero values, column indices, and row pointers. Figure 3 shows a representative Jacobian pattern, dominated by local-factor support with limited off-diagonal blocks introduced by nonlocal constraints. When row i is selected, only the entries between `row_ptr[i]` and `row_ptr[i+1]` are accessed. If $\mathcal{N}(i)$ denotes the corresponding set of nonzero columns, the sparse update is

$$\Delta x_j^{(\ell+1)} = \Delta x_j^{(\ell)} + \lambda \frac{b_i - \sum_{h \in \mathcal{N}(i)} a_{ih} \Delta x_h^{(\ell)}}{\sum_{h \in \mathcal{N}(i)} a_{ih}^2 + \eta} a_{ij}, \quad j \in \mathcal{N}(i). \quad (25)$$

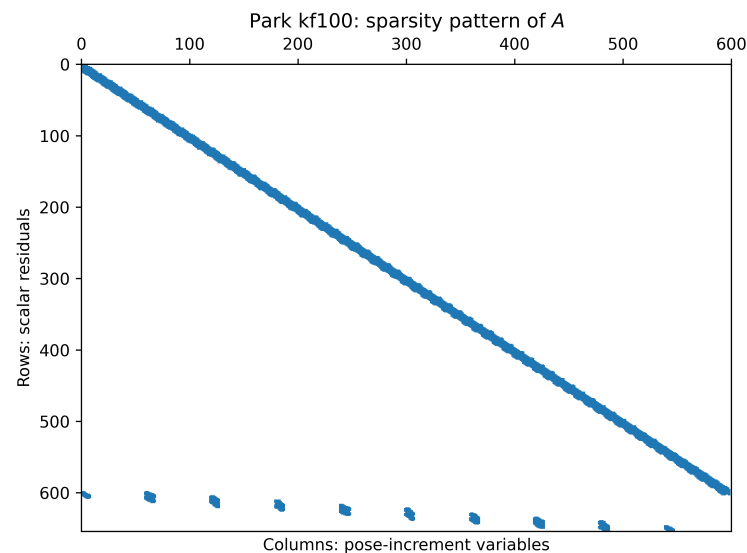


Figure 3. Sparsity pattern of the assembled Jacobian $\mathbf{A}$ for the controlled 100-keyframe Park pose graph. The dominant diagonal structure is produced by local relative-pose factors, whereas the lower off-diagonal blocks represent nonlocal constraints.

Following the inner row projections, the increment associated with pose i is applied on the manifold:

$$\mathbf{T}_i^+ = \mathbf{T}_i \text{Exp}(\delta \xi_i^\wedge). \quad (26)$$

The linear solve terminates when either the normalized residual or the successive-increment norm satisfies

$$\frac{\|\mathbf{A}\Delta\mathbf{x}^{(\ell)} - \mathbf{b}\|_2}{\|\mathbf{b}\|_2 + \epsilon} < \epsilon_r, \quad \text{or} \quad \|\Delta\mathbf{x}^{(\ell)} - \Delta\mathbf{x}^{(\ell-1)}\|_2 < \epsilon_x, \quad (27)$$

or when the prescribed row-projection budget is exhausted. Because practical SLAM systems are generally inconsistent owing to sensor noise, linearization error, and residual outliers, the row projections are used as bounded residual-reduction steps within each nonlinear iteration; they are not assumed to satisfy every factor row exactly or to recover the exact least-squares solution.

2.5. Structural Memory Model and Memory Guard

The principal solver-side storage terms are the CSR representation of $\mathbf{A}$, the vector $\mathbf{b}$, the pose increment, the row norms, and the sampling data structure. Their approximate memory requirement is

$$M_{\text{CSR}} \approx \text{nnz}(\mathbf{A})(s_{\text{val}} + s_{\text{col}}) + (m + 1)s_{\text{ptr}} + m s_b + n s_x + m s_p, \quad (28)$$

where m and n are the row and column counts, and the s terms denote the bytes required by the corresponding data types.

The full-pipeline memory estimate is

$$M_{\text{total}} = M_{\text{system}} + M_{\text{sensor}} + M_{\text{frontend}} + M_{\text{map}} + M_{\text{graph}} + M_{\text{solver}} + M_{\text{margin}}. \quad (29)$$

The memory guard monitors keyframe count, point count, factor count, residual-row count, $\text{nnz}(\mathbf{A})$, and process resident set size. Three operating regions are used. In the normal region, nominal keyframe and map-resolution settings are retained. In the caution region, keyframe thresholds and voxel size are increased, point caps are tightened, and weak factors are suppressed. In the critical region, new keyframes may be deferred, inactive

point clouds are compressed into submaps or descriptors, and the global optimization range may be bounded or skipped.

A new keyframe is accepted when at least one of the following conditions is met:

$$\|\mathbf{t}_t - \mathbf{t}_{\text{last}}\|_2 > \tau_t, \quad (30)$$

$$\left\| \text{Log}(\mathbf{R}_{\text{last}}^\top \mathbf{R}_t) \right\|_2 > \tau_R, \quad (31)$$

or

$$\text{overlap}(\mathcal{P}_t, \widetilde{\mathcal{M}}_{\text{local}}) < \tau_o. \quad (32)$$

The thresholds are adapted conservatively as the estimated memory margin decreases.

2.6. Experimental Data and Evaluation Protocol

Marine-domain validation used the three archived evaluation sequences denoted NTNU Ballast Water Tank missions 1–3 in this study. No other missions from the released NTNU Ballast Water Tank dataset were included in the quantitative trajectory evaluation. The public dataset contains ballast-tank data collected across three anonymized vessels, referred to as FPSO1, FPSO2, and Oil Tanker, and includes multiple side-tank, bilge, and double-bottom deployments with a collision-tolerant aerial robot and a handheld sensing platform [1,25]. The inspection platform and representative field deployments are described by Dharmadhikari et al. [1], while the data analyzed here are distributed through the NTNU Autonomous Robots Laboratory dataset repository [25]. The repository identifies the LiDAR as an Ouster OS0-64 (Rev-D; Ouster, Inc., San Francisco, CA, USA), and software-version metadata for the archived offline playback and recovery scripts were not retained in the archived logs. The evaluated sequences contained 851, 1202, and 1084 point-cloud frames, with reference path lengths of 58.48, 76.52, and 103.96 m, respectively. The public dataset anonymizes the vessels and the archived evaluation exports used here did not include vessel computer-aided-design models, tank-boundary polygons, or independently surveyed tank dimensions. Therefore, the spatial scale of each mission is reported using the reference-trajectory path length and bounding-box diagonal, and tank outlines were not overlaid on the trajectory plots to avoid introducing artificial geometry. Raw point counts were obtained from point-cloud headers and archived run logs. Intermediate quantities that were not recorded, including the exact point count after each voxel-filtering and local-map-compression stage, were not reconstructed retrospectively.

The Park dataset was used only for system-level memory profiling because the archived Park runs retained multiple operating-policy configurations, including stride, voxel resolution, point cap, and global-optimization range. Thus, the NTNU missions provide the marine-domain trajectory validation, whereas the Park runs provide controlled memory and runtime operating points for the system-level sensitivity analysis.

For a trajectory containing N paired positions, the translational root-mean-square error (RMSE) was computed as

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N \|\mathbf{p}_i - \mathbf{p}_i^{\text{ref}}\|_2^2}. \quad (33)$$

Here, $\mathbf{p}_i \in \mathbb{R}^3$ denotes the estimated sensor position at the i -th paired sample, and $\mathbf{p}_i^{\text{ref}} \in \mathbb{R}^3$ denotes the corresponding position on the stored comparison trajectory. The path-normalized error was

$$e_{\text{path}} = 100 \frac{\text{RMSE}}{L_{\text{ref}}}, \quad (34)$$

where L_{ref} is the reference path length. The environment-scale metric was

$$e_{\text{bbox}} = 100 \frac{\text{RMSE}}{D_{\text{bbox}}}, \quad (35)$$

where D_{bbox} is the diagonal of the reference trajectory bounding box.

The archived NTNU evaluation paired trajectory samples using the index-based HBA top-index procedure recorded with the experiments and then applied Umeyama Sim(3) alignment. The stored comparison trajectory was used as the reference trajectory for this archived evaluation. It should not be interpreted as an independently surveyed, metrology-grade ground truth; consequently, the RMSE values quantify deviation from the stored comparison trajectory under the recorded pairing and alignment procedure. Final robust local bundle adjustment (BA) trajectories were evaluated after the same retained pairing/alignment convention. Retained reference and estimated trajectory coordinate files and the recomputation script were audited during the revision. Re-evaluating the same stored trajectories with the same pairing and alignment settings reproduces the reported RMSE values deterministically up to numerical precision. However, repeated end-to-end SLAM reruns from the original bags with different stochastic seeds or runtime scheduling conditions were not archived. The baseline and robust values should therefore be interpreted as trajectory-level validation, not as a solver-isolated ablation performed under identical alignment conditions. Because the baseline and robust trajectories were archived with different alignment labels, the baseline errors may be overestimated relative to the robust local BA results.

A separate controlled benchmark compared direct, Krylov-family, least-squares, and Kaczmarz methods on identical $(\mathbf{A}, \mathbf{b})$ systems. Solver-side structural-memory estimates were kept distinct from end-to-end peak resident set size (RSS), which also includes point-cloud input/output, local maps, graph assembly, runtime libraries, and trajectory storage. CPU utilization, GPU utilization, thermal throttling, and real-time latency were not logged in the archived offline playback experiments; the results are therefore interpreted as memory-architecture feasibility evidence rather than a complete real-time edge-computing benchmark.

2.7. Use of Generative Artificial Intelligence in Manuscript Preparation

OpenAI ChatGPT (GPT-5.5 Pro) was used to assist with English-language revision, structural editing, reference-format consistency, and LaTeX preparation. It was not used to generate or alter experimental data. The authors reviewed all generated text and independently verified the technical statements, numerical results, citations, and figures; they take full responsibility for the published content.

3. Results

3.1. NTNU Ballast Water Tank Mission Scale

Table 2 summarizes the input scale of the three marine validation missions. Each frame contained approximately 32,800 raw LiDAR points, yielding tens of millions of points per mission.

The missions differed in path length and spatial extent but had similar per-frame LiDAR density. Absolute RMSE was therefore complemented by path- and bounding-box-normalized metrics for cross-mission comparison.

Table 2. Norwegian University of Science and Technology (NTNU) Ballast Water Tank mission scale. Physical tank-boundary models were not available in the archived evaluation exports; the path length, bounding-box diagonal, and start–end ratio therefore describe the spatial extent and topology of the stored comparison trajectory, not the physical tank dimensions.

Dataset	Frames	Raw Points	Mean Points per frame	Path Length (m)	Bounding-Box Diagonal (m)	Start–End Ratio	Path Topology
NTNU mission 1	851	27.9 million	32.8 thousand	58.48	9.486	0.092	Open
NTNU mission 2	1202	39.4 million	32.8 thousand	76.52	10.509	0.001	Near-return
NTNU mission 3	1084	35.5 million	32.8 thousand	103.96	15.411	0.001	Near-return

3.2. Trajectory Accuracy of the Baseline and Robust Local BA Configurations

Table 3 reports the archived trajectory-level results for the baseline and robust local BA configurations. Because the two archived states were not evaluated under identical alignment conditions, the comparison is descriptive rather than a controlled single-variable ablation.

Table 3. Archived NTNU trajectory errors for the baseline and robust local bundle adjustment (BA) configurations. RMSE denotes root-mean-square error, and BBox denotes the reference-trajectory bounding-box diagonal.

Dataset	Evaluation State	RMSE (m)	Maximum Error (m)	RMSE/Path (%)	RMSE/BBox (%)
Mission 1	Baseline verified trajectory	0.211	0.699	0.361	2.225
Mission 1	Robust local BA, verified Sim(3)	0.080	0.163	0.137	0.843
Mission 2	Baseline verified trajectory	0.284	1.663	0.372	2.706
Mission 2	Robust local BA, verified Sim(3)	0.110	0.181	0.143	1.047
Mission 3	Baseline verified trajectory	0.292	1.161	0.281	1.895
Mission 3	Robust local BA, verified Sim(3)	0.127	0.184	0.122	0.824

The baseline trajectories had RMSE values of 0.211–0.292 m, whereas the robust local BA trajectories had values of 0.080–0.127 m. The descriptive reductions in the archived RMSE values were 62.1%, 61.3%, and 56.5% for missions 1, 2, and 3, respectively. Maximum errors decreased from 0.699–1.663 m to 0.163–0.184 m.

The baseline path-normalized RMSE values were 0.361%, 0.372%, and 0.281%; all were below 0.4% of the respective path length. The robust local BA values were 0.137%, 0.143%, and 0.122%. Their spread was 0.021 percentage points, indicating comparable relative accuracy across the three mission lengths.

The bounding-box-normalized RMSE showed the same trend: 2.225% to 0.843% for mission 1, 2.706% to 1.047% for mission 2, and 1.895% to 0.824% for mission 3. This metric accounts for the different spatial extents of the three trajectories.

Figure 4 shows the regenerated top-down route visualizations. To prioritize route readability and avoid over-interpreting alignment-sensitive baseline overlays, the panels show the stored comparison/reference trajectory and the robust local BA trajectory. The baseline trajectory is retained in the quantitative comparison in Table 3 and Figure 5. In all three missions, the robust local BA trajectory follows the stored comparison trajectory without an evident route-scale discontinuity. The different route topologies should also be considered when interpreting the values: mission 1 has a larger visible start–end separation than missions 2 and 3, whereas missions 2 and 3 are near-return trajectories. A near-return or loop-like path can provide stronger consistency checks, while a trajectory with larger start–end separation may expose accumulated drift more directly. Path-normalized RMSE reduces the influence of path length but does not remove topology-dependent differences.

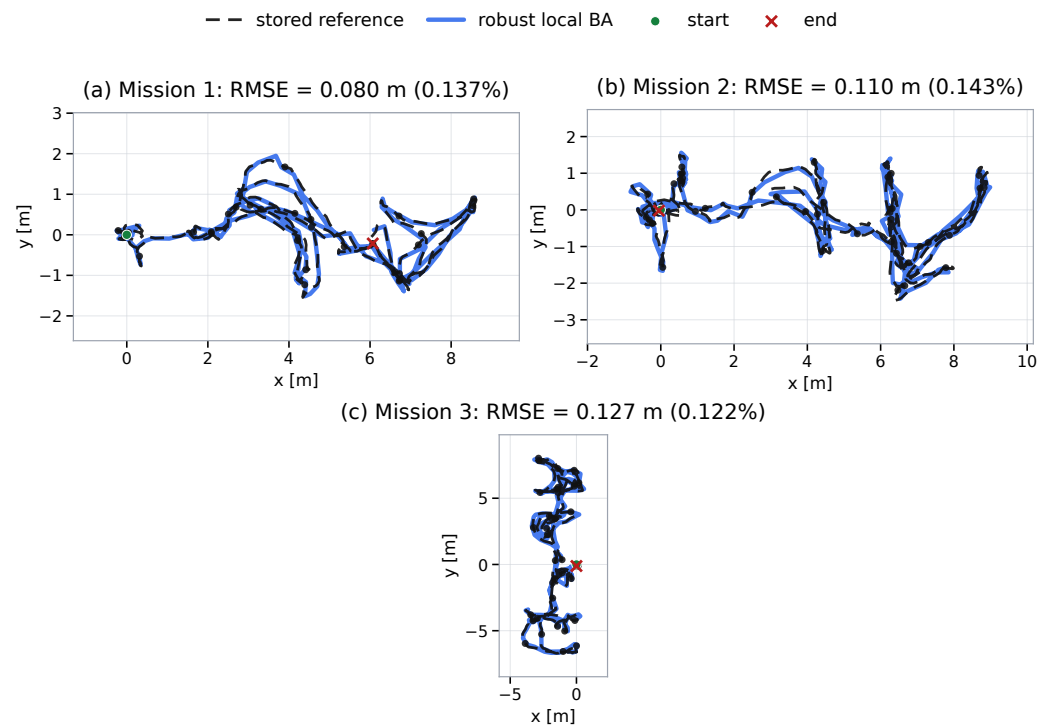


Figure 4. Top-down route visualization for NTNU Ballast Water Tank missions: (a) mission 1, (b) mission 2, and (c) mission 3. The black dashed curve with sparse markers denotes the stored comparison/reference trajectory, and the blue solid curve denotes the robust local bundle adjustment trajectory after the archived HBA top-index matching and Sim(3) alignment protocol. The reported robust RMSE values are 0.080 m, 0.110 m, and 0.127 m for missions 1–3, respectively, corresponding to 0.137%, 0.143%, and 0.122% of the reference path lengths. This figure is intended as a route-level visualization; baseline-versus-robust quantitative comparisons are reported in Table 3 and Figure 5. Baseline trajectories are not overlaid because the archived baseline and robust trajectories use the same evaluation protocol but different matched sample sets, which can invite over-interpretation as a strict single-transform ablation. Tank outlines are not overlaid because authoritative tank-boundary polygons or vessel CAD models were unavailable in the archived evaluation package. Axes are reported in metres.

Figure 5 summarizes the baseline-versus-robust quantitative comparison using the archived evaluation values.

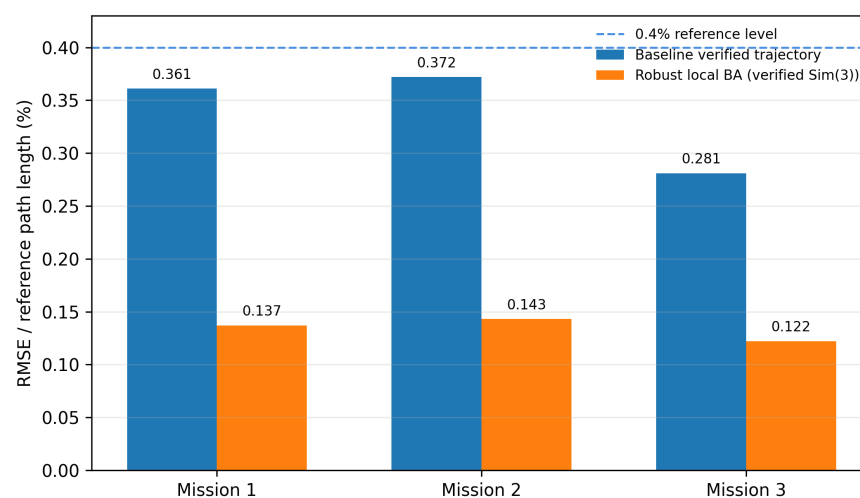


Figure 5. Path-normalized root-mean-square error (RMSE) for the baseline and robust local bundle adjustment configurations on NTNU Ballast Water Tank missions 1–3. The dashed reference indicates 0.4% of the corresponding reference path length.

3.3. Structural Behavior of the Kaczmarz Backend

The controlled row-access comparison showed that Kaczmarz did not yield the lowest linear residual. Its implementation nevertheless required only CSR row access, the right-hand side, the solution vector, row statistics, and sampling state. Table 4 reports a representative comparison, and Figure 6 shows the corresponding solver-side structural-memory estimates.

Table 4. Selected results from the controlled backend-structure comparison. CSR and CSC denote compressed sparse row and compressed sparse column, respectively; LSQR denotes least-squares QR; PCG denotes preconditioned conjugate gradient; and LDLT denotes an $\text{LDL}^\top$ factorization.

Backend	Storage Mode	Mean Residual	Mean Runtime (s)	Structural Memory (MB)	Working Set (MB)
Kaczmarz	CSR row-only	0.158	0.1210	0.075	0.062
LSQR	CSR and CSC transpose cache	0.011	0.0190	0.149	0.118
PCG	Explicit sparse $\mathbf{H} = \mathbf{A}^\top \mathbf{A}$	0.011	0.0047	0.177	0.094
Cholesky/LDLT	Explicit $\mathbf{H}$ and sparse factorization	0.011	0.0015	0.346	0.056

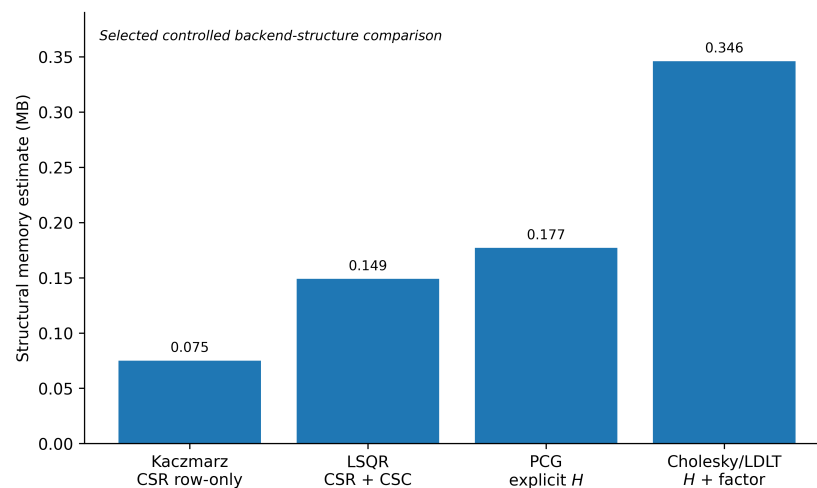


Figure 6. Solver-side structural-memory estimates for the Kaczmarz, LSQR, preconditioned conjugate-gradient (PCG), and Cholesky/ $\text{LDL}^\top$ backends in the controlled row-access comparison. These estimates are not equivalent to the full-pipeline peak resident set size (RSS).

Direct and Krylov-family methods achieved lower residuals and, in several controlled cases, shorter runtimes. Kaczmarz should therefore not be interpreted as a universal replacement for other graph solvers. In the representative case, its structural-memory estimate was 0.075 MB, compared with 0.149 MB for LSQR with a stored transpose cache, 0.177 MB for PCG with explicit $\mathbf{H}$, and 0.346 MB for the explicit- $\mathbf{H}$ Cholesky/ $\text{LDL}^\top$ configuration.

In the memory-budgeted streaming experiment, generating one Kaczmarz row at a time reduced the estimated working set to 0.044 MB, compared with 0.061 MB for chunked rows and 0.268 MB when the full CSR matrix was retained. These values quantify a solver-level working-set trade-off; they do not predict full-pipeline peak RSS. In the full SLAM system, peak RSS is dominated primarily by point-cloud storage, local maps, keyframe buffers, graph assembly, and runtime memory management, as shown by the operating-point results in Table 5.

Table 5. Representative full-pipeline operating policies obtained from the Park dataset. BA denotes bundle adjustment, RSS denotes resident set size, and RPE denotes relative pose error. The archived RPE RMSE is an origin-aligned audit metric using index-fallback pairing; it is included only to indicate whether additional memory reduced the archived trajectory-error proxy.

Operating Point	Principal Settings	Global BA State	Output Poses	Peak RSS (MB)	Runtime (s)	Retained RPE RMSE (m)
Memory-priority	Stride 7; leaf 0.8; 50,000-point cap	Disabled	57	165.1	6.585	8.467
Balanced	Stride 5; leaf 0.8; 50,000-point cap; 20-pose limit	Skipped by range guard	135	178.5	13.56	3.672
Map-density-priority	Stride 5; leaf 0.2; 100,000-point cap; 100-pose limit	Skipped by range guard	135	411.4	20.58	3.672

The broader solver-only benchmark similarly showed that LSQR/LSMR and Krylov methods were highly competitive in residual and runtime. Row-norm Kaczmarz produced a mean residual of 0.273 and a mean relative residual of 0.218, whereas the direct and Krylov-family methods produced mean residuals of approximately 0.0067 in the corresponding aggregate benchmark. In the retained Kaczmarz sensitivity audit, increasing the row-projection budget from 3 to 10 epochs reduced the mean solver-only residual from 0.274 to 0.109, while increasing the mean runtime from 0.016 to 0.051 s. These sensitivity values are solver-only residual/runtime curves and are not direct evidence of long-term map consistency. The result positions Kaczmarz as a memory-aware row-action option, not as a universally superior numerical solver.

3.4. System-Level Memory Operating Points

System-level controls were evaluated separately on the Park dataset, for which the archived runs covered multiple keyframe, point-cap, and global-optimization settings. These runs were used to profile memory and runtime under controlled operating-policy variations, not to establish a separate marine-domain accuracy benchmark. Retained origin-aligned audit metrics were nevertheless available for the three representative cases and are reported only as a trajectory-error proxy. Table 5 summarizes these operating policies.

Peak RSS increased from 165.1 MB in the memory-priority configuration to 411.4 MB in the map-density-priority configuration. The main contributors were map resolution, selected point count, keyframe density, and the permitted global-optimization range. The retained RPE RMSE decreased from 8.467 m in the memory-priority case to 3.672 m in the balanced case; however, the map-density-priority case did not improve this retained RPE RMSE relative to the balanced case despite increasing peak RSS from 178.5 to 411.4 MB. In the balanced and map-density-priority cases, the range guard skipped global BA because the pose count exceeded the configured limit, while the remaining pipeline still produced a trajectory. These cases are therefore range-controlled pipeline executions, not successful high-load global BA solves. Because the archived Park accuracy metric uses origin alignment with index-fallback pairing, it is reported as an audit metric rather than as a fully verified trajectory benchmark.

These results distinguish two complementary forms of memory control. The row-wise backend limits solver-side factorization storage, whereas the keyframe, map, and optimization-range policies limit the size of the problem presented to the solver. Both are required in a complete onboard architecture.

4. Discussion

4.1. Significance for Marine Inspection Robotics

The NTNU results show that trajectory error remained a small fraction of the traveled distance in the ballast-tank missions. The robust local BA trajectories had path-normalized RMSE values of 0.122–0.143%. Such normalized measures are more informative than absolute RMSE alone when mission lengths differ substantially.

For ship inspection, localization quality supports more than autonomous mobility. A consistent trajectory allows observations of corrosion, coating degradation, cracks, anodes, brackets, stiffeners, and weld regions to be registered within a three-dimensional structural reference frame. It also supports comparison across inspection campaigns and between observed geometry and design information. Large local discontinuities can otherwise duplicate structures, distort clearance estimates, or associate inspection records with incorrect locations.

The robust frontend targets failure modes that are characteristic of ship interiors. Residual weighting limits the influence of ambiguous matches; damping and step clamping bound updates in locally degenerate directions; and soft fallback preserves continuity when a local window lacks sufficient geometric information. Local-map compression additionally reduces redundant samples from broad planar surfaces while retaining informative structural boundaries.

The shape and topology of the tank can influence localization quality. Long planar bulkheads and repetitive stiffeners can reduce observability in particular motion directions, narrow openings can cause abrupt changes in LiDAR overlap, and loop-like route segments can provide stronger consistency checks than open trajectories. The present study reports mission-level results and path-normalized errors, but it does not isolate tank geometry as an independent experimental variable because tank-boundary polygons and vessel CAD models were not available in the archived evaluation exports.

4.2. Numerical Architecture and Operating-Policy Co-Design

The principal contribution is the co-design of numerical representation, solver access, frontend stabilization, and operating policy. Replacing a Cholesky solve with Kaczmarz while retaining unlimited keyframes and uncompressed point clouds would not resolve the end-to-end memory problem. Conversely, point-cloud compression alone would not prevent factorization storage from increasing as the pose graph becomes more densely connected.

The Kaczmarz backend offers three relevant system properties. First, it operates directly on whitened Jacobian rows. Second, its update does not require explicit storage of $\mathbf{A}^\top$, $\mathbf{H} = \mathbf{A}^\top \mathbf{A}$, or a Cholesky factor. Third, factor rows can be streamed, prioritized, down-weighted, clipped, or deferred. These properties support memory budgeting, incremental factor arrival, and robust row-selection policies.

The experiments also quantify the cost of this flexibility. Kaczmarz did not attain the residuals achieved by LSQR/LSMR, PCG, or direct factorization in the controlled benchmark. If the row-projection budget is too small, or if loop-closure constraints are weak, the higher residual floor can leave residual inconsistency in the global graph and may contribute to accumulated map distortion over longer missions. The archived NTNU missions did not show an evident route-scale discontinuity after robust local BA, but they do not constitute a long-duration closed-loop stress test of cumulative drift under incomplete batch convergence. The proposed architecture mitigates this risk by using robust local BA, conservative global-range control, and memory-aware map compression before global row projections, but it does not remove the numerical trade-off. Kaczmarz is therefore most appropriate when row-level controllability and bounded structural storage

are more important than obtaining the smallest linear residual in the shortest time. If the residual floor becomes an accuracy bottleneck, practical mitigation strategies include increasing the epoch budget, using confidence- or outlier-aware row selection, adopting factor-aligned block or mini-batch Kaczmarz updates, and applying periodic hybrid Krylov or direct refinement when memory permits.

This interpretation is consistent with recent resource-aware SLAM research, which treats algorithm and hardware constraints as a coupled design problem rather than optimizing accuracy in isolation [26].

4.3. Interpretation of Onboard Feasibility

The 8 GB Jetson Orin Nano represents a relevant memory and power class for a compact marine inspection robot. The reported experiments, however, were not an end-to-end deployment under a hardware-enforced 8 GB cap. The system-level RSS values should therefore be interpreted as measured pipeline operating points and design margins, not as evidence of real-time performance on every Jetson configuration.

The distinction between solver-side structural memory and full-pipeline peak RSS is essential. Solver-side memory isolates the data structures associated with $(\mathbf{A}, \mathbf{b})$ and the selected numerical method. Full-pipeline RSS also includes point-cloud loading, feature extraction, local mapping, graph construction, visualization or output buffers, memory allocators, and software libraries. Consequently, a lower solver-side estimate does not imply an equal reduction in process peak RSS when point-cloud storage dominates.

The results nevertheless support the architectural feasibility of onboard operation. The backend avoids factorization-related storage, while the surrounding policies regulate the number and size of keyframes, maps, and active factors. These mechanisms provide explicit control variables for trading trajectory quality against memory margin, runtime, power, and thermal limits.

4.4. Limitations and Future Work

Several limitations define the scope of the conclusions. First, the archived baseline and robust NTNU results were not generated under identical alignment conditions; the robust trajectories used verified Sim(3) alignment. Their numerical differences provide trajectory-level evidence but do not constitute a strictly controlled single-variable ablation. The alignment mismatch could artificially increase the baseline error relative to the robust local BA result, and the comparison is therefore treated as descriptive archived evidence rather than a definitive before–after ablation.

Second, the stored comparison trajectory used for the NTNU metrics is not an independently surveyed, metrology-grade ground truth. The RMSE values quantify deviation from the archived comparison trajectory under the recorded pairing and alignment procedure.

Third, the archived trajectory metrics are deterministic when recomputed from the same stored trajectories, but repeated end-to-end SLAM reruns were not archived.

Fourth, the trajectory results validate localization output but do not establish real-time closed-loop navigation, collision avoidance, or inspection coverage. A deployment study should additionally measure update rate, latency distribution, CPU and GPU utilization, thermal throttling, dropped sensor packets, and mission success on the target robot.

Fifth, Kaczmarz can converge slowly for ill-conditioned or strongly inconsistent systems, and its residual floor can exceed that of LSQR/LSMR or well-preconditioned Krylov methods. The proposed architecture does not remove this limitation.

Sixth, some intermediate point counts and compression ratios were not retained in the archived logs. Per-operating-point Park audit metrics were archived, but they use origin alignment with index-fallback pairing and are therefore not interpreted as fully verified

trajectory benchmarks. The study consequently separates raw points, selected inputs, final pose counts, solver-side structural memory, full-pipeline RSS, runtime, and retained audit metrics rather than merging them into a single accuracy–memory claim.

Seventh, the full-pipeline operating points were not measured under cgroup, container, or hardware-level memory enforcement. Future tests should impose reproducible 4, 8, and 16 GB limits and distinguish confirmed out-of-memory events from generic process termination.

Future work should develop adaptive solver selection based on graph topology, available memory, accuracy requirements, and time budget. LSQR or LSMR may be preferable when a low residual is required and transpose operations are affordable; PCG may be appropriate when an effective preconditioner is available; and Kaczmarz may be selected when row streaming or a strict working-set limit dominates. Factor-aligned block and mini-batch Kaczmarz variants should also be evaluated to improve the convergence–memory trade-off in ill-conditioned graph-SLAM systems. Long-duration embedded deployments should additionally consider memory fragmentation from repeated allocation and deallocation of point clouds and sparse solver buffers; memory pools or custom allocators may be required for robust operation. Finally, frontend confidence weights should be logged and propagated explicitly to global row selection, and hardware-capped experiments should record GPU usage, temperature, power consumption, and mission-completion statistics.

5. Conclusions

This study presented a memory-efficient graph SLAM architecture for autonomous inspection of ballast water tanks and other resource-constrained ship interiors. The architecture combines robust local hierarchical bundle adjustment with a CSR-based Kaczmarz backend, local-map compression, keyframe regulation, and global-optimization range control.

The frontend uses residual-adaptive weighting, damping, bounded translation and rotation updates, and soft fallback to suppress local divergence in repetitive or weakly constrained geometry. The backend operates directly on whitened factor rows and avoids explicit construction of the approximate Hessian, a Cholesky factor, and a transpose cache.

On NTNU Ballast Water Tank missions 1–3, baseline path-normalized RMSE values were 0.281–0.372%. The robust local BA trajectories yielded RMSE values of 0.080, 0.110, and 0.127 m, corresponding to 0.137%, 0.143%, and 0.122% of the respective path lengths.

The solver comparisons showed that Kaczmarz was not residual- or runtime-optimal relative to LSQR/LSMR, Krylov methods, or direct factorization. Its practical value lies in row-wise controllability, factorization-free storage, and compatibility with streaming and memory-budgeted operation. The contribution is therefore a numerical-architecture and operating-policy co-design for onboard marine robotics, not a new Kaczmarz variant or a claim of universal solver superiority.

Author Contributions: Conceptualization, S.C. and T.-w.K.; methodology, S.C. and W.Y.; software, S.C.; validation, S.C. and W.Y.; formal analysis, S.C.; investigation, S.C.; data curation, S.C.; writing—original draft preparation, S.C.; writing—review and editing, S.C., W.Y. and T.-w.K.; visualization, S.C.; resources, T.-w.K.; supervision, T.-w.K.; project administration, T.-w.K.; funding acquisition, T.-w.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Global Basic Research Laboratory (Advanced) Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Science and ICT (MSIT), grant number 00406127 (Mathematical-based theoretical research and technology development of vision models for generative AI).

Data Availability Statement: The associated inspection platform and representative field deployments are described by Dharmadhikari et al. [1]. The third-party NTNU Ballast Water Tank dataset analyzed in this study is openly available from the NTNU Autonomous Robots Laboratory through Hugging Face Datasets [25]. Repository documentation and mission metadata are available at https://github.com/ntnu-arl/ballast_water_tank_dataset (accessed on 22 June 2026). Retained processed trajectory figures, trajectory coordinate files used for the reported NTNU recomputation, solver configurations, benchmark logs, and revision-stage recomputation scripts are available from the corresponding author upon reasonable request.

Acknowledgments: The authors acknowledge the NTNU Autonomous Robots Laboratory for making the ballast water tank dataset publicly available.

Conflicts of Interest: W.Y. is employed by Avikus Co., Ltd. The remaining authors declare no conflicts of interest. The funder had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Dharmadhikari, M.; De Petris, P.; Kulkarni, M.; Khedekar, N.; Nguyen, H.; Stene, A.E.; Sjøvold, E.; Solheim, K.; Gussiaas, B.; Alexis, K. Autonomous Exploration and General Visual Inspection of Ship Ballast Water Tanks Using Aerial Robots. In Proceedings of the 2023 21st International Conference on Advanced Robotics, Abu Dhabi, United Arab Emirates, 5–8 December 2023; pp. 409–416. [\[CrossRef\]](#)
2. Poggi, L.; Gaggero, T.; Gaiotti, M.; Ravina, E.; Rizzo, C.M. Recent Developments in Remote Inspections of Ship Structures. *Int. J. Nav. Archit. Ocean Eng.* **2020**, *12*, 881–891. [\[CrossRef\]](#)
3. Christensen, L.; Fischer, N.; Kroffke, S.; Lemburg, J.; Ahlers, R. Cost-Effective Autonomous Robots for Ballast Water Tank Inspection. *J. Ship Prod. Des.* **2011**, *27*, 127–136.
4. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Trans. Robot.* **2016**, *32*, 1309–1332. [\[CrossRef\]](#)
5. Durrant-Whyte, H.; Bailey, T. Simultaneous Localization and Mapping: Part I. *IEEE Robot. Autom. Mag.* **2006**, *13*, 99–110. [\[CrossRef\]](#)
6. Zhang, J.; Singh, S. LOAM: Lidar Odometry and Mapping in Real-time. In Proceedings of the Robotics: Science and Systems, Berkeley, CA, USA, 12–16 July 2014. [\[CrossRef\]](#)
7. Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; Rus, D. LIO-SAM: Tightly-Coupled Lidar Inertial Odometry via Smoothing and Mapping. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Virtual Conference, 24 October 2020–24 January 2021; pp. 5135–5142. [\[CrossRef\]](#)
8. Liu, X.; Liu, Z.; Kong, F.; Zhang, F. Large-Scale LiDAR Consistent Mapping Using Hierarchical LiDAR Bundle Adjustment. *IEEE Robot. Autom. Lett.* **2023**, *8*, 1523–1530. [\[CrossRef\]](#)
9. Dellaert, F.; Kaess, M. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *Int. J. Robot. Res.* **2006**, *25*, 1181–1203. [\[CrossRef\]](#)
10. Kaess, M.; Ranganathan, A.; Dellaert, F. iSAM: Incremental Smoothing and Mapping. *IEEE Trans. Robot.* **2008**, *24*, 1365–1378. [\[CrossRef\]](#)
11. Kaess, M.; Johannsson, H.; Roberts, R.; Ila, V.; Leonard, J.J.; Dellaert, F. iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree. *Int. J. Robot. Res.* **2012**, *31*, 216–235. [\[CrossRef\]](#)
12. Kümmerle, R.; Grisetti, G.; Strasdat, H.; Konolige, K.; Burgard, W. g2o: A General Framework for Graph Optimization. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 3607–3613. [\[CrossRef\]](#)
13. Davis, T.A. *Direct Methods for Sparse Linear Systems*; SIAM: Philadelphia, PA, USA, 2006. [\[CrossRef\]](#)
14. Chen, Y.; Davis, T.A.; Hager, W.W.; Rajamanickam, S. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* **2008**, *35*, 1–14. [\[CrossRef\]](#)
15. George, A.; Liu, J.W.H. *Computer Solution of Large Sparse Positive Definite Systems*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1981; ISBN 978-0-13-165274-3.
16. NVIDIA Corporation. Jetson Orin Nano Super Developer Kit: Technical Specifications. Available online: <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/> (accessed on 22 June 2026).
17. Björck, Å. *Numerical Methods for Least Squares Problems*; SIAM: Philadelphia, PA, USA, 1996. [\[CrossRef\]](#)
18. Paige, C.C.; Saunders, M.A. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares. *ACM Trans. Math. Softw.* **1982**, *8*, 43–71. [\[CrossRef\]](#)

19. Fong, D.C.-L.; Saunders, M.A. LSMR: An Iterative Algorithm for Sparse Least-Squares Problems. *SIAM J. Sci. Comput.* **2011**, *33*, 2950–2971. [[CrossRef](#)]
20. Saad, Y. *Iterative Methods for Sparse Linear Systems*, 2nd ed.; SIAM: Philadelphia, PA, USA, 2003. [[CrossRef](#)]
21. Kaczmarz, S. Angenäherte Auflösung von Systemen linearer Gleichungen. *Bull. Int. Acad. Pol. Sci. Lett. Cl. Sci. Math. Nat. Sér. A* **1937**, *35*, 355–357.
22. Strohmer, T.; Vershynin, R. A Randomized Kaczmarz Algorithm with Exponential Convergence. *J. Fourier Anal. Appl.* **2009**, *15*, 262–278. [[CrossRef](#)]
23. Needell, D.; Srebro, N.; Ward, R. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz Algorithm. *Math. Program.* **2016**, *155*, 549–573. [[CrossRef](#)]
24. Needell, D.; Tropp, J.A. Paved with Good Intentions: Analysis of a Randomized Block Kaczmarz Method. *Linear Algebra Appl.* **2014**, *441*, 199–221. [[CrossRef](#)]
25. NTNU Autonomous Robots Laboratory. Ballast Water Tank Dataset; Hugging Face Datasets. Available online: https://huggingface.co/datasets/ntnu-arl/ballast_water_tank_dataset (accessed on 22 June 2026).
26. Kim, S.; Hsiao, R.; Nikolić, B.; Demmel, J.; Shao, Y.S. SuperNoVA: Algorithm–Hardware Co-Design for Resource-Aware SLAM. In Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Rotterdam, The Netherlands, 30 March–3 April 2025; pp. 1035–1051. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.