EasyColor: Reflectivity Assisted Dense Point Cloud RGB Colorizing Without Accurate Time Synchronization and Extrinsic Calibration

***Abstract*— A precise RGB-colored point cloud map serves as a reliable data foundation and a compact multi-sensor representation for tasks such as 3D perception, localization, and navigation. Therefore, efficient and accurate alignment between camera and LiDAR is essential. However, most existing color mapping methods require precise time synchronization and well-calibrated extrinsic value, and these colorizing frameworks are often hard to use. In this paper, we propose EasyColor, a precise colorization method that jointly leverages LiDAR reflectivity and visual constraint, effectively removing the need for time synchronization and extrinsic calibration between the two sensors. Our method follows a plug-and-play framework. Given an image sequence and a pre-built point cloud map, we first generate 2D reflectivity image sequence by exploiting variations in LiDAR reflectivity intensity. A cross-domain matching network, combined with a Perspective-n-Point (PnP) solver, is employed to align the reflectivity and camera image sequence. To further optimize camera poses and ensure photometric consistency, we fuse visual epipolar geometric constraint and PnP constraint. Our joint optimization produces both accurate and photometrically consistent colored map, along with optimized camera poses. Extensive experiments demonstrate that EasyColor achieves precise point cloud colorization without requiring synchronized LiDAR and camera data and extrinsic values. To facilitate future research, we will release EasyColor and dataset after review stage.**

I. INTRODUCTION

In the fields of autonomous driving and robotics, RGB-colored point cloud maps serve as reliable data foundations and compact multi-sensor representations, which are essential for tasks such as 3D perception, localization, and navigation [1]–[3]. In the past two decades, dense SLAM methods based on monocular cameras can directly generate colored maps; however, due to the fragile robustness and limited accuracy of visual odometry, the resulting colored maps are often imprecise and inconsistent [4], [5]. In recent years, as LiDAR can directly provide accurate environmental depth information, LiDAR-based point cloud map generation methods have gradually become mainstream [6], [7]. Therefore, integrating color texture information from camera with LiDAR point cloud to produce accurate colored point cloud maps has emerged as a promising research direction.

Therefore, to efficiently fuse camera and LiDAR data, some works attempt to couple visual sensors with existing LiDAR mapping systems, achieving real-time colored mobile mapping while improving odometry accuracy and robustness [1], [8]. However, these methods typically require precise time synchronization and well-calibrated extrinsic value to ensure accurate alignment between camera and LiDAR data (since the rates of these two sensors often differ significantly, making accurate time synchronization more difficult). To avoid the reliance on precise time synchronization, several studies have explored offline point cloud coloring in the post-processing stage by aligning image sequences with dense point cloud map. Colmap-PCD [9] obtains a coarse 3D point cloud via initial visual odometry and establishes point-plane feature associations with the LiDAR point cloud map for data fusion. Koide [10] generates a 2D image based on LiDAR reflectivity and performs point-to-point matching with the camera image to achieve more robust alignment. In addition to the aforementioned feature-based methods, OmniColor [11] generates high-quality colored point cloud maps by jointly optimizing visual odometry and LiDAR odometry. However, these methods typically require reliable initial camera poses. Although Koide leverages LiDAR reflectivity information to achieve point-to-point camera-LiDAR alignment, it does not consider visual constraints between image sequences. Meanwhile, motion-based methods (such as OmniColor) do not consider feature correspondences, making it difficult to achieve high-level accuracy, and the fragile visual odometry introduces noise and inconsistency to the colorization results.

In this paper, we propose EasyColor, a precise and robust point cloud colorization method that jointly utilizes cross-domain reflectivity alignment and intra-frame visual constraint, effectively removing the need for time synchronization and extrinsic calibration. Our method follows a plug-and-play manner. Given an image sequence and a pre-built point cloud map, we first generate 2D reflectivity image sequence by exploiting variations in LiDAR intensity. A cross-domain matching network, combined with a Perspective-n-Point (PnP) solver, is employed to align the reflectivity image with the image sequence. To further optimize the camera poses and ensure photometric consistency, we introduce visual epipolar geometric constraints by leveraging the photometric continuity between adjacent camera frames. Our joint optimization produces both accurate and photometrically consistent colored point cloud map. The main contributions of our work are summarized as follows:

- We propose EasyColor, a plug-and-play framework that only need pre-built point cloud map and image sequence to achieve accurate point cloud colorization without requiring time synchronization and extrinsic calibration.
- We propose an effective joint optimization method that utilizes cross-domain reflectivity alignment and intra-frame visual constraints, significantly improving the consistency of color mapping and camera pose accuracy.
- Extensive experiments on both public benchmarks and

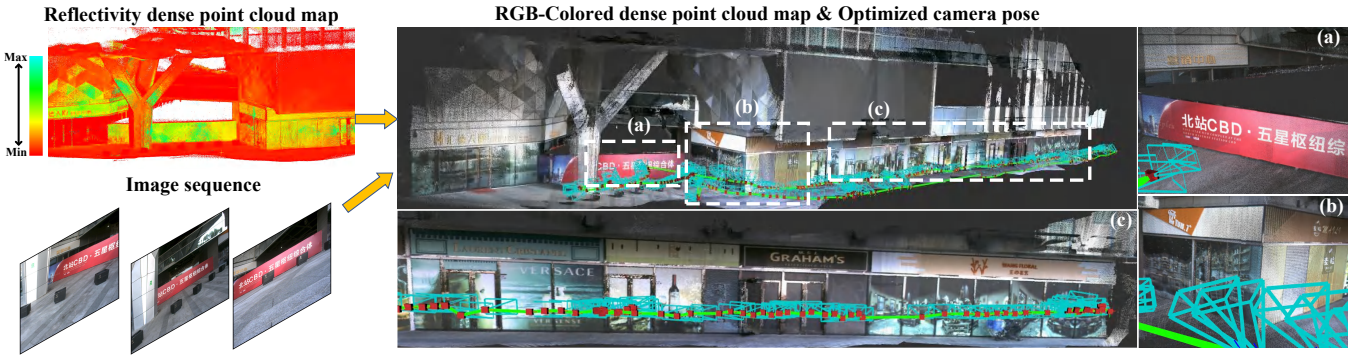

Fig. 1. Colorizing results on "CBD_Building_03" sequence of FastLIVO2 dataset [1]. Our proposed plug-and-play framework for point cloud colorizing requires only reflectivity point cloud map and image sequence as input.

our self-recorded datasets demonstrate that EasyColor achieves superior colorization accuracy and consistency compared to state-of-the-art methods, while being compatible with most LiDAR-camera sensor suites.

II. RELATED WORK

Here, we briefly introduce two types of camera-LiDAR calibration methods, along with their implementation details and respective advantages and disadvantages.

A. Feature-based Camera-LiDAR Alignment

The simplest approach to aligning data from two fundamentally different sensor modalities is to detect and associate common target features present in both types of data. In recent years, several excellent extrinsic calibration works have achieved pixel-level camera-LiDAR alignment by extracting and matching edge features from point cloud and image [12], [13]. These methods have been widely adopted as mature calibration modules in the preprocessing stages of autonomous systems. Colmap-PCD [9] is similar to our method, as it also takes point cloud map and coarse initial camera poses as input, and derives 2D-3D line correspondences to iteratively minimize the re-projection error for accurate camera pose estimation. However, relying only on geometric edge features makes such approaches unreliable in unstructured or textureless environments (such as tunnels, corridors, and plain walls), limiting their applicability for online alignment in arbitrary scenarios. Moreover, the reliance on accurate initial camera poses makes the above edge-based methods less user-friendly and harder to apply in practice.

To remove the reliance on accurate initial poses and enhance the robustness of alignment algorithms in degraded environments, some studies have explored using mutual information between LiDAR and camera data, such as the reflectivity intensity from LiDAR and illumination textures from camera images. Pandey [14], [15] improves the robustness of LiDAR-camera alignment by maximizing the mutual information between LiDAR reflectivity and image photometric features. Koide [10] generates 2D image-like data directly from LiDAR reflectivity information and performs point-to-point matching with RGB images to achieve robust registration. However, these methods only consider the constraint between single frames of the two sensors and do not incorporate epipolar constraints across consecutive frames in the image sequence.

B. Motion-based Camera-LiDAR Alignment

Recent advances in motion-based camera-LiDAR alignment have enabled the effective integration of visual and geometric information for high-quality point cloud colorization. Ve et al. [16] proposed an offline, plug-and-play colorization approach, but it obtains camera poses by interpolation, which results in low accuracy. OmniColor [11] achieves precise point cloud colorization by jointly optimizing the camera and LiDAR poses based on their co-visible regions. However, both of these methods lack feature-level alignment between the two sensors. SLAM methods such as FastLIVO2 [1] and R3LIVE [17] integrate camera, LiDAR, and IMU sensors to enhance localization accuracy and robustness, and generate colored point clouds as a byproduct. However, these methods cannot achieve precise colorization results without strict camera-LiDAR time synchronization. To enhance the alignment accuracy between the camera and LiDAR by combining cross-domain feature alignment with intra-modal inter-frame constraints, we draw inspiration from [10] and [11]. Specifically, we construct reflectivity-based cross-domain constraints to achieve feature alignment between modalities, while simultaneously establishing epipolar constraints between consecutive camera frames to ensure geometric consistency throughout the alignment process within the image sequence.

III. PROBLEM DEFINITION

In this section we define point cloud colorization and introduce notation. Let $\mathcal{L} = \{({}^{L}\mathbf{l}_m, r_m)\}_{m=1}^{N_L}$ be the input LiDAR point cloud, where ${}^{L}\mathbf{l}_m \in \mathbb{R}^3$ is the 3D coordinate and $r_m \in \mathbb{R}$ is the reflectivity measured by the sensor via the official ROS driver (no preprocessing). Let $\mathcal{I} = \{\mathbf{I}_i\}_{i=1}^{N_I}$ be the input image sequence, and let ${}^{W}\mathbf{T}_{C_i} = ({}^{W}\mathbf{R}_{C_i}, {}^{W}\mathbf{t}_{C_i}) \in SE(3)$ be the camera poses to estimate. We assume the scene is static during acquisition.

The problem is defined as follows: Given the input image sequence $\mathcal{I}$, the LiDAR point cloud map $\mathcal{L}$ (a general SLAM algorithm is sufficient to meet the accuracy requirements for the point cloud map), and the rough LiDAR–camera extrinsic value ${}^{C}\hat{\mathbf{T}}_{L}$, our goal is to estimate the optimal

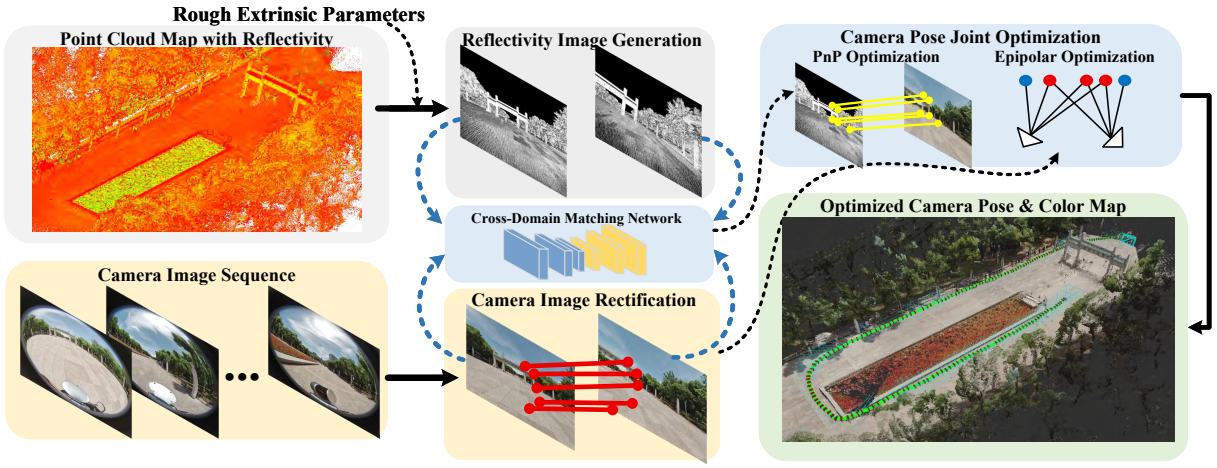

Fig. 2. Overview of the EasyColor framework. Given a pre-built point cloud map and an image sequence, we can get optimized camera poses and colorized point cloud map. The main technical modules include reflectivity image generation, cross-domain image matching, camera pose joint optimization, and point cloud colorization.

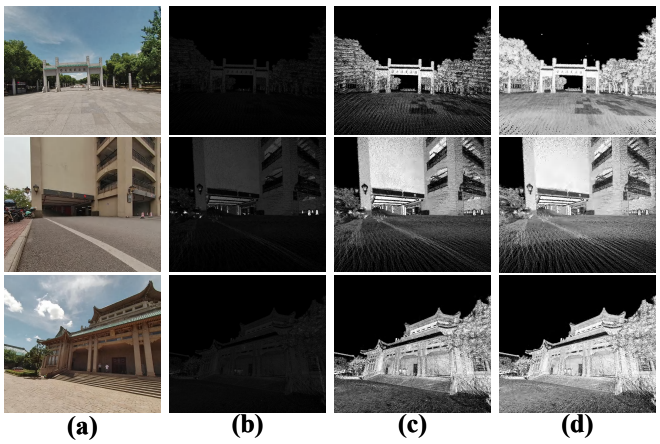

Fig. 3. Details of reflectivity image generation. (a) Camera image of the scene. (b) Original reflectivity image, (c) Reflectivity image after histogram equalization. (d) Reflectivity image after gamma correction.

camera poses ${}^{W}\mathbf{T}_{C_i}$ and colorize the point cloud map $\mathcal{L}$ with the corresponding RGB information from the camera image sequence $\mathcal{I}$. The camera poses allow us to project 3D points of the dense LiDAR point cloud map onto the 2D image plane using camera intrinsic parameters $\mathbf{K}$. Unlike LiDAR–camera odometry methods that require temporally and spatially synchronized image and LiDAR sequences, EasyColor works with a pre-built LiDAR map and an image sequence and does not require strict time synchronization or highly accurate extrinsics.

IV. Our EasyColor Framework

Here, we will discuss framework details of our proposed EasyColor and the pipeline is shown in Fig. 2. The inputs include a pre-built point cloud map, image sequence, and initial camera pose. The main technical modules comprise point cloud reflectivity image generation, cross-domain image matching, relative pose joint optimization, and point cloud colorization.

A. Reflectivity Image Generation

To generate 2D reflectivity image sequences for robust LiDAR-camera association, we project the LiDAR points $\boldsymbol{l}_m$ onto the image plane according to Eq. 1.

$$\mathbf{p}_m = \begin{cases} \Pi\left(\mathbf{K},\, {}^{C}\hat{\mathbf{T}}_{L}\, {}^{L}\boldsymbol{l}_m\right), & i = 1, 2 \\ \Pi\left(\mathbf{K},\, {}^{C_{i-1}}\bar{\mathbf{T}}_{W}\, {}^{L}\boldsymbol{l}_m\right), & i > 2 \end{cases} \tag{1}$$

where $\mathbf{p}_m \in \mathbb{R}^2$ is the pixel coordinate of the m-th LiDAR point in the corresponding image plane $\mathbf{P}_i$, and $\Pi(\mathbf{K}, \cdot)$ is the pinhole projection function (using camera intrinsic matrix $\mathbf{K}$) that maps the 3D point in the LiDAR coordinate system to the 2D pixel coordinate in the image plane. EasyColor processes two image frames at a time. Since LiDAR and camera are typically rigidly connected, for the initial frames $\mathbf{I}_1$ and $\mathbf{I}_2$, we can directly obtain the reflectivity images $\mathbf{P}_1$ and $\mathbf{P}_2$ using rough extrinsic value. For subsequent image frames, we can generate the reflectivity image using the optimized camera pose from the previous frame ${}^{W}\bar{\mathbf{T}}_{C_{i-1}}$.To avoid multiple LiDAR points being projected onto the same pixel during projection, we retain only the LiDAR point with the minimum depth for each pixel.

After projection, we obtain a reflectivity image sequence $\mathcal{P} = \{\mathbf{P}_i | i = 1...N_I\}$ with the same resolution and sequence length as the camera images. As shown in Fig. 3, the raw reflectivity images may contain holes and noise, and often exhibit low contrast. To address this, we enhance the images using histogram equalization and gamma correction ($\gamma = 0.5$), which significantly improves their quality and provides a reliable basis for subsequent LiDAR-camera feature association, as shown in Fig. 3.

B. Cross-Domain Image Matching

Benefiting from the generation of high-quality reflectivity image sequences (details in Sec. IV-A), we do not need to directly perform feature association between camera images and point cloud, which are two modalities with significant differences. Instead, we establish reliable LiDAR-camera association pairs by indirectly associating the reflectivity

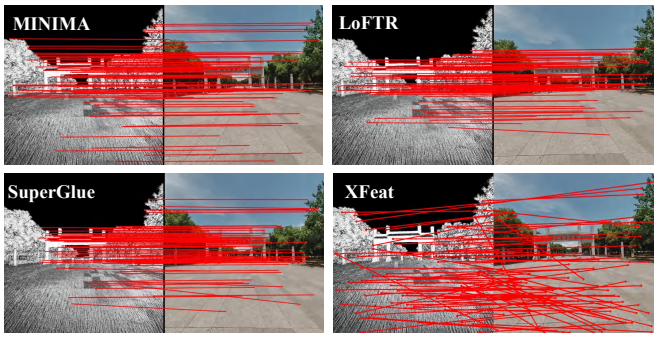

Fig. 4. Comparison of different cross-domain image matching methods.

and camera image sequence. Although a domain gap still exists, since both are 2D images, we can leverage existing cross-domain image matching networks to achieve robust feature matching. In this work, we compare several mainstream image matching methods, including SuperGlue [18], LoFTR [19], XFeat [20], and MINIMA [21]. We evaluated the performance of these matching methods on our collected dataset, and the results are shown in Fig. 4.

We find that MINIMA [21] provides the best trade-off between accuracy and robustness, yielding reliable matches even when features are visually ambiguous. Therefore, we adopt MINIMA for cross-domain matching. During reflectivity image generation we retain a index map so that 2D-2D matches can be converted into 2D–3D correspondences.

C. Camera Pose Joint Optimization

After extracting 2D–3D correspondences, PnP can directly provides an accurate estimate of camera–LiDAR relative pose. However, relying solely on PnP often yields poses that are spatially smooth but insufficient for accurate colorization, leading to subtle color inconsistencies across frames. To address this, EasyColor performs a joint optimization that integrates camera–LiDAR PnP constraints with inter-frame epipolar constraints. The combined optimization formulation further regularizes photometric consistency and produces more coherent colored point cloud.

LiDAR–camera PnP constraints: Given the 2D–3D correspondences $\{(\mathbf{u}_j^i, \mathbf{X}_j^i)\}_{j=1}^{N_c^i}$ for frame i and $\{(\mathbf{u}_k^{i+1}, \mathbf{X}_k^{i+1})\}_{k=1}^{N_c^{i+1}}$ for frame $i+1$ obtained from cross-domain matching, where $\mathbf{X}_\bullet^{(\cdot)} \in \mathbb{R}^3$ denotes a 3D point in the world coordinate system W, and $\mathbf{u}_\bullet^{(\cdot)} \in \mathbb{R}^2$ is its corresponding image coordinate. Let the two camera poses be ${}^{W}\mathbf{T}_{C_i}, {}^{W}\mathbf{T}_{C_{i+1}} \in SE(3)$, and denote their inverses as ${}^{C_i}\mathbf{T}_W = ({}^{W}\mathbf{T}_{C_i})^{-1}$, ${}^{C_{i+1}}\mathbf{T}_W = ({}^{W}\mathbf{T}_{C_{i+1}})^{-1}$. The PnP residuals with intrinsic matrix $\mathbf{K}$ are defined as:

$$\begin{aligned}\mathcal{E}_{\text{pnp}}^{i} &= \sum_{j=1}^{N_c^i} \left\| \mathbf{u}_j^i - \Pi\big(\mathbf{K},\, {}^{C_i}\mathbf{T}_W\, \tilde{\mathbf{X}}_j^i\big) \right\|_{\mathbf{\Sigma}_{\mathcal{E}_{\text{pnp}}^i}}^2 \\ \mathcal{E}_{\text{pnp}}^{i+1} &= \sum_{k=1}^{N_c^{i+1}} \left\| \mathbf{u}_k^{i+1} - \Pi\big(\mathbf{K},\, {}^{C_{i+1}}\mathbf{T}_W\, \tilde{\mathbf{X}}_k^{i+1}\big) \right\|_{\mathbf{\Sigma}_{\mathcal{E}_{\text{pnp}}^{i+1}}}^2\end{aligned} \tag{2}$$

where $\tilde{\mathbf{X}} = [\mathbf{X}^\top, 1]^\top$ is the homogeneous 3D point, and $\|\mathbf{e}\|_{\mathbf{\Sigma}}^2 = \mathbf{e}^\top \mathbf{\Sigma}^{-1} \mathbf{e}$.

Relative rotation from epipolar geometry: To ensure geometric consistency between consecutive frames, we estimate the essential matrix $\mathbf{E}_{i,i+1}$ from visual feature correspondences $\{(\mathbf{u}_l^i, \mathbf{u}_l^{i+1})\}_{l=1}^{N_v}$. Using normalized coordinates $\tilde{\mathbf{x}}_l^i = \mathbf{K}^{-1}\tilde{\mathbf{u}}_l^i$, $\tilde{\mathbf{x}}_l^{i+1} = \mathbf{K}^{-1}\tilde{\mathbf{u}}_l^{i+1}$, we have:

$$\tilde{\mathbf{x}}_l^{i+1\top}\, \mathbf{E}_{i,i+1}\, \tilde{\mathbf{x}}_l^i = 0, \qquad \mathbf{E}_{i,i+1} = [\mathbf{t}_{\text{rel}}]_\times\, \mathbf{R}_{\text{rel}} \tag{3}$$

where decomposition yields the relative rotation $\mathbf{R}_{\text{rel}} \in SO(3)$ and the translation direction $\mathbf{t}_{\text{rel}} \in \mathbb{S}^2$. Since the translation is only determined up to scale, we only use $\mathbf{R}_{\text{rel}}$ as a reliable constraint. Let the relative rotation implied by the two global poses be ${}^{C_{i+1}}\mathbf{R}_{C_i} = {}^{W}\mathbf{R}_{C_{i+1}}^\top\, {}^{W}\mathbf{R}_{C_i}$. The rotation consistency error is then defined as:

$$\mathcal{E}_{\text{rot}} = \left\| \text{Log}\big(\mathbf{R}_{\text{rel}}^\top\, {}^{C_{i+1}}\mathbf{R}_{C_i}\big) \right\|_{\mathbf{\Sigma}_{\mathcal{E}_{\text{rot}}}}^2 \tag{4}$$

where $\text{Log} : SO(3) \to \mathbb{R}^3$ denotes the Lie algebra logarithm mapping, measuring the rotational discrepancy as 3D vector.

Overall objective: The joint optimization problem is formulated as:

$$\min_{\boldsymbol{\xi}_i, \boldsymbol{\xi}_{i+1}} \; \mathcal{E}_{\text{pnp}}^i + \mathcal{E}_{\text{pnp}}^{i+1} + \lambda\, \mathcal{E}_{\text{rot}} \tag{5}$$

where $\boldsymbol{\xi}_i, \boldsymbol{\xi}_{i+1} \in \mathbb{R}^6$ are the $SE(3)$ Lie algebra parameterizations of ${}^{W}\mathbf{T}_{C_i}$, ${}^{W}\mathbf{T}_{C_{i+1}}$, and $\lambda > 0$ is a weighting factor. The optimization is solved using the Levenberg–Marquardt algorithm. The covariance matrices ($\mathbf{\Sigma}_{\mathcal{E}_{\text{pnp}}^i}$, $\mathbf{\Sigma}_{\mathcal{E}_{\text{pnp}}^{i+1}}$, $\mathbf{\Sigma}_{\mathcal{E}_{\text{rot}}}$) are all set to identity matrices for simplicity. Since different residual terms have different numerical scales, λ is used to balance their relative contributions. In practice, we set $\lambda = 1$, which empirically balances the pixel-level PnP reprojection errors (typically 0.01–0.1 pixel) and the Lie-algebra rotation discrepancy (typically 0.01–0.1 rad).

This joint optimization strategy ensures that the estimated camera poses are consistent with LiDAR–camera correspondences while also preserving inter-frame visual geometry.

Keyframe selection mechanism: To avoid the influence of potential outliers in any stage on the final optimization results, we designed a keyframe selection mechanism, ensuring that only qualified frame pairs are sent to the joint optimization module and subsequent point cloud colorization module. We score the reliability of a candidate pair $(i, i{+}1)$ by the composite inlier ratio.

$$S_{i,i+1} \;=\; \frac{N_{inlier}^i + N_{inlier}^{i+1}}{N_c^i + N_c^{i+1}} \tag{6}$$

where $N_{inlier}^{(\cdot)}$ is the number of inliers returned by PnP-RANSAC and $N_c^{(\cdot)}$ is the total number of 2D–3D correspondences for each frame. The pair is accepted as a keyframe pair iff $S_{i,i+1} \geq \tau_s$, where $\tau_s \in [0, 1]$ is a predefined reliability threshold (here we set it to 0.6).

D. Point Cloud Colorizing

After obtaining the optimized camera poses, we can project the LiDAR points onto the image plane of each camera frame to obtain the corresponding pixel coordinates. However, due to the different physical mechanisms of LiDAR and camera, direct 3D-to-2D projection can result in

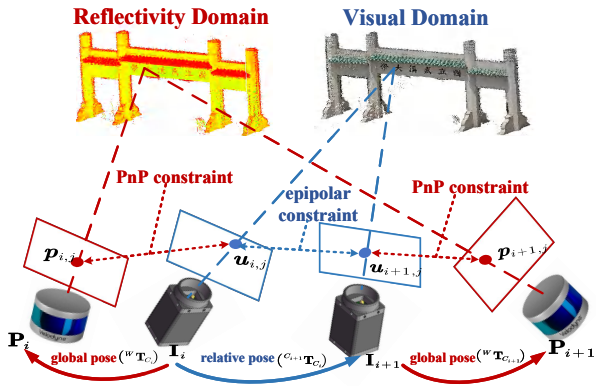

Fig. 5. Details of our proposed camera pose joint optimization framework.

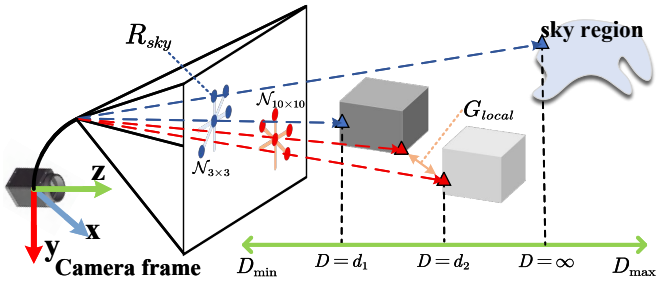

Fig. 6. Illustration of two types of unreliable edge region detection.

multiple LiDAR points at different depths corresponding to the same pixel, which leads to color bleeding. Additionally, due to the limited resolution of camera, LiDAR edge points with depth discontinuities can easily be incorrectly projected to inappropriate pixel coordinates (even when camera poses are highly accurate). To address these issues, we propose two strategies: color bleeding detection and unreliable edge region removal. The details are as follows:

Color bleeding detection: Given the optimized camera poses, we project the LiDAR points $\{{}^{L}\mathbf{l}_m\}_{m=1}^{N_L}$ onto the image plane to obtain a set of projected points $\mathcal{P} = \{(\mathbf{p}_m, d_m, \text{idx}_m)\}_{m=1}^{N}$ where $\mathbf{p}_m = (u_m, v_m) \in \mathbb{R}^2$ is the pixel coordinate obtained through $\mathbf{p}_m = \mathbf{\Pi}(\mathbf{K}^W\bar{\mathbf{T}}_{C_i}{}^L\mathbf{l}_m)$, $d_m \in \mathbb{R}^+$ is the depth in camera coordinates, and idx_m is the original point index in the LiDAR map.

We define a neighborhood function around each projected point. For a given projection location $\mathbf{p}$ with radius r, the neighborhood is defined as $N(p, r) = \{(u', v') \in \mathbb{Z}^2 \mid |u' - u| \le r,\ |v' - v| \le r\}$. For each pixel location $\mathbf{q} = (u', v')$, we maintain a depth buffer $D : \mathbb{Z}^2 \to \mathbb{R}^+$ and point index buffer $I : \mathbb{Z}^2 \to \mathbb{N}$. To ensure computational efficiency, we employ a block-wise parallel processing strategy where the image is partitioned into blocks $\mathcal{B} = \{B_k\}_{k=1}^{K}$. For each projected point $\mathbf{p}_m$ and each block B_k, we update the depth buffer only for pixels in the intersection region $\mathcal{R}_k(\mathbf{p}_m) = B_k \cap \mathcal{N}(\mathbf{p}_m, r)$. The depth buffer update follows:

$$D(\mathbf{q}) = \min(D(\mathbf{q}), d_m) \quad \text{for } \mathbf{q} \in \mathcal{R}_k(\mathbf{p}_m) \tag{7}$$

When a closer point is found, we also update the point index buffer $I(\mathbf{q}) = \text{idx}_m$. This ensures that each pixel receives color information from the geometrically closest LiDAR point, preventing color bleeding while maintaining computational efficiency through parallel block processing.

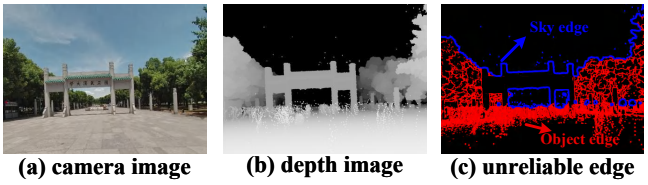

Fig. 7. Edge detection results. (a) Camera image. (b) Depth image generated by EasyColor. (c) Detected unreliable edge regions (blue line is sky edge, red line is object edge).

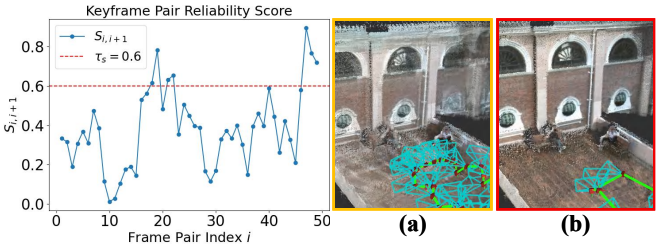

Fig. 8. The left figure shows reliable scores calculation. (a) shows the results without keyframe optimization. (b) shows the optimized result.

Unreliable edge region removal: For each pixel location $\mathbf{q} = (x, y)$ in the depth buffer D, we compute depth gradient measure $G(\mathbf{q})$ that considers two types of discontinuities: object boundaries and sky regions (details are shown in Fig. 6). First, we calculate the maximum local depth gradient within a 3×3 neighborhood:

$$G_{local}(\mathbf{q}) = \max_{(\delta x, \delta y) \in \mathcal{N}_{3\times3}} |D(\mathbf{q}) - D(\mathbf{q} + (\delta x, \delta y))| \tag{8}$$

where $\mathcal{N}_{3\times3} = \{(\delta x, \delta y) || \delta x|, |\delta y| \le 1, (\delta x, \delta y) \ne (0, 0)\}$ represents the 3×3 neighborhood. Second, we detect sky regions by computing the ratio of invalid depth pixels in a larger neighborhood $\mathcal{N}_{k\times k}$ with kernel size k (here we set $k = 10$):

$$R_{sky}(\mathbf{q}) = \frac{|\{(\delta x, \delta y) \in \mathcal{N}_{10\times10} | D(\mathbf{q} + (\delta x, \delta y)) = \infty\}|}{|\mathcal{N}_{10\times10}|} \tag{9}$$

The final gradient measure combines both components:

$$G(\mathbf{q}) = \max\left(G_{local}(\mathbf{q}), \alpha \cdot \mathbb{I}(R_{sky}(\mathbf{q}) > \tau_r)\right) \tag{10}$$

where $\mathbb{I}(\cdot)$ is the indicator function, τ_r is the invalid ratio threshold, and α is a weighting factor. Pixels with gradient values exceeding the depth discontinuity threshold τ_d are marked as unreliable edge regions and excluded from colorization (edge detection results are shown in Fig. 7). This strategy effectively identifies problematic regions including object boundaries and sky edges, ensuring that only reliable point-pixel correspondences are used for final colorization.

V. EXPERIMENTAL RESULTS

To comprehensively validate EasyColor's performance, we conducted extensive quantitative and qualitative analyses on public datasets (KITTI-360 [22], FAST-LIVO2 [1] and

TABLE I

TRANSLATION ERROR (%) AND ROTATION ERROR (DEGREE/M) OF DIFFERENT METHODS ON THE KITTI-360 DATASET [22].

Method	*0000*[1]	*0002*	*0003*	*0004*	*0005*	*0006*	*0007*	*0008*
Colmap-PCD	–[2]	–	0.99 / 0.0107	–	2.29 / 0.0097	2.61 / 0.0140	2.08 / 0.0116	–
EasyColor (w/o enhance[3])	0.88 / 0.0027	0.28 / 0.0015	0.22 / 0.0012	0.32 / 0.0017	0.25 / 0.0014	0.29 / 0.0016	0.23 / 0.0013	0.27 / 0.0015
EasyColor (w/o epipolar[4])	**0.11** / 0.0016	0.21 / 0.0081	0.17 / 0.0058	0.24 / 0.0073	0.19 / 0.0092	0.34 / 0.0049	0.18 / 0.0009	0.20 / 0.0011
EasyColor (w/o keyframe[5])	1.91 / 0.0279	1.09 / 0.0744	1.11 / 0.0672	0.71 / 0.0099	2.72 / 0.0803	1.53 / 0.0209	1.20 / 0.0592	2.37 / 0.0971
EasyColor (Full)	0.16 / **0.0008**	**0.19 / 0.0010**	**0.15 / 0.0007**	**0.22 / 0.0012**	**0.11 / 0.0047**	**0.20 / 0.0011**	**0.16 / 0.0008**	**0.18 / 0.0010**

[1] "0000" means "2013_05_28_drive_0000" sequence of KITTI-360 dataset. [2] Colorizing system failed to run on this sequence.
[3] Reflectivity image enhancement (histogram and gamma). [4] Visual epipolar constraint. [5] Keyframe selection mechanism.

TABLE II

QUANTITATIVE COMPARISON (PSNR / SSIM).

Dataset	Sequence	Colmap-PCD		Baseline[1]		Ours (full)	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
FastLIVO	hku1	15.44	0.65	15.86	0.66	**20.54**	**0.84**
	hku2	15.00	0.70	14.25	0.58	**19.12**	**0.79**
FastLIVO2	CBD_build	16.27	0.72	15.87	0.69	**18.64**	**0.78**
	Main_build	13.21	0.50	10.46	0.26	**20.17**	**0.72**
	Cul_center	–[2]	–	15.10	0.75	**17.95**	**0.80**
MaRSLVIG	HKisland	14.64	0.71	11.64	0.50	**17.11**	**0.75**
	AMvalley	**18.20**	**0.88**	17.50	0.85	18.05	0.67
	AMtown	–	–	15.90	0.77	**17.80**	**0.83**
Ours	Scene A	–	–	11.80	0.42	**19.00**	**0.70**
	Scene B	13.00	**0.70**	12.40	0.46	**19.50**	0.66
	Scene C	–	–	13.70	0.50	**20.30**	**0.84**

[1] Baseline result is obtained by coloring point cloud using FastLIO2 [6] and official extrinsic file. A manual 0.1 s time offset was added to ensure a fair comparison.
[2] Colorizing system failed to run on this sequence.

Fig. 9. Our mobile mapping system. It consists of a Velodyne VLP-16 LiDAR mounted on motor platform and an omnidirectional camera.

MaRSLVIG [23]) and our self-recorded datasets (Fig. 9 shows our sensor suite, where a rotating LiDAR helps us efficiently obtain dense point cloud. We undistort the omnidirectional images and treat them as pinhole images.). In the public datasets both camera and LiDAR are sampled at 10 Hz, we add a 0.1 s time offset in all experiments to verify that EasyColor does not require strict time synchronization. Our self-recorded system captures images at 2 Hz and LiDAR at 10 Hz. Rough extrinsics are taken from CAD drawings when available, otherwise we use the official extrinsic file. Note that LVBA [24] and OmniColor [11] are not publicly released, so they are excluded from direct comparisons. Runtime analyses were performed and all baselines were tuned. All experiments ran on a PC with an Intel Core i9-12900KF (3.2 GHz) and 32 GB RAM.

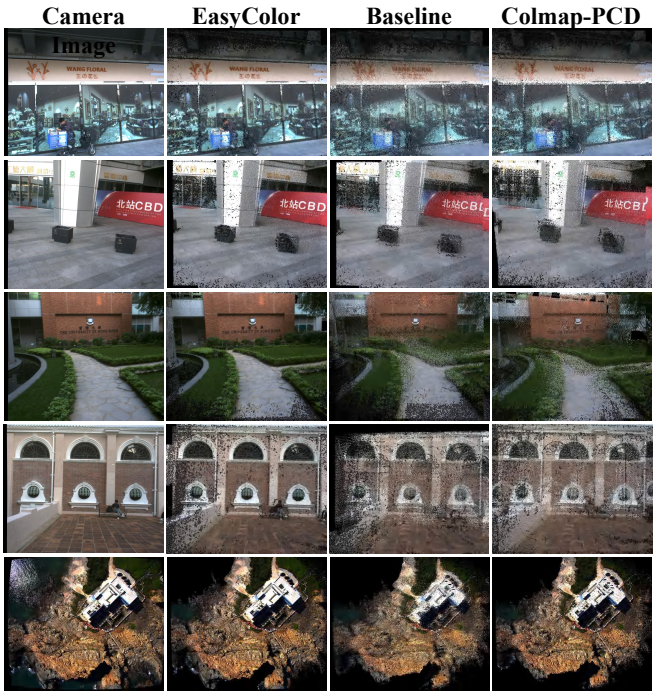

Fig. 10. Results of projecting colored point cloud generated by different methods on various public datasets onto image plane. Baseline result is obtained by coloring the point cloud using FastLIO2 [6] and official extrinsic file. A manual 0.1 s time offset was added to ensure a fair comparison.

A. Quantitative Analysis

In this section, we compare the optimized camera poses with the ground-truth camera poses from the KITTI-360 dataset to indirectly evaluate the accuracy of point cloud colorization by assessing camera pose precision. We also compute PSNR and SSIM between reprojected images and the corresponding camera image for a more direct quantitative and visual assessment.

Pose accuracy: As shown in Table I, EasyColor achieves consistently lower translation and rotation errors than Colmap-PCD across almost all sequences, reducing translation from above 2% to within 0.1–0.3% and rotation from 10^{-2} to around 10^{-3} degree/m across all sequences. The ablation results confirm the necessity of both modules: removing reflectivity enhancement (including histogram

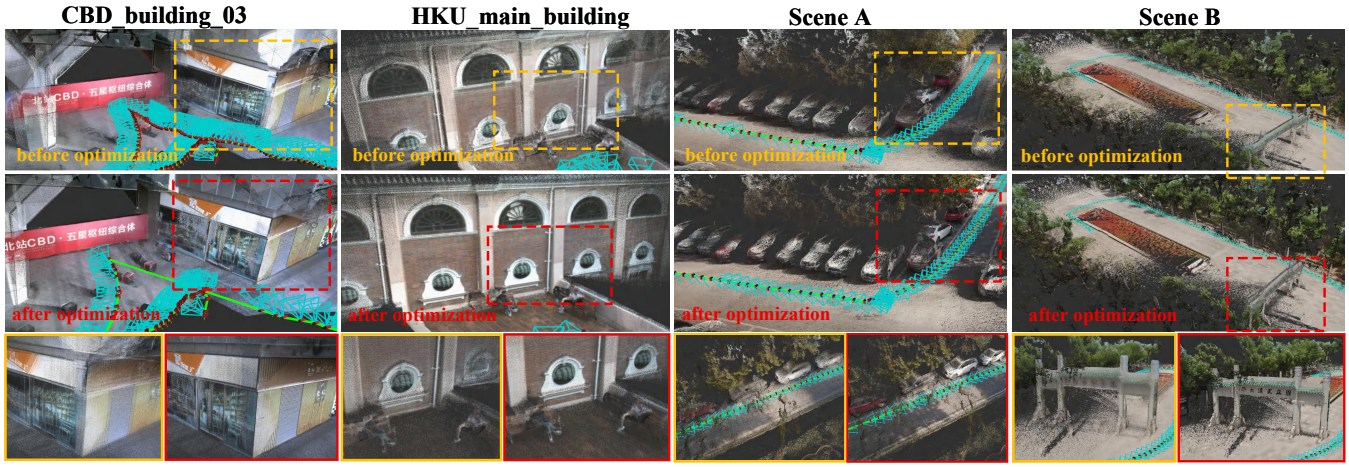

Fig. 11. Qualitative comparison of point cloud colorization results on FastLIVO2 dataset (CBD and main building sequence) and our self-recorded dataset (Scene A and Scene B). The reflectivity point cloud map for FastLIVO2 dataset is built using FastLIO2 [6], while the map for our dataset is generated with FastLIO2 [6] using our sensor suite (as shown in Fig. 9). Yellow boxes highlight unoptimized details. Red boxes highlight optimized details.

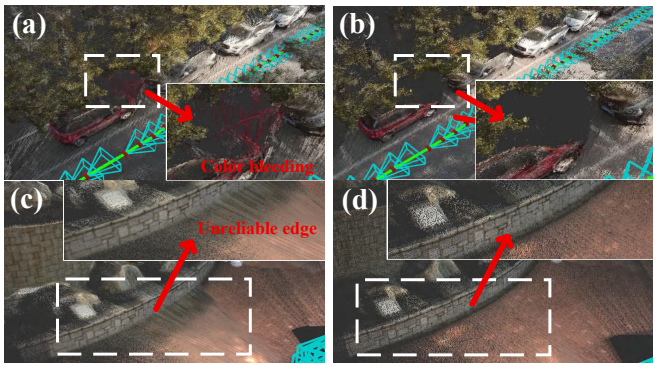

Fig. 12. Colorization comparison. (a) and (c) show colorizing result without bleeding detection and edge removal. (b) and (d) show our proposed colorizing result. White boxes highlight details (zoom in for details).

equalization and gamma correction) leads to 30–50% higher errors, while discarding epipolar constraints causes up to an order-of-magnitude degradation in rotational accuracy. The keyframe selection mechanism improves the pose accuracy by an order of magnitude. By integrating both, the full version consistently maintains the lowest errors, demonstrating that the synergy of appearance cues and geometric constraints is key to precise pose estimation and colorization.

Projection accuracy: We evaluate projection accuracy by reprojecting the colorized point clouds onto the camera image plane and comparing them with the original camera images using PSNR and SSIM. Fig. 10 presents example projection results, while Table II reports the measured PSNR/SSIM values. The results show that EasyColor achieves higher PSNR and/or SSIM on most sequences compared to the baseline and Colmap-PCD, indicating more faithful color recovery.

B. Qualitative Analysis

We conducted qualitative evaluation on the FastLIVO2 dataset and our self-recorded dataset. Beyond that, we also performed qualitative analysis of point cloud colorizing.

FastLIVO2 dataset: FastLIVO2 [1] dataset provides well-calibrated and time-synchronized point cloud and image. To demonstrate that EasyColor can work without strict time synchronization and extrinsic calibration, we did not use timestamp and extrinsic file. Instead, we first constructed a reflectivity point cloud map, which was then used together with the image sequence as input. Fig. 11 demonstrates that EasyColor significantly improves the quality of point cloud colorization compared to the unoptimized baseline, where the baseline result is obtained by coloring the point cloud using the poses provided by FastLIO2 [6] and official extrinsic file.

Self-recorded dataset: EasyColor supports most LiDAR-camera sensor suites. In theory, as long as a dense point cloud map with reflectivity intensity information and the corresponding camera image sequence are available, EasyColor can be used for point cloud colorization. Fig. 11 shows colorizing results on our self-recorded dataset (collected by sensor suite in Fig. 9). Unoptimized baseline directly uses the poses provided by SLAM (here we use FastLIO2) and well-calibrated extrinsic values (here we use Koide's work [10]) to colorize the point cloud. As you can see in Fig. 11, compared to the unoptimized baseline, EasyColor significantly improves the quality of point cloud colorization.

Point cloud colorizing: In Fig. 12, we specifically analyze the performance of point cloud colorization. It can be clearly seen that our designed color bleeding detection and unreliable edge region removal strategies effectively eliminate the effects of color bleeding and unreliable edge regions, resulting in a more consistent colored point cloud map.

C. Runtime Analysis

We conducted a detailed runtime analysis of each module of EasyColor, as summarized in Table III. Since related methods like Colmap-PCD employ global optimization strategies and cannot be directly compared with EasyColor, we only analyze EasyColor's runtime here. The total processing time for each cycle, which includes colorizing the point cloud map using two consecutive RGB images, is approximately 1.98 seconds for the FastLIVO2 dataset and 2.47 seconds for self-recorded dataset. The most time-consuming stage is the cross-domain image matching, which takes about 1.18

TABLE III

RUNTIME ANALYSIS (SECONDS PER PROCESSING CYCLE*).

Processing Stage	FastLIVO2	Self-recorded
Reflectivity Image Generation	0.36	0.52
Image Enhancement	0.01 × 2**	0.08 × 2
Projection & Indexing	0.17 × 2	0.18 × 2
Cross-Domain Matching	1.18	1.29
Joint Optimization	0.21	0.41
Point Cloud Colorization	0.23	0.25
Depth Buffer Processing	0.15	0.16
Edge Detection	0.08	0.09
Total	**1.98**	**2.47**

* Each processing cycle of EasyColor refers to colorizing the point cloud map with two consecutive RGB images.
** Reflectivity image generation needs to be executed twice.

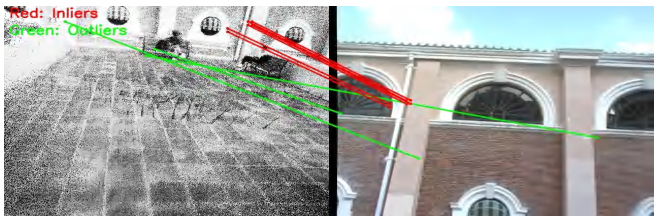

Fig. 13. A potential failure case of EasyColor.(red: inliers, green: outliers)

seconds and 1.29 seconds respectively. This is largely related to the performance of the GPU on the computing device (here, we use NVIDIA RTX 4060 8 GB). Except for the matching stage, the processing time of all other modules is within 0.5 seconds, demonstrating the potential of EasyColor to achieve real-time online colorization in the future.

D. Potential Failure Case

As shown in Fig. 13, when there is rapid inter-frame motion, very low-precision rough extrinsic estimates, or very large time offsets (larger than 5 s), the co-visible region between reflectivity image and camera image shrinks, causing cross-domain matching to produce incorrect or uneven matches and thus degrading the final point cloud colorization. Repeated textures and other factors that hurt image matching can also introduce potential errors in EasyColor's results.

VI. CONCLUSIONS

In this paper, we proposed EasyColor, a precise and robust point cloud rization method that jointly utilizes cross-domain reflectivity constraint and intra-frame visual epipolar constraint, effectively eliminating need for strict time synchronization and extrinsic calibration between camera and LiDAR sensors, enhancing the robustness and accuracy of point cloud colorization. In future work, we will further explore the integration of EasyColor with real-time mobile mapping system to achieve accurate online colored mapping.

REFERENCES

[1] C. Zheng *et al.*, "FAST-LIVO2: Fast, Direct LiDAR–Inertial–Visual Odometry," *IEEE Transactions on Robotics*, vol. 41, pp. 326–346, 2025, conference Name: IEEE Transactions on Robotics.

[2] K. Ebadi *et al.*, "Present and Future of SLAM in Extreme Environments: The DARPA SubT Challenge," *IEEE Transactions on Robotics*, vol. 40, pp. 936–959, 2024.

[3] X. Zhao *et al.*, "How Challenging is a Challenge? CEMS: a Challenge Evaluation Module for SLAM Visual Perception," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 1, p. 42, Mar. 2024.

[4] C. Campos *et al.*, "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[5] C. Kerl, J. Sturm, and D. Cremers, "Dense Visual SLAM for RGB-D Cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2100–2106.

[6] W. Xu *et al.*, "FAST-LIO2: Fast Direct LiDAR-Inertial Odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[7] T. Shan *et al.*, "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5135–5142.

[8] ——, "LVI-SAM: Tightly-coupled Lidar-Visual-Inertial Odometry via Smoothing and Mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5692–5698.

[9] C. Bai, R. Fu, and X. Gao, "Colmap-PCD: An Open-source Tool for Fine Image-to-point Cloud Registration," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 1723–1729.

[10] K. Koide *et al.*, "General, Single-shot, Target-less, and Automatic LiDAR-Camera Extrinsic Calibration Toolbox," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 11 301–11 307.

[11] B. Liu *et al.*, "OmniColor: A Global Camera Pose Optimization Approach of LiDAR-360Camera Fusion for Colorizing Point Clouds," pp. 6396–6402, 2024.

[12] C. Yuan *et al.*, "Pixel-Level Extrinsic Self Calibration of High Resolution LiDAR and Camera in Targetless Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7517–7524, 2021.

[13] X. Liu, C. Yuan, and F. Zhang, "Targetless Extrinsic Calibration of Multiple Small FoV LiDARs and Cameras Using Adaptive Voxelization," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022.

[14] G. Pandey *et al.*, "Automatic Targetless Extrinsic Calibration of a 3D Lidar and Camera by Maximizing Mutual Information," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, pp. 2053–2059, Sep. 2021.

[15] ——, "Automatic Extrinsic Calibration of Vision and Lidar by Maximizing Mutual Information," *Journal of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015.

[16] P. Vechersky *et al.*, "Colourising Point Clouds Using Independent Cameras," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3575–3582, 2018.

[17] J. Lin and F. Zhang, "R3LIVE: A Robust, Real-time, RGB-colored, LiDAR-Inertial-Visual Tightly-coupled State Estimation and Mapping Package," in *International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 672–10 678.

[18] P.-E. Sarlin *et al.*, "SuperGlue: Learning Feature Matching with Graph Neural Networks," in *CVPR*, 2020.

[19] J. Sun *et al.*, "LoFTR: Detector-Free Local Feature Matching with Transformers," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[20] G. Potje *et al.*, "XFeat: Accelerated Features for Lightweight Image Matching," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 2682–2691.

[21] J. Ren *et al.*, "MINIMA: Modality Invariant Image Matching," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 23 059–23 068.

[22] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3292–3310, 2023.

[23] H. Li *et al.*, "Mars-lvig dataset: A multi-sensor aerial robots slam dataset for lidar-visual-inertial-gnss fusion," *Int. J. Rob. Res.*, vol. 43, no. 8, p. 1114–1127, Jul. 2024.

[24] R. Li *et al.*, "LVBA: LiDAR-Visual Bundle Adjustment for RGB Point Cloud Mapping," 2024. [Online]. Available: https://arxiv.org/abs/2409.10868