

Article

LiDAR as a Geometric Prior: Enhancing Camera Pose Tracking Through High-Fidelity View Synthesis

Rafael Muñoz-Salinas ^{1,*} , Jianheng Liu ² , Francisco J. Romero-Ramirez ³ , Manuel J. Marín-Jiménez ¹ 
and Fu Zhang ² 

¹ Departamento de Informática e Inteligencia Artificial, Edificio Albert Einstein, Campus de Rabanales, Universidad de Córdoba, 14071 Córdoba, Spain

² Mechatronics and Robotic Systems (MaRS) Laboratory, Department of Mechanical Engineering, University of Hong Kong (HKU), Hong Kong, China; jianheng@connect.hku.hk (J.L.); fuzhang@hku.hk (F.Z.)

³ Departamento de Teoría de la Señal y Comunicaciones y Sistemas Telemáticos y Computación, Campus de Fuenlabrada, Universidad Rey Juan Carlos, 28942 Fuenlabrada, Spain; francisco.romero@urjc.es

* Correspondence: in1musar@uco.es; Tel.: +34-957212289

Abstract

This paper presents a robust framework for monocular camera pose estimation by leveraging high-fidelity, pre-built 3D LiDAR maps. The core of our approach is a render-and-match pipeline that synthesizes photorealistic views from a dense LiDAR point cloud. By detecting and matching keypoints between these synthetic images and the live camera feed, we establish reliable 3D–2D correspondences for accurate pose estimation. We evaluate two distinct strategies: an Online Rendering and Tracking method that renders views on the fly, and an Offline Keypoint-Map Tracking method that precomputes a keypoint map for known trajectories, optimizing for computational efficiency. Comprehensive experiments demonstrate that our framework significantly outperforms several state-of-the-art visual SLAM systems in both accuracy and tracking consistency. By anchoring localization to the stable geometric information from the LiDAR map, our method overcomes the reliance on photometric consistency that often causes failures in purely image-based systems, proving particularly effective in challenging real-world environments.

Keywords: camera pose estimation; LiDAR mapping; 3D reconstruction; visual SLAM



Academic Editor: Mauro Lo Brutto

Received: 15 July 2025

Revised: 26 July 2025

Accepted: 1 August 2025

Published: 7 August 2025

Citation: Muñoz-Salinas, R.; Liu, J.; Romero-Ramirez, F.J.; Marín-Jiménez, M.J.; Zhang, F. LiDAR as a Geometric Prior: Enhancing Camera Pose Tracking Through High-Fidelity View Synthesis. *Appl. Sci.* **2025**, *15*, 8743. <https://doi.org/10.3390/app15158743>

Copyright: © 2025 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) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Accurate camera pose estimation is a cornerstone of many computer vision and robotics applications, including autonomous navigation, augmented reality, and 3D scene reconstruction. The ability to precisely localize a camera within a known environment enables systems to interact meaningfully with the world, plan trajectories, overlay virtual content, or maintain spatial awareness. While conventional approaches often rely on sparse feature matching or visual–inertial fusion, recent advances in 3D sensing and mapping technologies offer new opportunities to enhance pose estimation using rich geometric priors.

In this paper, we investigate the potential of leveraging high-fidelity LiDAR-based 3D reconstructions to support robust and accurate camera pose tracking. Specifically, we employ FastLIVO2 [1], a state-of-the-art LiDAR (Light Detection and Ranging) odometry algorithm, to generate dense point clouds augmented with color information at each point. This detailed representation not only captures precise geometric structure but also encodes photometric cues, enabling us to synthesize realistic views of the scene

from arbitrary viewpoints. Figure 1 shows an example of a scene reconstructed using FastLivo2. The dataset was recorded using both an AVIA Livox LiDAR and an RGB camera. The top image, Figure 1a, shows the three-dimensional point cloud reconstruction. Then, Figure 1b,d show images captured with the RGB camera, and Figure 1c,e show rendered images from the exact camera location. As can be seen, the synthetic images are very similar to the real ones.



Figure 1. Retail Street sequence. (a) Three-dimensional reconstruction of the Retail Street sequence obtained with FastLivo2. (b,c) Real image and its corresponding image rendered from the point cloud. (d,e) Real image and its corresponding image rendered from the point cloud.

Our method benefits from the realism of the rendered views, which helps bridge the domain gap between synthetic and real images, thereby improving keypoint matching accuracy. Our primary approach uses this reconstruction to render a synthetic image from a known camera pose. Keypoints are then detected in both the rendered and real camera images, and correspondences between them are used to estimate the current pose. This process is iteratively applied to achieve continuous frame-to-frame tracking. To further improve robustness and provide a strong initial estimate for each frame, we integrate inertial data from the IMU (Inertial Measurement Unit) sensor to predict the camera's motion, using this prediction as a prior for our vision-based pose estimation.

However, the need to render photorealistic views at every frame introduces significant computational demands, which can limit deployment in resource-constrained platforms, such as embedded systems or low-power mobile robots. To mitigate this challenge, we propose a complementary Offline Keypoint-Map Tracking strategy designed for scenarios in which the camera trajectory is known a priori. Instead of performing rendering at runtime, we precompute a 3D keypoint map tailored to the expected viewpoints along the trajectory. This offline preprocessing shifts the computational load away from runtime, allowing for lightweight pose tracking during deployment by matching incoming frames to the precomputed keypoints.

To further evaluate this strategy, we explore two variants of the offline pipeline. The first uses the same point-based rendering technique employed in the real-time approach. The second adopts M2Mapping [2], an advanced view synthesis method that generates high-quality novel views by learning richer image-based representations. While this method produces highly realistic images, its significant computational requirements, including a lengthy training phase, make it unsuitable for real-time applications, restricting its use to our offline strategy. M2Mapping enhances the visual fidelity of the rendered images, potentially improving matching robustness in texture-sparse or geometrically complex scenes.

Through comprehensive experiments across diverse environments, we demonstrate that both the real-time and offline approaches offer viable solutions under different op-

erational constraints. Moreover, we demonstrate that incorporating learned synthesis techniques, such as M2Mapping, can significantly enhance pose estimation accuracy without compromising efficiency, thereby highlighting the promise of combining dense LiDAR reconstructions with advanced view synthesis for high-performance camera localization.

The remainder of this paper is structured as follows. Section 2 outlines the main works related to ours. Then, Section 3 explains our Online Rendering and Tracking, while Section 4 explains our Offline Keypoint-Map Tracking approach. Finally, Section 5 describes the experiments conducted to validate our proposal, and Section 6 draws some conclusions.

2. Related Works

Camera pose estimation has been a longstanding problem in robotics and computer vision, typically addressed through visual or visual–inertial odometry pipelines. These approaches are broadly categorized into indirect (feature-based) and direct (photometric) methods. Indirect methods, which our work belongs to, build a sparse map of landmarks and track the camera by matching distinctive keypoints. Direct methods, conversely, operate directly on pixel intensities, minimizing a photometric error over often dense or semi-dense regions of the image, which avoids the feature extraction step but can be more sensitive to illumination changes.

A dominant and highly successful approach within indirect methods is visual SLAM (Simultaneous Localization and Mapping). These systems build a map of 3D landmarks and track the camera’s pose by matching keypoints detected in incoming images to these landmarks. ORB-SLAM2 [3] is a landmark system in this domain, providing a complete solution for monocular, stereo, and RGB-D cameras using its fast and robust ORB (Oriented FAST and Rotated BRIEF) features. Building upon this success, ORB-SLAM3 [4] introduced a multi-map system with tighter visual–inertial integration, significantly improving accuracy and robustness. Similarly, UcoSLAM [5] offers a versatile solution that integrates natural feature points with artificial square markers, enhancing performance in texture-less environments. These highly optimized systems serve as a crucial benchmark for evaluating any new pose estimation pipeline.

The task of localizing within a pre-existing map, central to our second proposed method, is known as visual place recognition and is a critical component of large-scale localization. Modern systems often employ a hierarchical approach: first, an image retrieval stage identifies candidate locations in the map (coarse localization), followed by a fine-grained pose estimation step utilizing 3D–2D correspondences. The incorporation of learned features has significantly advanced this paradigm. For instance, SuperPoint [6] introduced a self-supervised CNN (Convolutional Neural Network) to jointly detect keypoints and compute descriptors, while SuperGlue [7] proposed a graph neural network with attention to perform highly accurate feature matching. These components are often combined in hierarchical localization pipelines [8], demonstrating the power of learned features for robust matching and localization, which informs the design of our own map-based tracking.

The fusion of LiDAR with visual and inertial modalities has also produced powerful odometry frameworks [9]. FAST-LIVO [10] and its successor FAST-LIVO2 [1] are notable examples of tightly coupled systems that fuse LiDAR, image, and inertial data. These systems enhance pose accuracy via plane priors and real-time dense mapping, which facilitates downstream tasks such as the rendering process used in our work. FusionGS-SLAM [11] extends this direction by incorporating global structural constraints into the optimization pipeline, improving global consistency and supporting photorealistic dense mapping at scale. Complementing these approaches, PRFusion [12] focuses on place

recognition, combining LiDAR and image features to enable robust location matching in visually degraded environments.

Numerous works address visual localization by matching camera inputs to previously acquired LiDAR maps. Early approaches include monocular [13] and stereo-based methods [14]. Recent systems, such as LM-LVIL [15], improve upon this by directly projecting LiDAR map points into the image frame and minimizing photometric errors. Learning-based approaches, such as CMRNet [16], also tackle camera-LiDAR registration using convolutional neural networks.

A fundamental challenge when using rendered data for training or tracking is the sim-to-real domain gap: keypoint detectors trained on real-world images often perform poorly on synthetic renderings due to differences in texture, lighting, and sensor noise. This motivates the generation of highly photorealistic views. Our approach of using high-fidelity LiDAR reconstructions is one way to mitigate this gap. Another powerful set of techniques is Neural Radiance Fields (NeRFs) [17] and its successors. Inspired by this, M2Mapping [2] unifies Neural Distance Fields and NeRFs to generate realistic renderings with structural consistency. More recently, 3D Gaussian Splatting [18] has emerged as a method for achieving real-time, high-quality novel view synthesis from point clouds, representing the state-of-the-art in rendering fidelity and efficiency. These advances in photorealistic rendering are crucial for bridging the domain gap and enabling robust keypoint matching between synthetic and real views.

Recent research efforts have sought to incorporate photorealistic rendering techniques into SLAM pipelines. Systems like VI-NeRF-SLAM [19] and NICE-SLAM [20] jointly optimize pose estimation and dense reconstruction using learned radiance representations. In a related direction, vMAP [21] introduces a vectorized object-centric representation for neural field SLAM, supporting efficient mapping and instance-level segmentation.

In contrast to radiance-based methods, SLAM3R [22] proposes a feed-forward approach that directly regresses dense 3D point maps from monocular input. Gaussian-based frameworks, such as G²-Mapping [23] and MAGiC-SLAM [24], offer scalable and real-time dense reconstruction using differentiable splatting representations. These approaches reflect a shift toward unified representations and learning-enhanced pipelines that improve accuracy, scalability, and computational efficiency.

While previous methods show powerful registration capabilities, they differ from our framework in key aspects. For instance, approaches like CMRNet are direct methods, employing a convolutional neural network to register projected LiDAR data with an image without explicit keypoint extraction. These learning-based systems can be computationally intensive, often requiring GPU acceleration, and their performance can be tied to the availability of representative training data. Similarly, emerging NeRF-based localization frameworks typically build their dense scene representation from images, which can involve a lengthy, per-scene training phase. Our framework, conversely, is designed to leverage a pre-existing geometric map from LiDAR, making it a feature-based approach that is computationally efficient at runtime and does not require per-scene training.

Our method builds on this body of work by combining dense LiDAR-based rendering with image-based keypoint matching. We directly compare our approach against leading visual SLAM pipelines to demonstrate its competitive performance, particularly leveraging the geometric accuracy of LiDAR to overcome the typical failure modes of purely visual systems while addressing the sim-to-real challenge through high-quality rendering.

3. Online Rendering and Tracking

Our proposed methodology for robust camera pose tracking leverages a high-fidelity 3D reconstruction of the environment to synthesize realistic reference views. By matching

features between these synthetic views and the live camera feed, we can accurately and continuously estimate the camera's pose. The core of our approach is an iterative process that minimizes the reprojection error between matched keypoints. An overview of the algorithm is presented below.

3.1. Algorithmic Overview

Let the world frame be denoted by $\mathcal{F}_W$. We assume that a dense, high-fidelity, colored 3D point cloud of the environment, $\mathcal{P} \subset \mathbb{R}^3 \times \mathbb{R}^3$, has been pre-acquired using a state-of-the-art LiDAR odometry algorithm such as FastLIVO2. Each point $\mathbf{P}_i \in \mathcal{P}$ is represented by its 3D coordinates $\mathbf{X}_i = [X_i, Y_i, Z_i]^T$ in $\mathcal{F}_W$ and its associated color information $\mathbf{C}_i = [R_i, G_i, B_i]^T$.

The tracking process is initialized with a known camera pose. For each subsequent frame, the algorithm performs the following steps:

1. IMU-based Pose Prediction: Using the final pose from the previous frame, we integrate IMU measurements to predict the camera's pose for the current frame. This serves as a strong prior for the subsequent steps.
2. Synthetic View Rendering: Given the IMU-predicted camera pose, a synthetic RGBD image is rendered from the 3D point cloud $\mathcal{P}$.
3. Keypoint Detection and Matching: Keypoints are detected in both the rendered synthetic image and the newly captured real camera image. These keypoints are then matched to establish correspondences.
4. Pose Estimation via PnP: The 3D–2D correspondences between the keypoints in the synthetic view (and thus, the 3D points in $\mathcal{P}$) and the keypoints in the real image are used to correct the IMU-based prediction and estimate the camera's final pose. This is formulated as a Perspective-n-Point (PnP) problem, where the objective is to minimize the reprojection error.
5. Iterative Refinement: The newly estimated pose serves as the corrected pose for the current frame, and the process is repeated, beginning with the next IMU prediction.

3.2. Formal Description

3.2.1. Point-Based RGBD Synthetic Rendering

This process, which we refer to as point-based rendering, generates a synthetic image by directly projecting points from the 3D cloud. Let the camera pose at time t be represented by a rigid body transformation $\mathbf{T}_t = [\mathbf{R}_t | \mathbf{t}_t] \in SE(3)$, where $\mathbf{R}_t \in SO(3)$ is the rotation matrix and $\mathbf{t}_t \in \mathbb{R}^3$ is the translation vector that transforms points from the world frame $\mathcal{F}_W$ to the camera frame $\mathcal{F}_C$.

The projection of a 3D point $\mathbf{P}_i = (\mathbf{X}_i, \mathbf{C}_i)$ from the point cloud $\mathcal{P}$ onto the image plane is given by the pinhole camera model:

$$\mathbf{p}_i^s = \mathbf{K}[\mathbf{R}_t | \mathbf{t}_t]\mathbf{X}_i \quad (1)$$

where $\mathbf{p}_i^s = [u_i^s, v_i^s, 1]^T$ are the homogeneous coordinates of the projected point in the synthetic image I^s , and $\mathbf{K}$ is the camera intrinsic matrix:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

Here, (f_x, f_y) are the focal lengths and (c_x, c_y) are the principal point coordinates.

By projecting every point $\mathbf{P}_i \in \mathcal{P}$ onto the image plane, we generate a synthetic RGB image I^s and a corresponding depth map D^s . The color of a pixel (u, v) in I^s is determined

by the color $\mathbf{C}_i$ of the projected point $\mathbf{P}_i$, and its depth value in D^s is the z-component of the point in the camera coordinate system. A z-buffer is employed to handle occlusions.

3.2.2. Keypoint Detection and Matching

For each new real image I^r captured by the camera, we detect a set of keypoints $\mathcal{K}^r = \{\mathbf{k}_j^r\}_{j=1}^{N_r}$, where each keypoint $\mathbf{k}_j^r$ is defined by its 2D coordinates $\mathbf{u}_j^r = [u_j^r, v_j^r]^T$ and an associated descriptor $\mathbf{d}_j^r$. Similarly, a set of keypoints $\mathcal{K}^s = \{\mathbf{k}_k^s\}_{k=1}^{N_s}$ with coordinates $\mathbf{u}_k^s$ and descriptors $\mathbf{d}_k^s$ is extracted from the rendered synthetic image I^s .

Correspondences between the two sets of keypoints are established by finding pairs of descriptors that have the minimum distance under a chosen metric (e.g., Euclidean distance for SIFT (Scale-Invariant Feature Transform) or Hamming distance for ORB). This results in a set of matched pairs $\mathcal{M} = \{(\mathbf{k}_j^r, \mathbf{k}_k^s)\}$. For each matched pair, we have a 2D point $\mathbf{u}_j^r$ from the real image and a corresponding 3D point $\mathbf{X}_k$ from the world point cloud, as the synthetic keypoint $\mathbf{k}_k^s$ was rendered from $\mathbf{X}_k$.

3.2.3. Pose Estimation Using PnP

The final step is to compute the camera's current pose $\mathbf{T}_{t+1}$ by leveraging the set of N 3D-2D correspondences $\{(\mathbf{X}_i, \mathbf{u}_i^r)\}_{i=1}^N$ obtained from the matching step. The PnP algorithm is employed to find the rotation $\mathbf{R}_{t+1}$ and translation $\mathbf{t}_{t+1}$ that minimize the reprojection error.

The reprojection error is defined as the sum of the squared distances between the observed 2D keypoints in the real image and the projected 3D points from the world point cloud. The objective function to be minimized is:

$$E(\mathbf{R}_{t+1}, \mathbf{t}_{t+1}) = \sum_{i=1}^N \|\mathbf{u}_i^r - \pi(\mathbf{R}_{t+1}, \mathbf{t}_{t+1}, \mathbf{X}_i)\|^2 \quad (3)$$

where $\pi(\cdot)$ is the projection function that maps the 3D point $\mathbf{X}_i$ onto the image plane given the pose $(\mathbf{R}_{t+1}, \mathbf{t}_{t+1})$:

$$\pi(\mathbf{R}_{t+1}, \mathbf{t}_{t+1}, \mathbf{X}_i) = \begin{pmatrix} f_x \frac{X'_i}{Z'_i} + c_x \\ f_y \frac{Y'_i}{Z'_i} + c_y \end{pmatrix} \quad (4)$$

and where $[X'_i, Y'_i, Z'_i]^T = \mathbf{R}_{t+1}\mathbf{X}_i + \mathbf{t}_{t+1}$.

This non-linear least-squares problem is typically solved using iterative algorithms, such as the Levenberg–Marquardt algorithm. To ensure robustness against outlier matches, the PnP solver is often embedded within a RANSAC (Random Sample Consensus) scheme. The resulting pose $\mathbf{T}_{t+1} = [\mathbf{R}_{t+1} | \mathbf{t}_{t+1}]$ is then used as the initial guess for the subsequent frame, continuing the tracking process.

4. Offline Keypoint-Map Tracking

The online rendering requirement of the previously described method can be computationally intensive, limiting its deployment on resource-constrained platforms. To address this, we propose an alternative approach specifically designed for scenarios where the camera's trajectory is known beforehand, such as in robotic applications involving repetitive tasks.

This method fundamentally alters the computational paradigm by bifurcating the process into an intensive offline preparation phase and an efficient online tracking phase. The core idea is to shift the computational load away from runtime. During the offline phase, we pre-render all necessary views along the known trajectory and extract a sparse map of stable 3D keypoints. Consequently, the online tracking phase becomes significantly more

lightweight, as its tasks are limited to retrieving candidate points from this pre-built map and matching them against the incoming camera frames. This strategy prioritizes runtime efficiency and predictability, making it the superior choice when the path is predefined.

4.1. Offline Map Generation

Let the predefined path of the robot be a sequence of M poses in the world frame $\mathcal{F}_W$, denoted by the trajectory $\mathcal{T} = \{\mathbf{T}_1^p, \mathbf{T}_2^p, \dots, \mathbf{T}_M^p\}$, where $\mathbf{T}_k^p = [\mathbf{R}_k^p | \mathbf{t}_k^p]$.

1. **View Rendering and Keypoint Extraction:** For each pose $\mathbf{T}_k^p$ along the trajectory, we render a synthetic RGBD image pair (I_k^s, D_k^s) using the projection model defined in Equation (1). From each synthetic image I_k^s , we detect a set of keypoints $\mathcal{K}_k^s = \{(\mathbf{u}_{k,i}^s, \mathbf{d}_{k,i}^s)\}$. Critically, since each keypoint $\mathbf{u}_{k,i}^s$ originates from rendering the dense point cloud $\mathcal{P}$, we know its precise 3D coordinates $\mathbf{X}_{k,i}$ in $\mathcal{F}_W$.
2. **Temporal Keypoint Filtering:** To build a robust map, we only retain keypoints that demonstrate stability across consecutive frames. A keypoint with 3D position $\mathbf{X}$ detected in frame k is considered stable if it is also visible and detectable in frame $k+1$. We formalize this by projecting the point $\mathbf{X}$ into the subsequent view using pose $\mathbf{T}_{k+1}^p$:

$$\mathbf{p}_{k \rightarrow k+1} = \pi(\mathbf{R}_{k+1}^p, \mathbf{t}_{k+1}^p, \mathbf{X}) \quad (5)$$

A keypoint is kept if a keypoint detected in frame $k+1$ exists in the neighborhood of $\mathbf{p}_{k \rightarrow k+1}$ with a similar descriptor. This filtering process yields a consistent set of trackable 3D points.

3. **Keyframe-based Map Construction:** The final map $\mathcal{M}_{map}$ is constructed from these filtered 3D points. To enable efficient spatial lookups during online tracking, we structure the map around keyframes. A keyframe $\mathcal{F}_k$ is defined by the tuple:

$$\mathcal{F}_k = (\mathbf{T}_k^p, \mathcal{M}_k) \quad (6)$$

where $\mathbf{T}_k^p$ is the pose of the keyframe and $\mathcal{M}_k$ is the set of stable 3D keypoints $\{(\mathbf{X}_j, \mathbf{d}_j)\}$ visible from that pose. The complete map is the collection of all such keyframes, $\mathcal{M}_{map} = \{\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_M\}$.

The effectiveness of this offline tracking paradigm is critically dependent on the quality and robustness of the pre-generated keypoint map. A key factor in the map's quality is the fidelity of the synthetic views used during its offline creation, as these views must be realistic enough to produce keypoints that reliably match with real-world images. To investigate this relationship, we evaluate two different rendering approaches for the map generation phase.

The first approach is the point-based rendering technique employed in our online method, which directly projects the 3D point cloud. As a more advanced alternative, we also generate a keypoint map using views synthesized by M2Mapping [2]. M2Mapping is a state-of-the-art method designed to produce high-quality, photorealistic novel views from LiDAR scans. Unlike direct point-based projection, M2Mapping involves a two-stage process. First, it requires an offline training phase where it uses the LiDAR scans and the corresponding video sequence to train a model of the scene's geometry and appearance. Second, in a separate rendering phase, this trained model is used to synthesize novel views from any desired viewpoint. Both stages are computationally intensive and time-consuming, precluding their use in our online method; however, M2Mapping is an excellent candidate for offline map generation, where rendering quality is crucial. By leveraging its advanced synthesis capabilities, we aim to create a more robust keypoint map that can further improve tracking accuracy, especially in visually complex scenes.

4.2. Online Tracking with the Pre-Generated Map

During the online phase, the robot localizes itself utilizing IMU data and the pre-built map to achieve localization. Given a refined pose from the previous frame, the algorithm first integrates IMU measurements to generate a predicted pose $\mathbf{T}_t$ for the current frame. This prior is then refined as follows:

1. **Keyframe Retrieval:** The algorithm first identifies the most relevant keyframe(s) from the map. This is achieved by finding the keyframe $\mathcal{F}_{k^*}$ whose position is closest to the IMU-predicted position $\mathbf{t}_t$:

$$k^* = \arg \min_{k \in \{1, \dots, M\}} \|\mathbf{t}_t - \mathbf{t}_k^p\| \quad (7)$$

The set of 3D keypoints associated with this keyframe, $\mathcal{M}_{k^*} = \{(\mathbf{X}_j, \mathbf{d}_j)\}$, is retrieved as the set of candidate points for matching.

2. **Projection and Matching:** The candidate 3D points $\{\mathbf{X}_j\}$ are projected onto the current real camera image plane using the current pose estimate $\mathbf{T}_t$:

$$\mathbf{p}_j = \pi(\mathbf{R}_t, \mathbf{t}_t, \mathbf{X}_j) \quad (8)$$

Simultaneously, keypoints are detected in the real image I_t^r . A search is performed to find correspondences between the projected map points $\{\mathbf{p}_j\}$ and the detected real keypoints $\{\mathbf{u}_i^r\}$. A match is established if a real keypoint is found within a search radius of a projected point and their descriptors (the precomputed descriptor $\mathbf{d}_j$ and the new descriptor $\mathbf{d}_i^r$) are sufficiently similar.

3. **Pose Estimation:** The resulting set of N 3D-2D correspondences, $\{(\mathbf{X}_i, \mathbf{u}_i^r)\}_{i=1}^N$, is used to compute the updated camera pose $\mathbf{T}_{t+1}$. As with the first method, this is achieved by using a PnP algorithm within a RANSAC scheme to solve the reprojection error minimization problem defined in Equation (3). The refined pose $\mathbf{T}_{t+1}$ is the final output for the current frame and serves as the starting point for the next IMU integration cycle.

5. Experiments

This section outlines the experiments conducted to validate our proposal. To that end, we have employed a total of nine sequences from the FastLivo2 Dataset [1]. These sequences were recorded simultaneously using a Livox Avia LiDAR and an RGB camera. In the dataset, both the LiDAR and the camera capture data at 10 Hz, while the IMU operates at a higher frequency of 100 Hz. The datasets are recorded in a wide variety of outdoor environments covering distances of up to 300 m. The sequences were first processed using FastLivo2 to obtain the point cloud and the camera poses. These poses are considered the ground truth for our experiments. The generated point cloud had a large number of points and, in some cases, some noise. Therefore, we filtered the point cloud in two stages using the Point Cloud Library [25]. We first downsampled it using a voxel grid filter (2 mm leaf size) and then cleaned it with a statistical outlier removal filter. Figure 2 shows some of the point clouds from the dataset employed.

This section is divided into the following subsections. Section 5.1 explains the experimental setup. In Section 5.2, we show the results of our Online Rendering and Tracking method (Section 3), while Section 5.3 explains the results obtained with our Offline Keypoint-Map Tracking approach (Section 4). In this case, as already explained, we will test two different rendering methods: the point-based rendering method and M2Mapping [2]. Section 5.4 evaluates state-of-the-art visual SLAM methods on the same datasets, and

Section 5.5 performs a final comparison of our methods and the visual SLAM methods. Finally, Section 5.6 reflects on the main limitations of the proposed method.



Figure 2. Examples of LiDAR reconstructions in the dataset. (a) HITGraffitiWall dataset. (b) HKU-Landmark dataset. (c) SYSU dataset. (d) CBDBuilding dataset. (e) Garden dataset. (f) RedSculpture dataset.

5.1. Experimental Setup and Performance

All experiments were conducted on a system equipped with an Intel(R) Xeon(R) Silver 4316 CPU at 2.30 GHz, running Ubuntu 20.04.6 LTS. An NVIDIA GeForce RTX 3090 GPU with 24 GB of memory was used for the computationally intensive training and rendering tasks required by M2Mapping.

For keypoint detection, we selected the AKAZE [26] and ORB [27] feature detectors, as both are computationally efficient and widely used in real-time SLAM systems [3–5,28]. To further optimize performance, keypoint detection in both our online and offline methods is parallelized. The input camera image is divided into three horizontal regions, and a

dedicated thread is used to compute keypoints in each region simultaneously. This multi-threaded approach allows the main tracking loop to achieve approximately 30 FPS when using AKAZE features and up to 80 FPS with the faster ORB detector.

In the Online Rendering and Tracking approach, we employed a thread for image rendering and three threads for keypoint detection. For the Offline Keypoint-Map Tracking method, three threads were employed for keypoint detection. This highlights the fundamental difference in runtime load: the first method requires a dedicated, continuous computational effort for rendering, which is entirely eliminated during the tracking phase of the second method.

For the point-based rendering method, we developed an optimized algorithm that runs at 16 FPS. This optimization relies on a preprocessing step where the 3D LiDAR point cloud is organized into a grid of 10 cm voxels. At runtime, the system first performs a fast check by projecting only the centers of these voxels onto the image plane. A full projection of all points within a voxel is only performed if its center falls within the current camera's field of view. This hierarchical culling strategy significantly reduces the number of points that must be processed for each frame, making real-time rendering feasible. Nevertheless, a GPU-based rendering could significantly speed up this process even further.

The selection of the system's parameters was guided by an empirical analysis to balance performance and quality. The 2 mm distance threshold for point filtering represents a good trade-off between removing sensor noise and preserving the geometric fidelity of surfaces, ensuring the point cloud is clean for rendering. Similarly, the 10 cm voxel grid used for rendering optimization was chosen to maximize view-frustum culling efficiency without introducing the computational overhead of a finer grid or the precision loss of a coarser one. Together, these values enable the generation of high-quality synthetic views at high frame rates.

5.2. Experiments with Online Rendering and Tracking

This experiment analyzes the Online Rendering and Tracking method, as explained in Section 3. Each sequence has been processed using both the AKAZE and ORB descriptors, measuring the ATE (Absolute Trajectory Error) and the total number of tracked frames, as in previous works [5].

Table 1 presents the performance of our online tracking method using two different keypoint detectors: AKAZE and ORB. The comparison, based on the Absolute Trajectory Error (ATE) for accuracy and the percentage of tracked frames (Trck) for robustness, shows that AKAZE consistently outperforms ORB across all nine datasets.

Table 1. Tracking results of the online rendering and tracking approach, comparing AKAZE and ORB detectors. ATE values are rounded to three decimal places.

Dataset	AKAZE		ORB	
	ATE (m)	Trck (%)	ATE (m)	Trck (%)
Garden	0.031	84.1	0.045	81.3
RetailStreet	0.037	92.8	0.056	92.8
CBDBuilding	0.048	93.5	0.091	79.0
HITGraffitiWall	0.050	94.8	0.069	94.7
HKULandmark	0.055	83.6	0.083	80.2
Drive	0.070	93.2	0.132	89.2
HKUCulturalCenter	0.086	90.7	0.163	78.6
RedSculpture	0.107	90.5	0.210	72.9
SYSU	0.103	78.4	0.122	75.1

On average, AKAZE reduces the ATE by approximately 37% compared to ORB. This accuracy improvement is particularly significant in challenging sequences, with the error being reduced by nearly 49% in the *RedSculpture* dataset (0.107 m vs. 0.210 m) and by 47% in the *CBDBuilding* dataset (0.048 m vs. 0.091 m).

In terms of robustness, AKAZE also demonstrates a clear advantage, consistently tracking a higher percentage of frames. The improvement is most pronounced in the *HKUCulturalCenter* sequence, where AKAZE successfully tracks over 12% more of the trajectory than ORB. Even in the one dataset where both detectors achieve an identical tracking rate, *RetailStreet* (92.8%), AKAZE remains significantly more accurate.

These results strongly suggest that the features provided by AKAZE are more stable and discriminative for matching between real and rendered images. Consequently, AKAZE is the more suitable choice for our tracking pipeline, especially in real-world scenarios with substantial variations in viewpoint or texture.

The architecture is designed for maximum responsiveness. The rendering and tracking processes run in parallel and are decoupled. The tracking thread operates at its maximum speed, using the most recently completed synthetic image as its reference without being blocked by the rendering process. This results in a tracking loop latency determined only by the keypoint stage: approximately 33.3 ms (30 FPS) for AKAZE and 12.5 ms (80 FPS) for ORB. The rendering thread updates the reference view in the background at its own pace of 16 FPS. This design ensures the system is highly responsive and can comfortably process a 30 Hz camera feed in real-time. While this introduces a minor, acceptable delay in the *freshness* of the rendered view, the tracking itself remains fast.

The Impact of IMU Data

To assess the contribution of the Inertial Measurement Unit (IMU) to our Online Rendering and Tracking pipeline, we conducted an ablation study where the IMU-based pose prediction step was entirely removed. In this configuration, the pose estimated from the previous frame serves as the sole prior for the current frame. This experiment allows us to isolate the performance of the vision-only tracking component and quantify the benefits of incorporating inertial data. The results of this experiment are presented in Table 2.

Table 2. Tracking results of the online rendering and tracking approach, without using IMU data.

Dataset	AKAZE		ORB	
	ATE (m)	Trck (%)	ATE (m)	Trck (%)
Garden	0.031	84.1	0.055	79.5
RetailStreet	0.037	92.8	0.068	90.4
CBDBuilding	0.057	89.2	0.167	71.5
HITGraffitiWall	0.061	92.1	0.186	73.2
HKULandmark	0.071	81.1	0.167	72.9
Drive	0.077	90.9	0.199	69.7
HKUCulturalCenter	0.102	88.4	0.217	71.8
RedSculpture	0.122	88.3	0.298	64.2
SYSU	0.115	77.5	0.281	62.6

A comparison between these results and those in Table 1 reveals the influence of the IMU on tracking performance. For the AKAZE detector, removing the IMU results in a minor degradation in performance. The ATE increases slightly across most datasets, and the tracking percentage sees a slight reduction. For instance, in the *CBDBuilding* sequence, the ATE increases from 0.048 m to 0.057 m, while tracking robustness drops from 93.5% to 89.2%. In several cases, such as *Garden* and *RetailStreet*, the impact is

negligible, demonstrating the strong standalone capability of the vision-based matching with AKAZE features.

In contrast, the ORB detector shows a greater dependence on the IMU's motion prior. Without it, both accuracy and robustness decline significantly. The ATE for ORB more than doubles in some sequences, such as *HITGraffitiWall* (from 0.069 m to 0.186 m) and *RedSculpture* (from 0.210 m to 0.298 m). Similarly, tracking rates fall more sharply, with the *Drive* sequence dropping from 89.2% to 69.7%.

This analysis indicates that while the IMU provides a clear benefit to the overall system, its influence is most pronounced when using the ORB detector. As ORB is the less discriminative feature descriptor in our pipeline, it relies more heavily on the predictive power of the IMU to constrain the search space for feature matching. Ultimately, this experiment confirms that the IMU is a valuable component for enhancing robustness but is not strictly necessary to obtain good results, especially when a strong feature detector like AKAZE is employed. Although not explicitly presented here for brevity, similar findings were observed for the offline tracking methods, as they rely on the same fundamental pose estimation process.

5.3. Experiments with Offline Keypoint-Map Tracking

This section shows the results of the Offline Keypoint-Map Tracking, explained in Section 4. As already indicated, the idea consists of rendering images along the expected path that the camera will follow and obtaining the corresponding keypoint map. In this experiment, we have tested two different approaches to building the keypoint map. In the first one, we have used the point-based rendering approach employed in the Online Rendering and Tracking method. In the second approach, we have employed the novel view synthesis method M2Mapping [2], which takes as input the laser scans and obtains a three-dimensional representation of the environment, which can be used later to render views from novel camera viewpoints. As noted, this method requires a lengthy offline training phase followed by a rendering phase. Although not measured with high precision, on our hardware, the training phase took approximately two hours per sequence, and the subsequent rendering of all necessary views took about one hour. This significant computational overhead confirms its unsuitability for any real-time application but makes it ideal for precomputing high-quality keypoint maps where time is not a constraint. However, runtime performance is dictated solely by the lightweight tracking thread, achieving high frame rates of approximately 30 FPS (33.3 ms latency) with AKAZE and 80 FPS (12.5 ms latency) with ORB.

Figure 3 shows the differences between the point-based rendering approach and M2Mapping. Figure 3a shows the three-dimensional reconstruction obtained with FastLivo2, and Figure 3b is a real image from the dataset. Figure 3c shows the image rendered by the point-based projection approach, and Figure 3d the one obtained by M2Mapping. As can be seen, the last one is less noisy. It is true, though, that some fine-grained details are lost, and the image is somewhat blurry.

Below, we show the results obtained using both approaches.

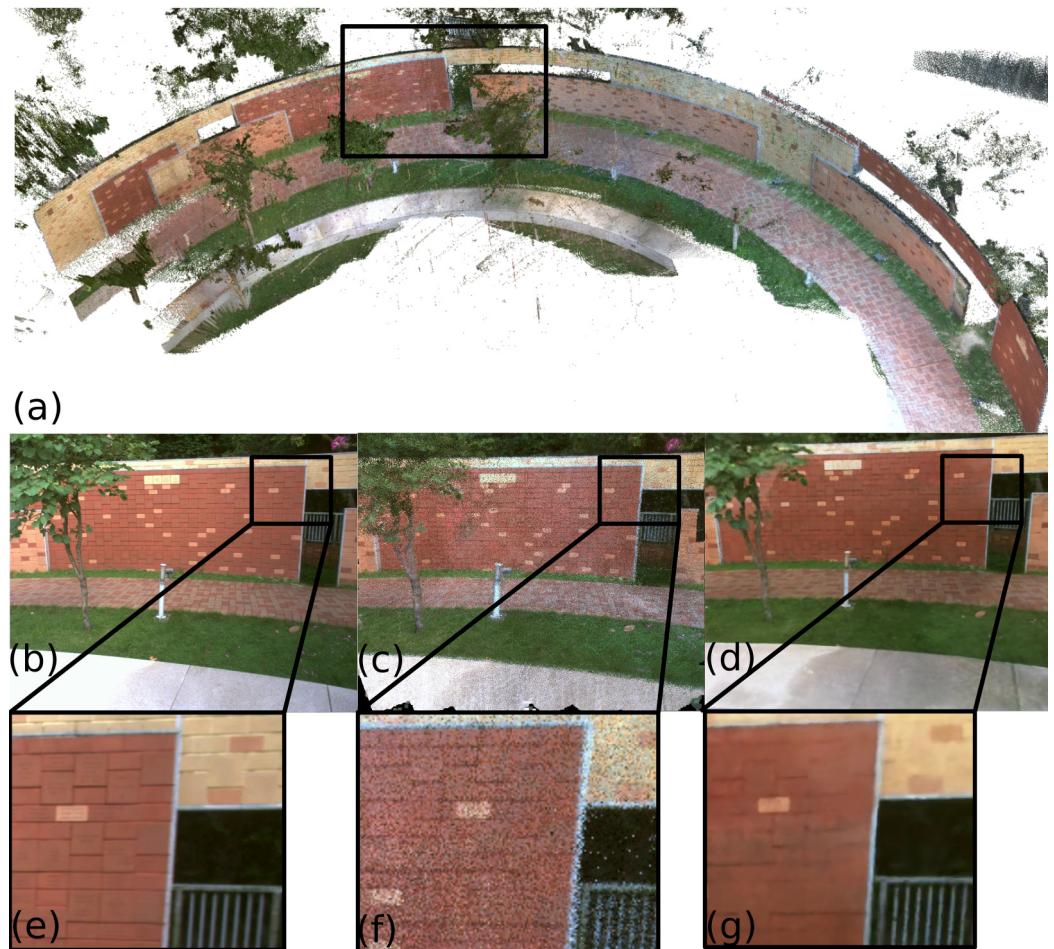


Figure 3. Rendering example. (a) Three-dimensional reconstruction of the Garden sequence. (b) Real image camera image. (c) Image rendered using the point-based rendering approach. (d) Image rendered using M2Mapping. (e–g) Zoom in on a specific area of the image. See text for more details.

5.3.1. Point-Based Rendering

The results for the offline tracking method using point-based rendering, shown in Table 3, present the performance of the AKAZE and ORB detectors in this configuration.

AKAZE is shown to be effective, delivering consistent performance with high tracking rates and low ATE values across the tested datasets. In contrast, ORB exhibits considerable instability with this method. In several sequences, the tracking rate is very low, falling below 2% in four cases, indicating a general failure to maintain tracking. A particular case is the *Drive* dataset, where ORB reports a low ATE. This result, however, corresponds to a tracking duration of only 0.3% of the sequence, making the ATE value unrepresentative of the overall performance.

When comparing these findings to the online method's results in Table 1, a notable difference in the behavior of ORB can be observed. While ORB was a usable high-speed alternative in the online method, it does not provide the same level of reliability with this offline point-based rendering approach. This suggests that for the offline strategy presented here, AKAZE is the more dependable choice for achieving consistent results.

Table 3. Tracking results of the Offline Keypoint-Map Tracking method using point-based rendering. ATE values are rounded to three decimal places.

Dataset	AKAZE		ORB	
	ATE (m)	Trck (%)	ATE (m)	Trck (%)
Garden	0.040	84.1	0.310	38.3
RetailStreet	0.037	92.8	0.082	89.3
CBDBuilding	0.056	95.5	0.305	0.6
HITGraffitiWall	0.035	94.8	0.377	2.0
HKULandmark	0.056	83.8	0.222	58.9
Drive	0.088	93.1	0.001	0.3
HKUCulturalCenter	0.076	92.2	0.000	0.0
RedSculpture	0.108	85.0	0.218	0.6
SYSU	0.094	81.6	0.217	11.2

5.3.2. M2Mapping Rendering

Table 4 presents the results of the enhanced version of the Offline method, which incorporates a refined rendering approach M2Mapping [2]. This modification aims to improve keypoint projection and stability. As with the previous versions, we evaluate performance using AKAZE and ORB across nine datasets, reporting both the Absolute Trajectory Error (ATE) and the tracking percentage (Trck).

Table 4. Results of the Offline Keypoint-Map Tracking method using M2Mapping rendering. ATE values are rounded to three decimal places.

Dataset	AKAZE		ORB	
	ATE (m)	Trck (%)	ATE (m)	Trck (%)
Garden	0.039	84.1	0.011	1.1
RetailStreet	0.037	92.8	0.061	90.3
CBDBuilding	0.054	96.8	0.166	2.2
HITGraffitiWall	0.025	94.8	0.165	4.9
HKULandmark	0.035	86.2	0.074	1.9
Drive	0.056	93.2	0.073	91.1
HKUCulturalCenter	0.063	93.8	0.357	1.2
RedSculpture	0.103	90.4	0.003	0.3
SYSU	0.086	75.3	0.110	26.9

AKAZE maintains strong performance with M2Mapping, achieving low ATEs (ranging from 0.025 to 0.103) and high tracking percentages in most cases, notably surpassing 90% in five datasets. The improvements in stability and accuracy are particularly evident in sequences like *CBDBuilding* (ATE: 0.054 m, Trck: 96.8%) and *RetailStreet* (ATE: 0.037 m, Trck: 92.8%), demonstrating a slight improvement over the previous projection method.

In contrast, ORB continues to struggle with robustness. While it achieves low ATEs in isolated cases (e.g., *Garden*, ATE: 0.011 m; *RedSculpture*, ATE: 0.003 m), these results are misleading due to extremely low tracking percentages (1.1% and 0.3%, respectively). In general, ORB's tracking drops dramatically across all datasets, with values below 5% in six out of nine sequences. The only exceptions where ORB approaches usable performance are *RetailStreet* (90.3%) and *Drive* (91.1%), but even in those, ATE remains higher than AKAZE.

Compared to point-based rendering, the M2Mapping enhancement improves AKAZE's tracking robustness slightly (e.g., +1.3% in *CBDBuilding*, +0.6% in *Drive*) while maintaining a low error. However, it does not resolve the weaknesses of ORB, whose performance remains unstable and highly scene-dependent.

These results further support the conclusion that descriptor choice is more critical than rendering refinement. While M2Mapping enhances projection accuracy, reliable tracking is only achievable when paired with robust and discriminative descriptors such as AKAZE.

5.4. Comparison with Visual SLAM Approaches

In this section, we compare the results obtained by our method with those of state-of-the-art visual SLAM approaches, namely, ORB-SLAM2 [3], ORB-SLAM3 [4], OpenVSLAM [28], and UcoSLAM [5]. To validate each method, a map of the environment was built for each video sequence in the dataset. These pre-built maps were then used in a second run in localization mode, and the performance was evaluated in terms of the Absolute Trajectory Error (ATE) and the percentage of correctly tracked frames (Trck). The results are presented in Table 5.

Table 5. Quantitative evaluation of ORB-SLAM3, OpenVSLAM, UcoSLAM, and ORB-SLAM2 results on the datasets. The table reports the Absolute Trajectory Error (ATE) and tracking percentage (Trck).

Dataset	ORB-SLAM3		OpenVSLAM		UcoSLAM		ORB-SLAM2	
	ATE (m)	Trck (%)	ATE (m)	Trck (%)	ATE (m)	Trck (%)	ATE (m)	Trck (%)
Garden	22.875	100.0	0.209	83.8	0.017	86.9	0.073	39.7
RetailStreet	-	-	0.036	99.1	14.397	46.2	0.032	100.0
CBDBuilding	31.527	100.0	0.315	100.0	0.100	100.0	0.065	48.9
HITGraffitiWall	-	-	0.075	98.0	0.089	96.6	3.517	26.7
HKULandmark	36.124	100.0	0.030	93.7	0.039	95.3	0.022	97.8
Drive	39.113	100.0	1.297	40.7	0.485	92.1	0.033	15.8
HKUCulturalCenter	-	-	-	-	-	-	0.207	22.7
RedSculpture	17.705	100.0	0.931	41.7	0.007	2.8	1.578	10.0
SYSU	6.797	100.0	0.062	100.0	0.102	100.0	0.067	99.3

The results in the comparison table present a clear picture of the trade-offs inherent in purely visual SLAM systems.

- ORB-SLAM3 suffers from very high trajectory errors in the sequences where it can be initialized, despite maintaining a high tracking rate (100%). This behavior is likely exacerbated by the low camera frame rate of 10 Hz, as feature-based SLAM systems often rely on higher frame rates for more reliable frame-to-frame motion estimation. It also fails to produce results for some datasets.
- ORB-SLAM2, its predecessor, shows a different performance profile. It achieves the best accuracy among all tested SLAM systems on certain datasets, such as HKULandmark (ATE: 0.022 m) and RetailStreet (ATE: 0.032 m). However, it proves to be highly unreliable, with tracking failures on the majority of the other sequences, where its tracking rate often drops below 50%.
- OpenVSLAM performs significantly better than ORB-SLAM3 in terms of accuracy but exhibits variable tracking robustness. It notably fails to track well in more dynamic or large-scale scenes, such as Drive (Trck: 40.7%) and RedSculpture (Trck: 41.7%).
- UcoSLAM exhibits a more balanced performance in some cases but can be unpredictable, failing completely in others (e.g., Trck: 2.8% on RedSculpture).

These comparisons highlight a key weakness in purely visual SLAM systems: a lack of consistent reliability. Each system excels in certain conditions but is fragile in others. In contrast, our proposed framework, by anchoring its localization to a geometrically stable LiDAR map, achieves comparable or superior performance across a broader range of challenging datasets. This demonstrates that our approach is a valid and highly competitive

alternative to traditional visual SLAM, particularly in situations where robustness and consistency are crucial.

5.5. Overall Performance Comparison

To provide a conclusive performance benchmark, this section compares our proposed framework against state-of-the-art visual SLAM systems. The evaluation focuses on robust tracking, where a system must not only be accurate but also operate consistently. Therefore, for this head-to-head comparison, we only consider results from systems that successfully track more than 75% of the frames in a given sequence. A “-” in the table indicates that the tracking rate for that system fell below this reliability threshold. For our methods, all results are reported using the superior AKAZE detector. The comprehensive results, summarized in Table 6, reveal the strengths and weaknesses of each approach.

Table 6. Overall ATE (m) performance comparison focusing on robust trackers (Track > 75%). All results from our methods use the AKAZE detector. A ‘-’ indicates the tracking rate was below 75%. The best result for each dataset is shown in bold.

Dataset	Our Methods (AKAZE)			Visual SLAM Methods			
	Online	Offline (Point-Based)	Offline (M2Mapping)	ORB-SLAM3	OpenVSLAM	UcoSLAM	ORB-SLAM2
Garden	0.031	0.040	0.039	22.875	0.209	0.017	-
RetailStreet	0.037	0.037	0.037	-	0.036	-	0.032
CBDBuilding	0.048	0.056	0.054	31.527	0.315	0.100	-
HITGraffitiWall	0.050	0.035	0.025	-	0.075	0.089	-
HKULandmark	0.055	0.056	0.035	36.124	0.030	0.039	0.022
Drive	0.070	0.088	0.056	39.113	-	0.485	-
HKUCulturalCenter	0.086	0.076	0.063	-	-	-	-
RedSculpture	0.107	0.108	0.103	17.705	-	-	-
SYSU	0.103	0.094	0.086	6.797	0.062	0.102	0.067

Our proposed framework demonstrates an advantage in the most challenging scenarios. Across the nine datasets, our methods achieve the lowest ATE in five cases when compared to reliable visual SLAM systems.

The robust comparison is particularly insightful in challenging sequences, such as *Drive* and *RedSculpture*. On these datasets, most visual SLAM systems fail to maintain tracking and are thus excluded from this analysis. The few that remain operational, such as UcoSLAM on *Drive* or ORB-SLAM3 on *RedSculpture*, produce substantial errors. In stark contrast, our Offline (M2Mapping) method provides a significant improvement in accuracy while maintaining high tracking rates, achieving an ATE of just 0.056 m on *Drive* and 0.103 m on *RedSculpture*.

Furthermore, on the *HKUCulturalCenter* sequence, no competing visual SLAM system could meet the 75% tracking threshold, leaving our framework as the only effective and reliable solution presented. Our Offline (M2Mapping) approach successfully tracked this sequence with an ATE of 0.063m. In other datasets, such as *CBDBuilding* and *HITGraffitiWall*, our methods also deliver top performance.

While highly optimized visual SLAM systems like UcoSLAM, ORB-SLAM2, and OpenVSLAM achieve the best results in some of the more conventional sequences (*Garden*, *RetailStreet*, *HKULandmark*, and *SYSU*), they lack the consistent reliability of our approach. This analysis confirms the central thesis of our work: by leveraging the geometric stability of a pre-built LiDAR map, our framework provides a more robust and consistently accurate localization solution than state-of-the-art visual-only methods.

5.6. Limitations of the Proposed Framework

While the proposed framework demonstrates strong performance across various benchmarks, it is important to acknowledge its limitations.

The Offline Keypoint-Map Tracking method is explicitly designed for scenarios where the camera's trajectory is known beforehand. Its primary weakness is its high sensitivity to deviations from the predefined path. The online tracking phase relies on retrieving keyframes from the map based on the camera's predicted proximity to the precomputed path. If the camera deviates significantly from the expected trajectory, the retrieved 3D points will not align well with the live view, resulting in matching failures and a loss of tracking. This makes the offline approach unsuitable for exploratory navigation or situations where the path cannot be guaranteed.

A core characteristic of the framework is its reliance on a static, pre-built LiDAR map. This presents a limitation in dynamic real-world environments. In its current form, the system does not explicitly model or account for dynamic objects. Consequently, significant changes in the scene can introduce discrepancies between the static rendered view and the live camera feed. This may affect the reliability of keypoint matching, especially if the dynamic elements occlude a large portion of the static background. While the RANSAC-based pose estimation and the IMU data provide inherent robustness to a certain number of outlier matches, widespread or drastic changes to the environment could degrade tracking accuracy or lead to temporary failures. Addressing this would require online map updating capabilities, as noted in our future work. Similarly, the system is susceptible to unexpected occlusions not present in the original map, such as a person walking directly in front of the camera. Such events can temporarily reduce the number of visible, valid keypoints, and if the occlusion is substantial, it could compromise the tracker's stability until a clear view is re-established.

Both of our methods use IMU data to predict the camera's pose, which serves as a strong prior for the vision-based matching step. While this improves efficiency and robustness in most cases, the system can fail if there is abrupt, unexpected motion that is not well predicted by the IMU. A poor initial pose guess can prevent the system from finding enough correct 2D–3D matches, leading to tracking failure.

6. Conclusions

In this paper, we presented a robust and accurate framework for monocular camera pose estimation by leveraging high-fidelity, pre-built LiDAR maps. Our core contribution is a render-and-match pipeline that bridges the domain gap between a sparse 3D point cloud and a 2D camera image. By generating synthetic views from the dense LiDAR reconstruction, we establish reliable keypoint correspondences for precise pose estimation. The integration of IMU data as a motion prior provides a strong initial guess for each frame, significantly enhancing the robustness and efficiency of the tracking process.

We demonstrated the effectiveness of this framework through two distinct strategies: an Online Rendering and Tracking method that renders synthetic views in real-time, and an Offline Keypoint-Map Tracking method that precomputes a 3D keypoint map from the known trajectory to enable lightweight localization. Both approaches leverage the pre-existing, geometrically accurate LiDAR map to anchor the localization process, overcoming the reliance on photometric consistency that is a primary failure point for purely image-based methods.

Our experimental validation showed marked improvements in both accuracy and robustness compared to several state-of-the-art visual SLAM systems. The proposed methods maintained consistent, low-error tracking in challenging real-world conditions where visual-only systems often failed. The effectiveness of both the Online Rendering and Track-

ing and the Offline Keypoint-Map Tracking strategies confirms that the rich geometric detail from LiDAR is a powerful and consistent source of information for camera localization.

Future work could extend this framework in several promising directions. One key area is exploring methods to update the LiDAR map online to account for dynamic objects or long-term environmental changes. Additionally, investigating more advanced rendering techniques could further improve the realism of the synthetic views and, consequently, the matching performance. Finally, adapting the system for use with other sensor modalities, such as thermal or event cameras, could open up new applications for robust localization in even more extreme environments.

Author Contributions: Conceptualization, R.M.-S. and M.J.M.-J.; methodology, R.M.-S. and F.J.R.-R.; software, R.M.-S. and J.L.; validation, R.M.-S., J.L. and F.J.R.-R.; formal analysis, R.M.-S. and M.J.M.-J.; investigation, R.M.-S.; resources, R.M.-S., J.L. and F.J.R.-R.; data curation, R.M.-S., J.L. and F.J.R.-R.; writing—original draft preparation, R.M.-S.; writing—review and editing, R.M.-S., M.J.M.-J., J.L., F.Z. and F.J.R.-R.; visualization, R.M.-S.; supervision, R.M.-S.; project administration, R.M.-S., M.J.M.-J. and F.Z.; funding acquisition, R.M.-S., M.J.M.-J. and F.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded under the projects PID2023-147296NB-I00 of the Ministry of Science, Innovation and Universities of Spain and the Hong Kong Research Grants Council (UGC) General Research Fund (GRF), project number 17205924.

Data Availability Statement: Publicly archived datasets can be found at <https://www.uco.es/investiga/grupos/ava/> (accessed on 31 July 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zheng, C.; Xu, W.; Zou, Z.; Hua, T.; Yuan, C.; He, D.; Zhou, B.; Liu, Z.; Lin, J.; Zhu, F.; et al. FAST-LIVO2: Fast, Direct LiDAR-Inertial-Visual Odometry. *IEEE Trans. Robot.* **2025**, *41*, 326–346. [CrossRef]
2. Liu, J.; Zheng, C.; Wan, Y.; Wang, B.; Cai, Y.; Zhang, F. Neural Surface Reconstruction and Rendering for LiDAR-Visual Systems. *arXiv* **2024**, arXiv:2108.10470.
3. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Trans. Robot.* **2017**, *33*, 1255–1262. [CrossRef]
4. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM. *IEEE Trans. Robot.* **2021**, *37*, 1874–1890. [CrossRef]
5. Muñoz-Salinas, R.; Medina-Carnicer, R. UcoSLAM: Simultaneous Localization and Mapping by Fusion of KeyPoints and Squared Planar Markers. *Pattern Recognit.* **2020**, *101*, 107193. [CrossRef]
6. DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperPoint: Self-Supervised Interest Point Detection and Description. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, USA, 18–22 June 2018; pp. 337–33712. [CrossRef]
7. Sarlin, P.E.; DeTone, D.; Malisiewicz, T.; Rabinovich, A. SuperGlue: Learning Feature Matching With Graph Neural Networks. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 4937–4946. [CrossRef]
8. Sarlin, P.E.; Cadena, C.; Siegwart, R.; Dymczyk, M. From Coarse to Fine: Robust Hierarchical Localization at Large Scale. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 12708–12717. [CrossRef]
9. Fan, Z.; Zhang, L.; Wang, X.; Shen, Y.; Deng, F. LiDAR, IMU, and camera fusion for simultaneous localization and mapping: A systematic review. *Artif. Intell. Rev.* **2025**, *58*, 174. [CrossRef]
10. Zheng, C.; Zhu, Q.; Xu, W.; Liu, X.; Guo, Q.; Zhang, F. FAST-LIVO: Fast and Tightly-coupled Sparse-Direct LiDAR-Inertial-Visual Odometry. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 4003–4009. [CrossRef]
11. Phan, T.D.; Kim, G.W. FusionGS-SLAM: Multiple Sensors Fusion for Localization and Real-Time Photorealistic Mapping. *IEEE Robot. Autom. Lett.* **2025**, *10*, 8690–8697. [CrossRef]

12. Wang, S.; Kang, Q.; She, R.; Zhao, K.; Song, Y.; Tay, W.P. PRFusion: Toward Effective and Robust Multi-Modal Place Recognition With Image and Point Cloud Fusion. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 20523–20534. [\[CrossRef\]](#)
13. Caselitz, T.; Steder, B.; Ruhnke, M.; Burgard, W. Monocular camera localization in 3D LiDAR maps. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016; pp. 1926–1931. [\[CrossRef\]](#)
14. Kim, Y.; Jeong, J.; Kim, A. Stereo Camera Localization in 3D LiDAR Maps. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–9. [\[CrossRef\]](#)
15. Leng, J.; Sun, C.; Wang, B.; Lan, Y.; Huang, Z.; Zhou, Q.; Liu, J.; Li, J. Cross-Modal LiDAR-Visual-Inertial Localization in Prebuilt LiDAR Point Cloud Map Through Direct Projection. *IEEE Sens. J.* **2024**, *24*, 33022–33035. [\[CrossRef\]](#)
16. Cattaneo, D.; Vaghi, M.; Ballardini, A.L.; Fontana, S.; Sorrenti, D.G.; Burgard, W. CMRNet: Camera to LiDAR-Map Registration. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 1283–1289. [\[CrossRef\]](#)
17. Mildenhall, B.; Srinivasan, P.P.; Tancik, M.; Barron, J.T.; Ramamoorthi, R.; Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* **2021**, *65*, 99–106. [\[CrossRef\]](#)
18. Kerbl, B.; Kopanas, G.; Leimkuehler, T.; Drettakis, G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* **2023**, *42*, 1–25. [\[CrossRef\]](#)
19. Liao, D.; Ai, W. VI-NeRF-SLAM: A real-time visual-inertial SLAM with NeRF mapping. *J. Real-Time Image Process.* **2024**, *21*, 30. [\[CrossRef\]](#)
20. Zhu, Z.; Peng, S.; Larsson, V.; Xu, W.; Bao, H.; Cui, Z.; Oswald, M.R.; Pollefeys, M. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 12786–12796.
21. Kong, X.; Liu, S.; Taher, M.; Davison, A.J. VMAP: Vectorised Object Mapping for Neural Field SLAM. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 952–961.
22. Liu, Y.; Dong, S.; Wang, S.; Yin, Y.; Yang, Y.; Fan, Q.; Chen, B. Slam3r: Real-time dense scene reconstruction from monocular rgb videos. In Proceedings of the Computer Vision and Pattern Recognition Conference, Nashville, TN, USA, 11–15 June 2025; pp. 16651–16662.
23. Chen, L.; Hu, B.; Wang, J.; Bu, S.; Wang, G.; Han, P.; Chen, J. G²-Mapping: General Gaussian Mapping for Monocular, RGB-D, and LiDAR-Inertial-Visual Systems. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 12347–12357. [\[CrossRef\]](#)
24. Yugay, V.; Gevers, T.; Oswald, M.R. Magic-slam: Multi-agent gaussian globally consistent slam. In Proceedings of the Computer Vision and Pattern Recognition Conference, Nashville, TN, USA, 11–15 June 2025; pp. 6741–6750.
25. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Nashville, TN, USA, 11–15 June 2011; pp. 1–4. [\[CrossRef\]](#)
26. Alcantarilla, P.; Nuevo, J.; Bartoli, A. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. In Proceedings of the British Machine Vision Conference, Bristol, UK, 9–13 September 2013; BMVA Press: Aberdeen, UK, 2013.
27. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 2564–2571. [\[CrossRef\]](#)
28. Sumikura, S.; Shibuya, M.; Sakurada, K. OpenVSLAM: A Versatile Visual SLAM Framework. In Proceedings of the MM '19: 27th ACM International Conference on Multimedia, Nice, France, 21–25 October 2019; pp. 2292–2295. [\[CrossRef\]](#)

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.