TerraSLAM: Towards GPS-Denied Localization

Jingao Xu, Mihir Bala, Thomas Eiszler, Xiangliang Chen, Qifei Dong, Aditya Chanana, Padmanabhan Pillai, Mahadev Satyanarayanan
Carnegie Mellon University
{jingaox,mbala,teiszler,xianglic,qifeid,achanana,pspillai}@andrew.cmu.edu
satya@cs.cmu.edu

Abstract

A long-standing concern with GPS-based location sensing is its vulnerability to satellite signal loss. This may arise from adversarial attacks or natural causes such as urban canyons. Today, there are no real alternatives to GPS for providing absolute global coordinates. We address this concern by introducing TerraSLAM, a new global positioning system that uses a 3D GIS model to bridge relative and absolute coordinate systems, thereby enhancing visual SLAM to function as a global positioning solution. TerraSLAM offers localization accuracy and efficiency comparable to GPS-RTK even in GPS-denied settings. Extensive evaluation on drone localization scenarios shows that TerraSLAM achieves an average global positioning accuracy of 0.21m and a 99th percentile within 0.67m, outperforming advanced GPS solutions by over 70% (0.72m) and 80% (3.62m). Additionally, when integrated with ORB-SLAM3, the localization latency per frame is 16.7ms, achieving a 60% reduction compared to the baseline of 41.3ms. Code is available at https://github.com/cmusatyalab/TerraSLAM.

CCS Concepts

• **Computer systems organization** → **Embedded and cyber-physical systems**; **Real-time systems**.

Keywords

Mobile Computing; Edge Computing; Drone Localization; SLAM

ACM Reference Format:
Jingao Xu, Mihir Bala, Thomas Eiszler, Xiangliang Chen, Qifei Dong, Aditya Chanana, Padmanabhan Pillai, Mahadev Satyanarayanan. 2025. TerraSLAM: Towards GPS-Denied Localization. In *The 23rd Annual International Conference on Mobile Systems, Applications and Services (MobiSys '25), June 23–27, 2025, Anaheim, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3711875.3729144

1 Introduction

The Global Positioning System (GPS[1]) and related Global Navigation Satellite Systems (GNSS[1]), like BeiDou[2] and Galileo[3], provide essential global and absolute geospatial coordinates—longitude, latitude, and altitude (Fig.1(a))—that are foundational for modern industries such as aviation, maritime, and transportation[1]. Today, numerous autonomous devices like drones, robots, and vehicles need GPS for tasks like surveying[4], disaster response[5], and autonomous driving[6], all of which require precise and reliable global geospatial coordinates for navigation.

Table 1: Overall System Performance Comparison

Solution	Accuracy (m)		Latency (ms)		Global Coordinates	
	Mean	P99	Mean	P99	Normal	GPS-Denied
GPS	3.77	10.61	<10	<10	✔	✗
DGPS	0.72	3.62	10-15	10-15	✔	✗
RF[14, 15]	0.5-1	2-3	10-20	20-30	✗	✗
Radar[16, 17]	0.1-0.5	1-2	20-30	30-40	✗	✗
SLAM[18–20]	0.1-0.5	1-2	30-50	60-80	✗	✗
TerraSLAM	**0.21**	**0.67**	**16.7**	**31.9**	✔	✔

Ground truth from GPS-RTK with post-processing[21].

However, as demonstrated in Table 1, GPS frequently encounters localization errors over 10 meters in environments like urban canyons or dense foliage, failing to meet the precision demands of autonomous technologies[7–9]. While differential GPS (DGPS) and real-time kinematics (RTK) enhance accuracy, they rely on densely pre-deployed ground stations, increasing costs and limiting universality[10, 11]. In scenarios like military zones where satellite services are typically attacked and denied, devices cannot receive GPS signals, rendering all GPS-related technologies ineffective. Generally, GPS unreliability and denial are escalating issues, impacting a wide range of areas and technologies[1, 12, 13].

As alternative localization solutions for mobile devices, a broad spectrum of systems based on RF signals[14, 15, 22–24], radar[16, 17, 25, 26], and vision[8, 27–29] have recently emerged. Among them, visual Simultaneous Localization and Mapping (SLAM), noted for its precision and minimal sensor dependency (i.e., only a monocular camera), has become pivotal in drone flight control and robotic operations[30–32]. As shown in Fig.1(b), SLAM processes continuous image sequences to determine the camera's pose and generate an environmental model (i.e., a set of 3D map points)[18–20].

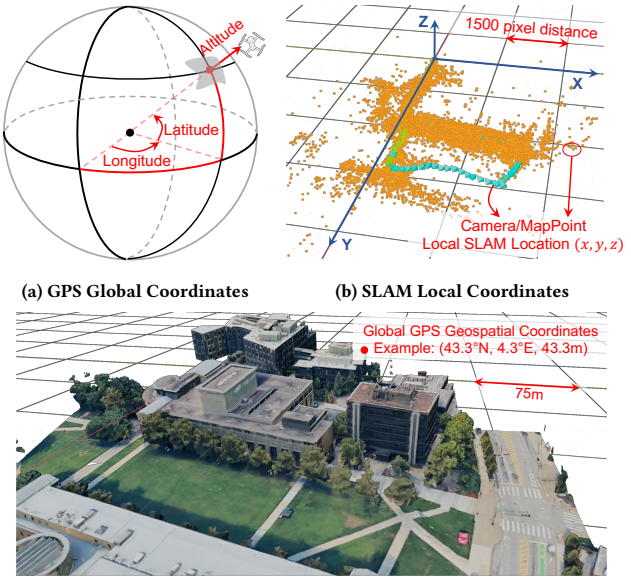
Unfortunately, these technologies merely provide coordinates within user-specified, relative, and *local* coordinate systems, rather than geospatial, absolute, and *global* ones like GPS. For instance, SLAM yields relative locations based on the camera's initial frame[30]. To pursue global coordinates, they inevitably depend on GPS; hence still face GPS shortcomings. We therefore ask:
"As computer vision techniques—particularly visual SLAM—rapidly advance, how can we transform SLAM from local positioning into a global localization method comparable to GPS?"

In this work, we answer this question and demonstrate that SLAM's advanced capabilities can enable mobile devices to locate in geospatial space. Recent advancements in 3D geographic information system (GIS) present the key opportunity. 3D GIS services are extensively leveraged in urban planning, industrial inspection,

[1]For simplicity, we use GPS to represent GNSS solutions in this work.

MobiSys '25, June 23–27, 2025, Anaheim, CA, USA

ACM ISBN 979-8-4007-1453-5/25/06
https://doi.org/10.1145/3711875.3729144



(a) GPS Global Coordinates (b) SLAM Local Coordinates

(c) 3D GIS Model

Figure 1: TerraSLAM is the first work exploiting (c) to bridge the gap between (a) and (b).

and agriculture to visualize geospatial data with 3D models, enhancing user interaction and improving property management efficiency[33, 34]. Prominent platforms like Google Earth[35] and ArcGIS[36] showcase the wide coverage of GIS services.

Our key insight is illustrated in Fig.1. As shown, most points in 3D GIS models possess global geographic coordinates [34]. Additionally, these models can align with point clouds generated by SLAM to calculate a coordinate transformation matrix. Such a dual linkage provides the connection between local SLAM and global GPS coordinates (detailed in §4.1). However, translating the insight into a practical system and elevating visual SLAM to a general-purpose localization service on par with GPS face two challenges.

• **Extreme heterogeneity between SLAM and GIS models.** Environmental point clouds generated by SLAM and those from 3D GIS models exhibit significant differences in density, scale, and orientation. Additionally, those pre-established GIS models are frequently outdated across various regions. Creating a transformation matrix between such heterogeneous point clouds is generally regarded as an NP-hard problem[9, 37] and traditional Iterative Closest Point (ICP) based solutions [38] fail to address these discrepancies.

• **Considerable discrepancies in SLAM and GPS localization latency.** As a visual-based technique, SLAM achieves high-precision localization via computationally intensive methods for local and global location optimization[31, 32]. On mobile devices, this often exceeds 40ms per frame, paling in comparison to GPS chips which can operate at over 60Hz (16.7ms). Such inefficiency also hinders SLAM from replacing GPS, in services requiring frequent localization, like drone flight control[39, 40].

We tackle the above challenges and propose **TerraSLAM**, the first system to elevate visual **SLAM**, originally a local positioning technology, to deliver global geospatial (**Terra-**) coordinates by fully leveraging the GIS model as a bridge. TerraSLAM can deliver global coordinates with accuracy and efficiency comparable to GPS-RTK, especially in GPS-denied environments. TerraSLAM contains two key plug-in modules, as described below.

Firstly, we introduce a *Semantic-based World Alignment* framework to align 3D environmental point clouds from SLAM with those from the GIS model with semantic information assistance. On this basis, we calculate a transformation matrix that converts localization results from local to global coordinates (§4); Secondly, we propose a *Hierarchical SLAM Acceleration* strategy, which encompasses a suite of methods designed to speed up SLAM, while maintaining high accuracy (§5).

We have applied TerraSLAM to drone localization scenarios, evaluating its capability to obtain geospatial coordinates in various environments. Table 1 presents the overall performance comparison of TerraSLAM relative to current practices. The experiments involved over three hours of drone flight data, capturing more than 360,000 video frames. Evaluation results show that TerraSLAM achieves an average localization accuracy of 0.21m with a 99% tail accuracy of 0.67m, improving accuracy by 71% and 81% over mainstream DGPS, respectively. Additionally, TerraSLAM reduces the average frame processing delay to 16.7ms, a 60% decrease compared to the original ORB-SLAM3.

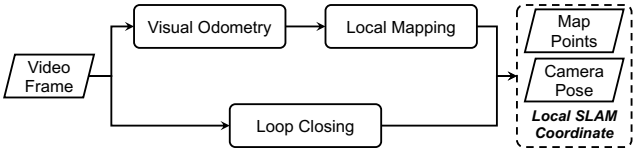
In summary, this paper makes the following contributions.
(i) We propose TerraSLAM, the first system to elevate SLAM outputs from constrained, human-defined local coordinates to universal, global coordinates even in GPS-weak or denied environments.
(ii) We show that GIS models can bridge SLAM and GPS coordinates. We further propose two plug-in modules for a practical system that makes the accuracy and efficiency of TerraSLAM comparable to GPS-RTK.
(iii) We deploy TerraSLAM in drone localization scenarios, and present extensive evaluation results to demonstrate its superior performance.

2 Background and Related Work

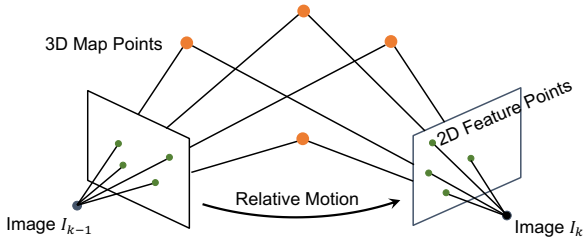
2.1 Localization Services

GPS and related GNSS localization services are vital for navigation and mapping, using trilateration from three or more satellites to determine positions. However, inaccuracies often arise from atmospheric delays affecting satellite signals[1]. Differential GPS (DGPS) reduces errors by referencing ground-based or cellular stations [10]. GPS Real-Time Kinematic (RTK) achieves centimeter-level precision via carrier phase enhancement and, when raw data is further post-processed, can reach sub-centimeter accuracy[11]. Recent enhancements use filters to optimize DGPS exploiting historical trajectories[41]. However, all these methods rely heavily on GPS signals and falter in satellite denial scenarios[1, 12, 13].

As sensors proliferate on mobile devices, diverse localization technologies employing inertial measurement units (IMU)[42–44], radio frequency (RF) signals[14, 15, 23, 24, 45, 46], visual images[8, 27–29, 47], millimeter-wave radar[16, 25, 26], and LiDAR[48–51] have developed, achieving accuracy down to sub-meter and centimeter levels. RF-Diffusion[52] establishes a time-frequency diffusion theory, and proposed the first generative diffusion model for RF sensing and localization with millimeter-level accuracy. However, these methods provide locations only within local rather



(a) Typical Visual SLAM Architecture



(b) Visual Odometry

Figure 2: Visual SLAM and Visual Odometry

than GPS global coordinate systems[30]. Recent works leverage OpenStreetMap[53] and Google Street View[54] images for global positioning, but achieve only approximate region-level locations with errors over $5m$[55, 56]. In summary, there are no real alternatives to GPS for providing accurate global coordinates.

2.2 Visual SLAM

Visual SLAM, a cornerstone in robotics, has been intensively studied for decades[30]. It involves simultaneously constructing a model of the environment and estimating the robot's state within that space. These systems commonly utilize monocular[57, 58], stereo[18], or RGB-D cameras[59]. Top-ranked implementations include LSD-SLAM[57], VINS[60], and ORB-SLAM[61]. We use monocular ORB-SLAM3[19] to illustrate how SLAM operates. As shown in Fig.2(a), three interconnected modules calculate the camera's pose and create an environmental 3D model from input images.

• **Visual Odometry** (VO) extracts 2D ORB feature points from each video frame and estimates camera poses (i.e., location and orientation) based on spatial geometry from feature matching between frames, as illustrated in Fig.2(b). However, due to cumulative errors in 2D feature extraction and matching, visual odometry offers only a rough estimate of the camera's pose.

• **Local Mapping** (LM) utilizes nonlinear optimization algorithms, like Bundle Adjustment (BA[62]), to refine camera poses and generate new 3D map points by optimizing feature points within a defined local time window.

• **Loop Closing** (LC) identifies and further corrects cumulative errors by recognizing previously visited locations (e.g., repeated paths, similar scenes). It applies global BA to refine the entire map, ensuring consistent and accurate mapping over long durations.

Although SLAM technology has significantly advanced, two major issues hinder its application as a general-purpose localization service: (i) SLAM outputs are confined to a local coordinate system, using the camera's initial position as the origin and pixel distances for scale, both vastly different from GPS's global coordinates; (ii) The visual feature extraction and matching in VO, and the optimizations in LM and LC, are computationally intensive for mobile devices[31, 63]. Recent studies explore edge computing to boost

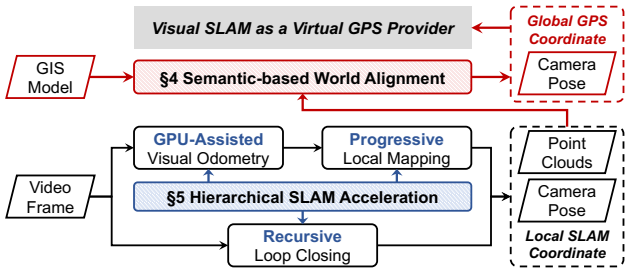


Figure 3: Overview of TerraSLAM

mobile-side running efficiency[31, 32, 64]. edgeSLAM2[65] reshapes the edge-assisted SLAM paradigm by unleashing the power of on-chip intelligence, outperforming existing methods on resource-constrained devices. TerraSLAM directly optimizes these modules, which is orthogonal and complementary to existing works.

2.3 3D GIS Services and Models

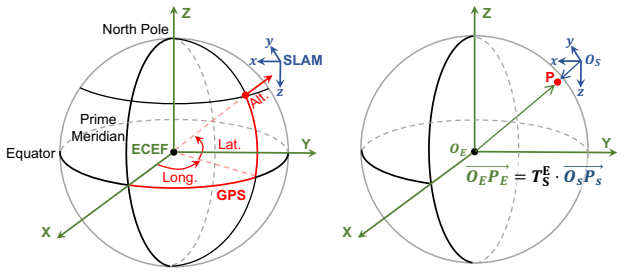
"*A geographic information system (GIS) is a conduit that connects data to geographic maps, transforming originally static database entries–such as industrial production, agricultural, and urban mobility information–into observable, analyzable, and interactive elements through GIS models*"[33]. Today, 3D GIS models are the cornerstone of GIS, offering comprehensive three-dimensional representations of physical spaces in world coordinate systems, serving as containers for data visualization and analysis. Additionally, as industries increasingly digitize, 3D GIS models themselves have gradually become valuable digital assets for businesses[34].

The popularity of 3D GIS services has also surged due to technological advances. Improvements in GPS-RTK and computer vision have yielded effective algorithms for creating high-precision 3D models in world coordinate systems using cameras[66] and LiDAR[17]. Furthermore, the evolution of 3D GIS platforms has significantly advanced GIS technology. Systems like ArcGIS[36] provide tools for easily creating customized 3D GIS models and services. Moreover, well-known platforms such as Google Earth[35] and Apple Map[67] offer 3D GIS models for many global locations, expanding the reach of 3D GIS models and services. Amid this trend, we leverage the detailed geographical and geometric information behind 3D GIS models to bridge SLAM and GPS coordinate systems.

3 System Overview

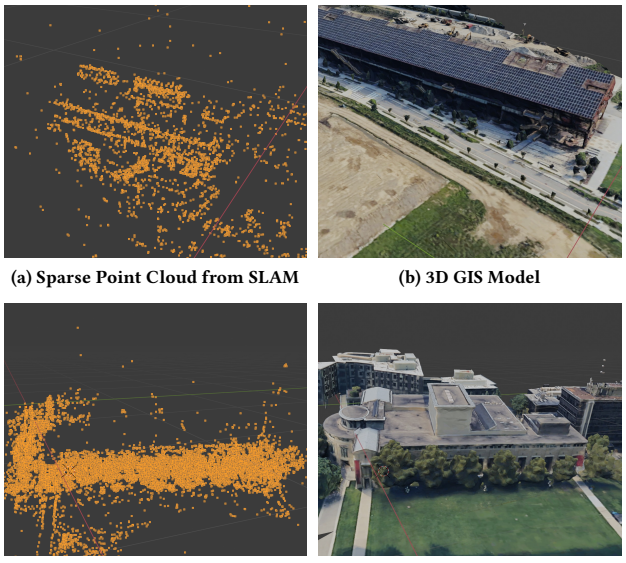
The system architecture of TerraSLAM is depicted in Fig.3. Notably, TerraSLAM does not introduce a new SLAM algorithm; rather, it includes two plug-in modules, *Semantic-based World Alignment* (SWA) and *Hierarchical SLAM Acceleration*. They are compatible with most existing SLAM algorithms, enabling them to provide global geospatial coordinates for mobile devices in real-time.

As for the specific data flow, when images are input, visual SLAM continuously determines the camera's location and generates environmental point clouds in SLAM's local coordinate system. During this process, the *Hierarchical SLAM Acceleration* strategy, with its three plug-in sub-modules, accelerates the Visual Odometry, Local Mapping, and Loop Closing stages, significantly reducing localization latency to levels comparable with current GPS solutions (e.g., DGPS). Subsequently, the *Semantic-based World Alignment* module aligns the generated SLAM point clouds with pre-acquired 3D GIS



(a) (b)

Figure 4: Coordinate Systems in TerraSLAM. (a) Illustration of GPS, SLAM, and ECEF coordinate systems. (b) Coordinates transformation from SLAM to ECEF.



(a) Sparse Point Cloud from SLAM (b) 3D GIS Model

(c) Sparse Point Cloud from SLAM (d) 3D GIS Model

Figure 5: Comparison of SLAM and 3D GIS models. (a) and (b) show a factory setting, while (c) and (d) depict a university.

models (from platforms like Google Earth or ArcGIS, or through manual surveying). It calculates a coordinate transformation matrix, which will be eventually leveraged to translate localization results from local to global coordinates.

4 Semantic-based World Alignment

4.1 Coordinate System Description

Three coordinate systems are utilized during model alignment as depicted in Fig.4(a). The local SLAM and global GPS coordinate systems serve as the input and output systems, respectively. 3D GIS models are typically positioned in the Earth-Centered Earth-Fixed (ECEF) coordinate system[68], which can be geometrically converted to and from GPS coordinates using the Earth's radius and eccentricity [69]. Additionally, both ECEF and SLAM systems employ Cartesian coordinates, which allow for transformations from SLAM to ECEF using an affine transformation matrix, denoted as T_S^E in Fig.4(b).

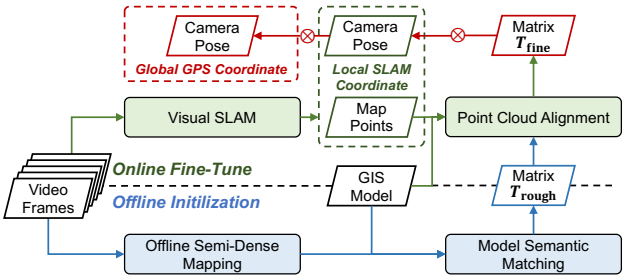


Figure 6: SWA Workflow.

4.2 Design Insight and Workflow

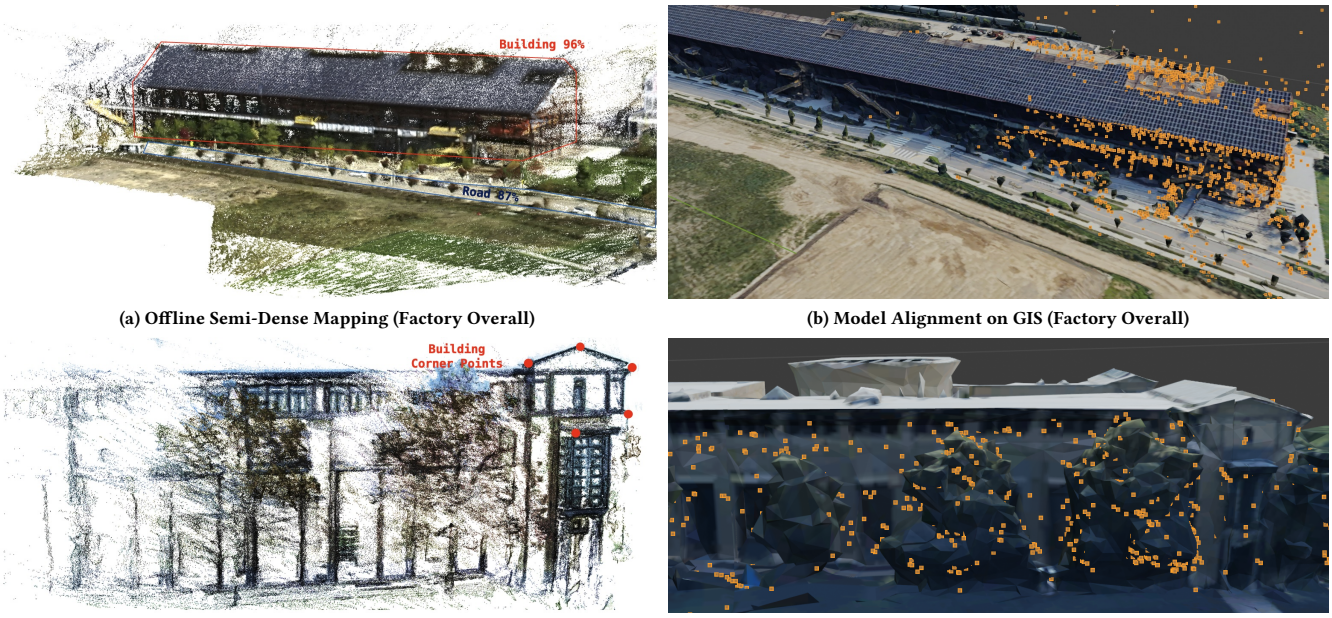
To calculate the vital T_S^E, a straightforward approach involves using the classic ICP algorithm to optimize it through point cloud registration[37]. Specifically, with point clouds in the SLAM coordinate system, $\{p_i^S\}$, and the GIS model point clouds in the ECEF system, $\{p_i^E\}$, ICP determines T_S^E by optimizing the function:

$$\min_{T_S^E, \sigma(i)} \sum_i \| T_S^E p_i^S - p_{\sigma(i)}^E \|^2, \quad (1)$$

where $\sigma(i)$ represents the optimal correspondence mapping between the point clouds. An affine transformation matrix is typically a 4×4 matrix with 12 degrees of freedom, allocated for scaling (3), shearing (3), translation (3), and rotation (3). Therefore, to optimize T_S^E from Eq.1, we require at least 12 pairs of well-matched points. However, even with recent enhancements [9, 38], ICP assumes substantial overlap between point clouds from two coordinate systems and minimal differences in scale, orientation, and density. These assumptions are invalid when significant disparities exist between the SLAM and GIS model point clouds, as illustrated in Fig.5.

To tackle this challenge, we introduce the SWA workflow depicted in Fig.6. Our key design insight is initially establishing coarse model alignment during the *offline initialization* phase using advanced 3D visual techniques, such as point cloud semantic segmentation[70, 71] and spatial matching[72]. Such methods overcome the challenges of aligning SLAM with GIS models across varied scales, orientations, and densities. The outcome of this phase is an initial transformation matrix, T_{rough}, which lays the foundation for further refinement. During the *online fine-tuning* phase, our enhanced ICP algorithm refines T_{rough} to achieve greater precision in model alignment, resulting in T_{fine}. This refined matrix facilitates converting points from the SLAM coordinate system to the ECEF and eventually the global coordinate system.

It is important to note that in TerraSLAM, the computationally intensive *offline initialization* phase is required only once and can be performed on a server. Specifically, for any scenario, once the first device conducts a site survey and collects environmental images, TerraSLAM will compute and store T_{rough} in its GIS database (detailed in §6). This matrix is calculated once and can be used continuously; thus, subsequent devices utilizing TerraSLAM in the same setting need not undergo offline initialization again. Despite environmental variations encountered by subsequent devices or the potential obsolescence of the GIS model, the offline phase provides only a preliminary matrix for refinement. Precise adjustments are made during the *online fine-tuning* phase, where T_{fine} is finalized and GPS coordinates are acquired in real-time. We describe the details of this two-step process in §4.3 and §4.4 below.



(a) Offline Semi-Dense Mapping (Factory Overall)

(b) Model Alignment on GIS (Factory Overall)

(c) Offline Semi-Dense Mapping (University Building Detail)

(d) Model Alignment on GIS (University Building Detail)

Figure 7: Semi-Dense Mapping and Alignment Results. (a) and (b) show the results of semantic segmentation and model matching on a large scale; (c) and (d) provide a zoomed-in view of a building, highlighting semantic contours and corners.

4.3 Offline Initialization

4.3.1 Semi-Dense Mapping. Given a series of video frames from a site survey, TerraSLAM initially runs an *offline semi-dense mapping* module. In this phase, advanced visual SLAM frameworks capable of generating and optimizing semi-dense or dense point clouds, such as ColMap[73] and other DNN-based SLAM algorithms[74], are employed to create a more detailed SLAM environmental model. As shown in Fig.5, compared to the sparse point clouds generated by real-time SLAM during the online phase (e.g., ORB-SLAM3[19], LSD-SLAM[57]), the point clouds produced in the offline phase are denser, offer stronger environmental representation, and contain richer color and semantic information. However, despite differences in appearance, both types of point clouds are located within the same SLAM coordinate system, as they are both optimized from video sequences and share similar original point and pixel scales. The transformation matrix T_S^E calculated using the semi-dense model can also be applied in the online phase with those sparse models.

4.3.2 Model Semantic Segmentation and Matching. We employ PointNet++[71] to perform semantic segmentation on both the generated dense model and the 3D GIS model. Subsequently, based on the segmentation results, we extract key objects that are easily distinguishable from various angles and remain static over long periods. In this work, our focus is on leveraging buildings and roads to roughly match the scale and orientation between the models as shown in Fig.7(a).

For model alignment, we initially calculate the corner coordinates for each segmented road and building, as exhibited in Fig.7(c). We then set the centroid of the largest building in the scene as the new reference origin. From this origin, we compute relative vectors to each corner, resulting in two vector sets for roads and buildings, $\{\boldsymbol{r}_i\}$ and $\{\boldsymbol{b}_i\}$. These procedures are applied to both the semi-dense model in the SLAM coordinate system and the GIS model in the ECEF coordinate system. Similar to Eq.1, we solve for T_{rough} by:

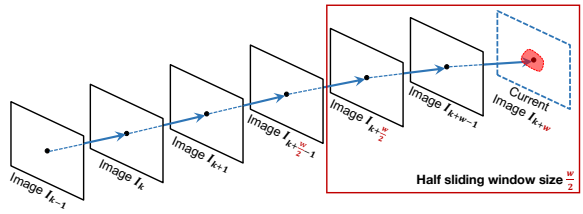
$$\min_{T_{\text{rough}},\sigma(i)} \sum_i \|T_{\text{rough}}\boldsymbol{r}_i^S - \boldsymbol{r}_{\sigma_r(i)}^E\|^2 + \|T_{\text{rough}}\boldsymbol{b}_i^S - \boldsymbol{b}_{\sigma_b(i)}^E\|^2, \quad (2)$$

where $\sigma(i)$ still represents the correspondence mapping.

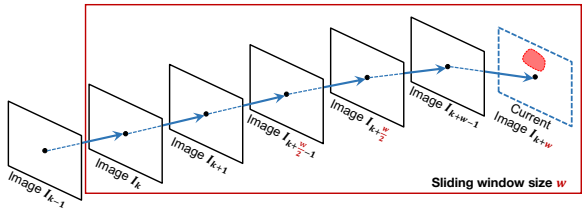
Furthermore, to address the variations in segmentation results across different models and inherent discrepancies between SLAM and GIS models, we implement random sample consensus (RANSAC) to enhance the robustness of Eq.2. RANSAC[18] involves running multiple iterations of Eq.2 by randomly selecting subsets of $\{\boldsymbol{r}_i\}$ and $\{\boldsymbol{b}_i\}$ during each round of optimization, and eventually choosing the optimal T_{rough} for the output.

4.4 Online Fine-Tuning and Globalization

4.4.1 Transformation Matrix Fine-Tuning. When subsequent mobile devices run visual SLAM, the online phase aligns the GIS model with the newly generated sparse SLAM model to optimize T_{fine}. As mentioned above, T_{rough} aligns the SLAM and ECEF coordinate systems in scale, origin, and orientation, enabling refinement using the ICP algorithm. To boost accuracy and efficiency, we adopt two strategies from the Open3D ICP library[75]: (i) we enhance traditional ICP with Fast Point Feature Histograms (FPFH) and normal vectors to assess the geometric similarity of point clouds; (ii) we down-sample the GIS model, focusing on corners and line features to enhance processing speed. The final model alignment is illustrated in Fig.7(b) and Fig.7(d), showing SLAM point clouds that are scaled, rotated, translated, and attached to GIS models. The online optimization module is activated every 200 frames during



(a) VO estimated camera location in a reliable area, allowing local mapping to proceed using only half the window size.



(b) VO estimated camera location is outside the reliable area, so it uses the original window size.

Figure 8: Illustration of Progress Local Mapping. The blue dashed line shows the intermediate result by VO; the red area denotes the reliable area based on historical trajectory.

SLAM operations, where it generates a batch of 3D map points. Optimizing T_{fine} based on T_{rough} can run in real-time with latency <2ms on a lightweight Ubuntu laptop (details in §6).

4.4.2 Global Coordinates Acquisition. After obtaining T_{fine}, we can transform the camera's local SLAM coordinates $\boldsymbol{p}$ into the ECEF as $T_{\text{fine}}\boldsymbol{p} = (x_E, y_E, z_E)$. Finally, leveraging the World Geodetic System 1984 (WGS-84[69]) earth ellipsoid model, we can obtain the camera's GPS coordinates:

$$\text{lon.} = \arctan 2(y_E, x_E), \tag{3-a}$$

$$N = \frac{a}{\sqrt{1 - e^2 \cdot \sin^2(\text{lat.})}}, \tag{3-b}$$

$$\text{lat.} = \arctan\left(\frac{z_E + e^2 \cdot N \cdot \sin(\text{lat.})}{\sqrt{x_E^2 + y_E^2}}\right), \tag{3-c}$$

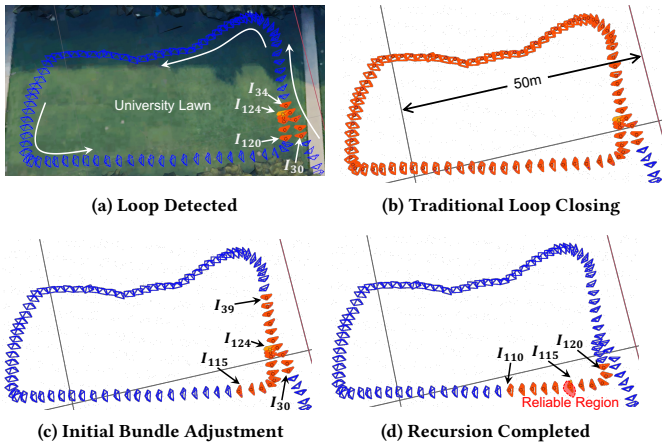
$$\text{alt.} = \frac{\sqrt{x_E^2 + y_E^2}}{\cos(\text{lat.})} - N, \tag{3-d}$$

where a is Earth's semi-major axis and e its first eccentricity. Calculating latitude should iteratively optimize N and lat. within the implicit expressions, Eq.(3-b) and Eq.(3-c), until convergence.

5 Hierarchical SLAM Acceleration

5.1 Design Goal and Insight

The design goal of our proposed *hierarchical SLAM acceleration* is twofold: Firstly, to enhance the speed of visual SLAM on mobile devices or cloudlets, matching the latency of existing GPS solutions on drones and robots (e.g., DGPS with latency around 15ms). Secondly, the acceleration modules should be portable, easily integrating as plug-ins into various SLAM systems, particularly those using keyframes and map points, with minimal code adjustments.



(a) Loop Detected **(b) Traditional Loop Closing**

(c) Initial Bundle Adjustment **(d) Recursion Completed**

Figure 9: Illustration of Recursive Loop Closing. The highlighted frames are leveraged for BA optimization.

To achieve these goals, we design three sub-modules within TerraSLAM to enhance all three SLAM components, as depicted in Fig.3. As mentioned in §2.2 and current practice [31, 32], the computationally intensive tasks in SLAM include the operations on visual features in VO, and the complex BA optimization in LC and LM. To handle the former, we leverage a *GPU-Accelerated VO* strategy. Specifically, the input image is divided into 256 patches, and each patch undergoes feature extraction, descriptor computation, and inter-frame feature matching in parallel on GPU cores. The acceleration is compatible with advanced GPUs on edge servers or cloudlets (e.g., NVIDIA RTX/A/H series) and lightweight GPUs on mobile devices (e.g., NVIDIA Tegra and Pascal series). Currently, we focus on ORB features, and TerraSLAM supports all ORB-SLAM variants[18–20, 61].

To accelerate BA optimization in LC and LM, we recognize that runtime heavily depends on window size w, which theoretically scales quadratically, $O(w^2)$. Existing approaches often use a fixed w (e.g., $w = 10$), but we propose dynamically adjusting w based on the current tracking state. For instance, stable tracking and accurate VO estimates negate the need for a large w, while poor VO tracking necessitates a larger w for detailed optimization. In TerraSLAM, we leverage this insight in LM and LC optimizations, introducing two hierarchical solutions as detailed below.

5.2 Progressive Local Mapping

We introduce a *progressive local mapping* module to accelerate the LM in SLAM. As shown in Fig.8, after VO estimates the pose for the current video frame I_{k+w}, LM conducts local BA using frames and map points within a sliding window of size w, enhancing pose accuracy for I_{k+w}. In TerraSLAM, we dynamically adjust w based on a reliable region $\mathbf{R}_{k+w}$ calculated from positions and velocities of frames from I_{k-w} to I_{k-1}:

$$\begin{aligned} \mathbf{R}_{k+w} &= \left\{\tilde{P}_{I_{k+w}} \in \mathbb{R}^3 \mid \left\|\tilde{P}_{I_{k+w}} - c_{k+w}\right\| \leqslant r_{k+w}\right\}, \\ c_{k+w} &= P_{I_{k+w-1}} + \frac{P_{I_{k+w-1}} - P_{I_k}}{w},\ r_k = \frac{P_{I_{k+w-1}} - P_{I_k}}{3w}. \end{aligned} \tag{4}$$

Algorithm 1: Recursive Loop Closing

Input: Set of keyframes $\{I_j \mid j \in [I_{\text{start}}, I_{\text{current}}]\}$; Sliding window size w

1 Perform initial bundle adjustment on frames from $[I_{\text{start}}, I_{\text{start}+w}] \cup [I_{\text{current}-w}, I_{\text{current}}]$.
2 **if** *Camera location of* $I_{current-w}$ *is within the reliable region from* $I_{current-w-1}$ **then**
3 Recursive Loop Closing completed.
4 **return**
5 **else**
6 $I_{\text{temp}} \leftarrow I_{\text{current}-w}$
7 **while** $I_{temp} > I_{start}$ **do**
8 Perform bundle adjustment on frame set $[I_{\text{temp}-w}, I_{\text{temp}+w}]$.
9 **if** *Camera location of* I_{temp-w} *is within the reliable region from* $I_{temp-w-1}$ **then**
10 Recursive Loop Closing completed.
11 **return**
12 **else**
13 $I_{\text{temp}} \leftarrow I_{\text{temp}-w}$

If the VO's intermediate result lies within $\mathbf{R}_{k+w}$, indicating stable motion (see Fig.8(a)), we reduce w by half, maintaining accuracy while cutting optimization time to a quarter. Conversely, an outcome outside $\mathbf{R}_{k+w}$ as illustrated in Fig.8(b) indicates dynamic movement, requiring full-window BA for detailed optimization.

In numerous applications like drone inspections and robotic surveying, where stable environmental observation is crucial, device acceleration typically does not experience sudden changes. According to our tests, most frames align with the scenario depicted in Fig.8(a). Overall, *progressive local mapping* can reduce the time spent on LM optimizations by approximately 50% (§7.5).

5.3 Recursive Loop Closing

In SLAM systems, loop closing significantly corrects cumulative errors and enhances positioning accuracy. However, it also contributes to prolonged localization latencies. Typically, for keyframes that activate loop closing, localization latency can spike to over 100ms. Taking a real LC example, as depicted in Fig.9(a), a drone flies over a lawn and counterclockwise around a building. The system identifies content similarities between video frames I_{120} to I_{124} and I_{30} to I_{34}, triggering the LC. Subsequently, SLAM performs a global BA optimization on these 95 frames (highlighted in Fig.9(b)) and their associated 3D map points. Similar to the local mapping discussed earlier, with an $O(n^2)$ algorithmic complexity, this process significantly increases system latency.

However, in TerraSLAM, we observe that the drone's flight trajectory is relatively stable, and SLAM tracks the camera's pose effectively, resulting in minimal cumulative error when loops are detected. This observation implies that a blanket optimization of all data within a loop may be unnecessarily heavy-handed. Based on this insight, we propose a *recursive loop closing* strategy, whose pseudocode is presented in Algorithm 1. The rationale behind it is that since localization errors progressively accumulate from the start (I_{30}) to the current (I_{124}) frame, our optimization should proceed in the opposite direction: starting with the most recent frames and map points detected in the loop to address larger errors first (Line 1 in Algorithm 1), and recursively moving the optimization window backward (Line 7–Line 13) until the cumulative error is within acceptable limits or the window reaches the start frame (Line 2 and Line 9).

Returning to the example in Fig.9, TerraSLAM initially utilizes the current frames where the loop is detected for BA optimization, specifically from I_{115} to I_{124} and I_{30} to I_{39}, as shown in Fig.9(c). After this initial optimization, if the position of I_{115} falls within the reliable region $\mathbf{R}_{115}$—calculated from I_{114} as described in §5.2—no further LC operations are needed. However, if I_{115} is outside $\mathbf{R}_{115}$, TerraSLAM will extend the optimization to include frames from I_{110} to I_{120} for further refinement, as depicted in Fig.9(d). Such recursive optimization continues moving backward until a frame falls within the reliable region or reaches the starting frame I_{30}.

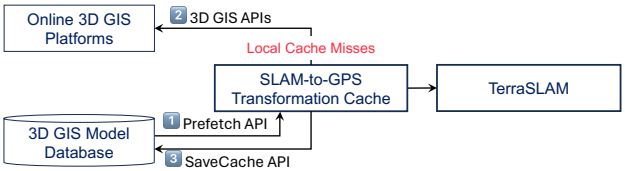


Figure 10: TerraSLAM's API for GIS Model Caching.

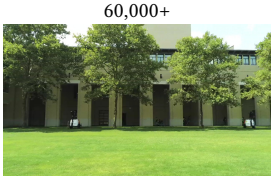
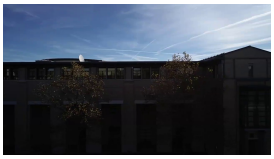
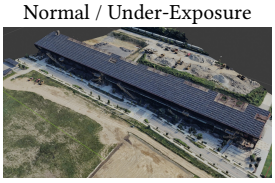
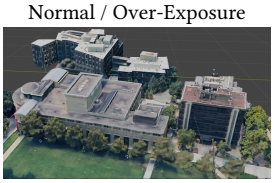
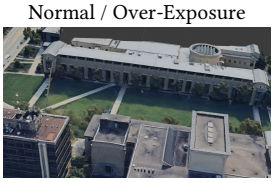
Figure 11: Field Study and GPS-RTK Setup on the Drone

6 Implementation

TerraSLAM's API for supporting different GIS sources. To effectively utilize various GIS data sources and enhance data reuse, TerraSLAM introduces a unified API comprising three key components as illustrated in Fig.10.

• `Pre-fetch API`. When TerraSLAM starts, the API first checks a local GIS database to determine if the required GIS model and corresponding SLAM-to-GPS coordinate transformation matrix are cached locally. If a cache hit occurs, these matrix will be directly leveraged by TerraSLAM for global positioning.

Table 2: Dataset Description

#	Factory-A (F-A)	Factory-B (F-B)	University-A (U-A)	University-B (U-B)
Total Frames	120,000+	120,000+	60,000+	60,000+
(Jul.–Sep.) Sample Image from Drone				
(Oct.–Dec.) Sample Image from Drone				
Lighting	Normal / Under-Exposure	Normal / Under-Exposure	Normal / Over-Exposure	Normal / Over-Exposure
3D GIS Model				
Model Staleness	Moderate-outdated	Outdated	Up-to-Date	Up-to-Date

• `3D GIS API`. If the required 3D GIS model is missing (i.e., a cache miss), this API fetches the necessary 3D GIS model from multiple online 3D GIS platforms like ArcGIS[36], Google Earth[35], or OpenTopography[76]. When the system starts, the API fetches 3D GIS models (or 3D point cloud) from GIS sources within a radius of $10km$ around the drone's rough initial GPS position. Once fetched, these models will be utilized by TerraSLAM.

• `SaveCache API`. After fetching these GIS models and computing the SLAM-to-GPS coordinate transformation, the API saves both the model and the corresponding transformation matrix back to the local GIS database for future usage.

TerraSLAM for drone localization. We implement TerraSLAM for real-time global geospatial localization on drones, using a DJI Phantom 4 Pro as shown in Fig.11b. The online localization part of TerraSLAM is built upon ORB-SLAM3 and is readily portable to other SLAM systems that utilize keyframes and map points. Due to the absence of onboard computing resources on the drone, we relayed its video frames in real-time through RTSP to an Ubuntu laptop configured with an Intel i7-8650U 1.90GHz 4-core CPU and a lightweight 256-core NVIDIA Pascal^{TM} GPU, which can be leveraged for VO acceleration. Recently, an increasing number of drones and robots have been equipped with advanced computing resources, such as the Nvidia Jetson TX2 and AGX Xavier. The laptop we currently use for processing, which has comparable computational capabilities, demonstrates that TerraSLAM can be directly deployed onboard in the future.

The offline part of TerraSLAM leverages ColMap[73, 77] for semi-dense mapping and PointNet++[71] for point cloud semantic segmentation, running on a cloudlet equipped with an AMD Ryzen Threadripper 7960X 24-Core CPU and dual Nvidia RTX 4090 GPUs. The offline computation, required only once, aligns the SLAM, ECEF, and GPS global coordinate systems. In our factory setting, the cold start to completion of this computation takes around 2 hours.

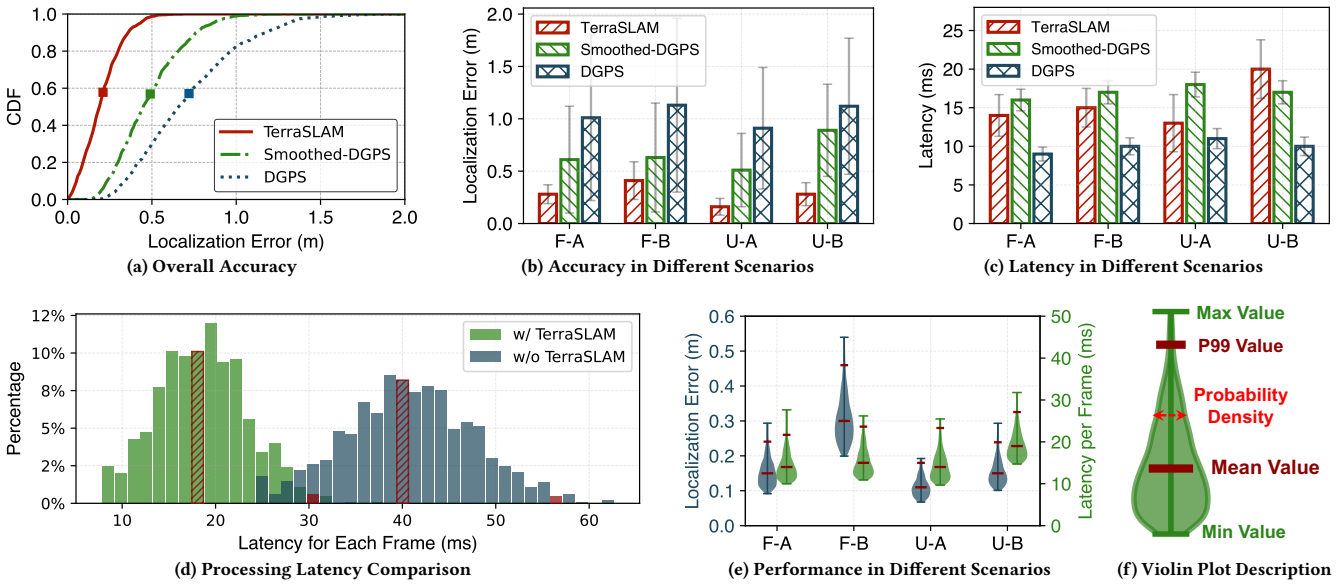
7 Evaluation

7.1 Experimental Methodology

Field Studies. We conduct drone global localization experiments across four representative scenes near a university and a factory (denoted F-A, F-B, U-A, U-B), as depicted in Fig.11a and Table 1. Our experiments, spanning six months, gather over 360,000 frames under varied seasonal, weather, and lighting conditions to evaluate the robustness of TerraSLAM. Additionally, the accuracy of GIS models varies across scenes: university models are regularly updated and highly precise, whereas factory models are less maintained. Specifically, model F-A is moderately outdated, and F-B is significantly outdated, missing updates like new buildings and lawn renovations. These challenging conditions further validate TerraSLAM's effectiveness in environments where GIS models are infrequently updated, demonstrating its robustness across more common scenarios.

Ground Truth Acquisition. We assess localization accuracy within the global ECEF coordinate system. Our ground truth is from GPS-RTK equipment, which includes a mobile rover with a dual-frequency GPS antenna mounted on the drone (Fig.11b), and a ground station (Fig.11c). We collect raw GPS data from both the mobile rover and ground station for post-processing kinematic (PPK) optimization. The results are then transformed into the ECEF coordinate system to serve as ground truth. A synchronizer is mounted on the drone's camera controller and provides μs-level synchronization accuracy between GPS and camera readings, crucial for aligning localization results and measuring system latency.

Baselines. To fairly demonstrate TerraSLAM's capabilities against standard GPS, which often has an error exceeding 3 meters, we also deploy differential GPS (DGPS) as a baseline. DGPS, crucial for mobile and vehicular localization, uses dual-frequency GPS antennas and differential algorithms to enhance accuracy by periodically updating GPS results with correction signals from cellular base



(a) Overall Accuracy (b) Accuracy in Different Scenarios (c) Latency in Different Scenarios

(d) Processing Latency Comparison (e) Performance in Different Scenarios (f) Violin Plot Description

Figure 12: Overall System Performance Comparison. (f) highlights the key components of a violin plot, which is used in the visualizations of Fig.12(e), Fig.13(c), Fig.13(d), and Fig.14(a).

stations. In our experiments, the RTK ground station provides correction signals every 5 seconds to the mobile rover for DGPS. We also use state-of-the-art technology to smooth DGPS results in real-time[41], allowing further comparison with TerraSLAM's accuracy. Additionally, to validate TerraSLAM's portability, we integrated it with various SLAM systems, including original ORB-SLAM3[19], PL-ORB[20], and ORB-SLAM3 with dense mapping (Dense-ORB), and compared their performance.

7.2 Overall Performance

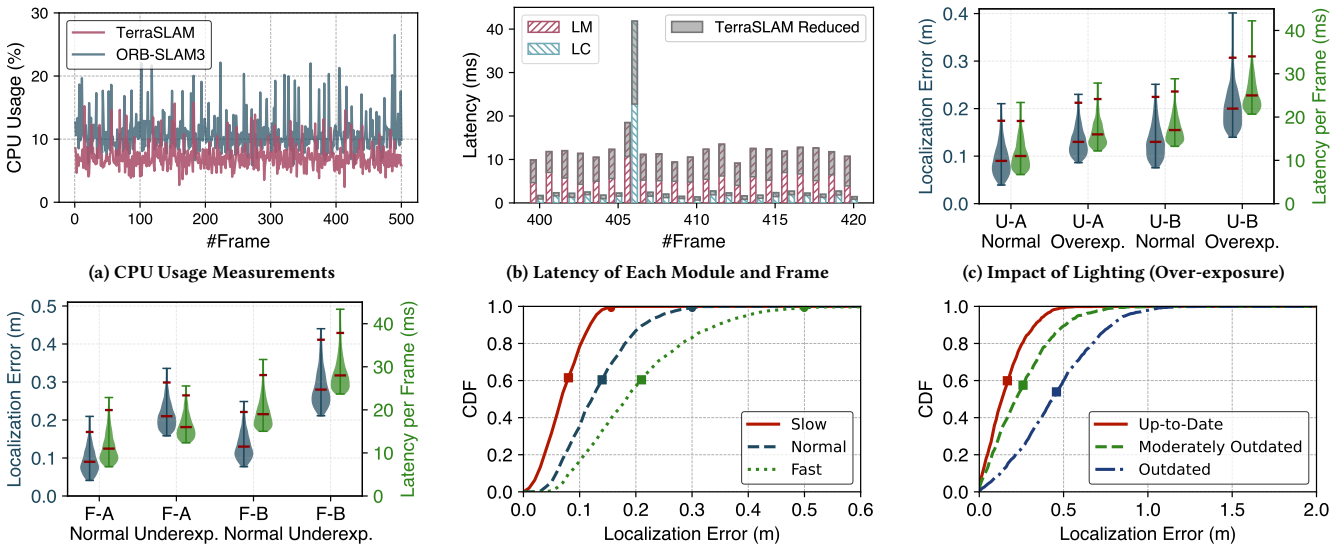
7.2.1 Accuracy. We first evaluate TerraSLAM against two GPS-based technologies, DGPS and DGPS enhanced with a smoothing function (Smoothed-DGPS). The results, depicted in Fig.12(a), demonstrate TerraSLAM's superior performance. With an average localization accuracy of 0.21m, TerraSLAM outperforms Smoothed-DGPS and DGPS by 62.4% and 71.2%, respectively. Furthermore, at the 99th percentile, TerraSLAM achieves a localization accuracy of 0.67m, compared to 0.98m for smoothed DGPS and 3.62m for standard DGPS, surpassing these systems by 31.7% and 81.5%, respectively.

We analyze TerraSLAM's localization accuracy across scenarios, as shown in Fig.12(b). In environments F-A, F-B, U-A, and U-B, TerraSLAM outperformed smoothed DGPS by over 45%, achieving localization accuracies of 0.23m, 0.39m, 0.15m, and 0.23m, respectively. Notably, TerraSLAM exhibited significantly lower variance, approximately 0.13, compared to 0.4 and 0.65 for smoothed-DGPS and DGPS. The results highlight TerraSLAM's superior global positioning capabilities, especially in GPS-challenged environments such as F-A and F-B, where excessive metal construction materials disrupt signal reception. In these settings, TerraSLAM achieves more reliable results than traditional advanced GPS solutions.

7.2.2 Latency. We then evaluate the latency of each system. We define latency for TerraSLAM as the processing time per frame, and for DGPS and smoothed-DGPS as the algorithm delay in optimizing GPS results based on correction signals. The measurements, shown in Fig.12(c), depict that TerraSLAM's average latencies in environments F-A, F-B, U-A, and U-B are 14.1ms, 15.2ms, 12.8ms, and 20.3ms, respectively. These results are higher than DGPS by around 5–8ms but lower than DGPS w/Smooth by 2–4ms. Notably, GPS-based solutions demonstrate consistent latency across different scenarios with minimal variance. However, as a vision-based solution, TerraSLAM experiences more variance in processing times due to varying visual textures across scenes, with variances noted as 2.7, 2.5, 3.6, and 3.9 in different scenarios.

We further focus on TerraSLAM itself to explore its impact on the efficiency of the original SLAM system (i.e., ORB-SLAM3 in this experiment). The results, displayed in Fig.12(d), reveal that the average latency per frame and P99 latency of ORB-SLAM3 decreased from 41.2ms and 57.2ms without TerraSLAM, to 16.7ms and 31.8ms with TerraSLAM, respectively. Additionally, the latency distribution became more concentrated. The above performance illustrates TerraSLAM's effectiveness in accelerating SLAM. The contributions brought by each module within TerraSLAM will be detailed in subsequent evaluations (§7.5).

7.2.3 TerraSLAM's performance in different scenarios. TerraSLAM's accuracy and latency across four different scenarios are illustrated in Fig.12(e), with average and P99 values previously reported in comparative experiments. Fig.12(e) clearly shows the worst localization accuracy in scenario F-B, with average and P99 values at 0.31m and 0.48m respectively. Such decreased precision is due to the outdated GIS model in the F-B area, as shown in Table 1, which affects coordinate alignment accuracy. Regarding latency, the U-B scenario, characterized by many overexposed images, necessitates more time-consuming visual feature extraction and matching. This results in an increase of 3ms and 8ms in average and P99 latency,



(a) CPU Usage Measurements

(b) Latency of Each Module and Frame

(c) Impact of Lighting (Over-exposure)

(d) Impact of Lighting (Under-exposure)

(e) Impact of Different Flight Speed

(f) Impact of 3D GIS Model Staleness

Figure 13: System Robustness and Resource Overhead Evaluation

respectively, compared to scenario U-A. More detailed analysis is presented in a later section §7.4.

7.3 Resource Overhead

We select a continuous segment of 500 frames from a flight trajectory in scenario F-A and record the CPU usage when running on the Ubuntu Laptop described in §6. As shown in Fig.13(a), TerraSLAM achieves an average CPU usage of 7.8%, compared to 14.8% for the original ORB-SLAM3, reducing CPU consumption nearly by half. Additionally, TerraSLAM experiences fewer CPU usage peaks, with a maximum usage of only 15.3%, compared to 28.2% in the original ORB-SLAM3. Given that SLAM is a foundational yet lower-level localization task for drones, reducing resource consumption in this area frees up computational resources to support higher-level applications.

We further provide a detailed zoom-in analysis from Frame #400 to #420 to illustrate how TerraSLAM accelerates the local mapping and loop closing modules. As shown in Fig.13(b), TerraSLAM reduces latency for pose optimization within the local mapping module by approximately 24.6% to 56.5% per frame. The PLM module (§5.2) dynamically selects optimization window sizes based on the stability of VO tracking. Typically, for frames with stable tracking status, the optimization window can be reduced by half compared to the original ORB-SLAM3, substantially decreasing both latency and computational resource consumption.

For the Loop Closing module, every frame has to undergo loop detection. Since this process consumes only a small fraction of computational resources (e.g., latency within 3ms), TerraSLAM does not propose a dedicated acceleration module for it. However, when a loop is detected, such as at frame 406, the RLC module (§5.3) will accelerate loop closing optimization. As illustrated in Fig.13(b), TerraSLAM notably reduces latency from 43.5ms to 22.4ms, achieving a 48.5% reduction, which considerably lowers the overall computational overhead of SLAM.

7.4 Robustness Evaluation

7.4.1 Impact of Lighting. As a vision-based technique, the robustness under varying lighting conditions is critical because changes in lighting significantly affect image quality. Figures 13(c) and 13(d) illustrate how different lighting conditions, specifically overexposure and underexposure, impact TerraSLAM's performance.

Fig.13(c) shows that overexposure due to direct sunlight hitting the drone's camera in the U-B scenario (as detailed in Table 1) reduces the clarity of visual textures, thereby significantly degrading SLAM performance. Under normal lighting, average localization accuracies for U-A and U-B are 0.09m and 0.13m, respectively. In overexposed conditions, these accuracies deteriorate to 0.13m and 0.2m, marking a decline of approximately 44% and 54%. In the U-B scenario, the P99 accuracy escalates from 0.28m to 0.45m, accompanied by tracking loss occurrences. Underexposure similarly impacts system accuracy, as shown in Fig.13(d). Compared to normal lighting, the average localization accuracy in dark (underexposed) environments for scenarios F-A and F-B decreases by 57% (i.e., from 0.09m to 0.21m) and 52% (i.e., from 0.14m to 0.29m), respectively.

Regarding latency, the instability of the visual odometry module in extracting sufficient visual features frequently necessitates more complex optimization processes in the local mapping module. For instance, joint optimization of tracking and mapping results over extended frame windows is required more often, increasing computational demand. Additionally, in ORB-SLAM, extracting enough visual features often involves adding computational layers to the image pyramid[61], further adding to computational overhead.

Under normal lighting conditions, Fig.13(c) shows average latencies for scenarios U-A and U-B at 11.2 ms and 17.1 ms, respectively. However, under overexposed conditions, latency increases to 16.3 ms and 25.2 ms, representing increments of approximately 46% and 47%. Fig.13(d) further illustrates latency increases under underexposure conditions in scenarios F-A and F-B, from 11.3 ms and 21.2 ms under normal lighting to 17.6 ms and 28.7 ms, respectively. In

particular, P99 latency exceeds 36 ms in both U-B (overexposed) and F-B (underexposed) scenarios, adversely affecting the real-time performance of TerraSLAM.

Despite these challenges, TerraSLAM demonstrates better accuracy compared to DGPS under overexposed and underexposed conditions. The broader adoption of HDR cameras could help mitigate these lighting-related performance issues.

7.4.2 Impact of Flight Speed. We further test the localization accuracy of TerraSLAM at varying flight speeds within the F-A scenario. As shown in Fig.11a, the total flyable length of the factory building area is around 400 meters. We conduct multiple flights over this distance in different durations and classify the average flight speed as *Fast* (i.e., > $10m/s$), *Normal* (5-$10m/s$), and *Slow* (< $5m/s$). The results, shown in Fig.13(e), indicate that at slow, normal, and fast speeds, TerraSLAM achieved average localization accuracies of $0.08m$, $0.14m$, and $0.21m$, respectively, with P99 accuracies of $0.16m$, $0.31m$, and $0.51m$. In practical applications such as industrial inspections and environmental surveying, drones/robots typically fly/run at moderate speeds (classified as *slow* or *normal* in this experiment). In these scenarios, TerraSLAM consistently delivers high-precision global localization.

As for high-speed flying (i.e., speed > $10m/s$), in our experiments, the drone is equipped with a 60 FPS rolling shutter camera, which is prone to motion blur and rolling shutter effects, making the visual features in the input video frames less distinct and thus reducing TerraSLAM localization accuracy. However, advancements in camera technology will enable the deployment of higher frame rate cameras (e.g., 100FPS), which can mitigate these issues.

7.4.3 Impact of GIS Model Staleness. The staleness of the GIS model critically impact the alignment accuracy between the SLAM and GPS coordinate systems in TerraSLAM, thereby significantly influencing its performance. To assess the effect of GIS model staleness on TerraSLAM's accuracy, we explore scenarios with varying model recency. As described earlier and shown in Table 1, U-A and U-B have up-to-date GIS models, while the F-B's model is mostly outdated with new buildings and updated lawns not reflected; F-A faces some changes in building contours.

Experimental results, depicted in Fig.13(f), reveal that under three degrees of model staleness, TerraSLAM achieves mean accuracies of $0.12m$, $0.19m$, and $0.43m$, with P99 accuracies of $0.58m$, $0.72m$, and $0.92m$, respectively. The model deterioration causes a substantial increase in mean localization error, tripling in scenarios where the model is outdated. However, even in such worst-case, TerraSLAM maintains higher accuracy compared to DGPS systems, underscoring its effectiveness against model aging. Technically, as long as the primary buildings and roads within the scene remain relatively unchanged, our proposed SWA can align the SLAM and GPS coordinates. This feature broadens TerraSLAM's applicability, as it operates effectively without relying exclusively on highly accurate or frequently updated GIS models. Models from widely covered sources such as Google Earth and other map tools are sufficient to support TerraSLAM's functionality.

7.4.4 Impact of Different Computing Devices. Table 3 illustrates the performance of TerraSLAM on running various devices as detailed in §6. When deployed on a mobile platform such as the Nvidia Jetson TX2, TerraSLAM achieves a mean accuracy of $0.32m$ and a P99 accuracy of $0.89m$, with mean and P99 latencies of $25.67ms$ and $43.28ms$ respectively. The performance improves on our Ubuntu laptop and reaches its peak on the cloudlet, showcasing the benefits of increased computational power. These improvements are primarily due to the increased efficiency in visual feature extraction and the faster execution of optimization algorithms that benefit from higher processing power. With the advent of new computing architectures like Neural Processing Units (NPU) on mobile devices, TerraSLAM is poised for promising real-time, high-accuracy localization across an even wider array of applications and environments.

Table 3: TerraSLAM Performance on Different Devices

Device	Cost	Accuracy (m)		Latency (ms)	
		Mean	P99	Mean	P99
Nvidia Jetson TX2	$700	0.32	0.89	25.7	43.3
Ubuntu Laptop	$2,000	0.21	0.67	16.8	31.9
Cloudlet (RTX 4090)	$10,000	0.18	0.62	9.2	21.2

Table 4: Average Point Alignment Error (m)

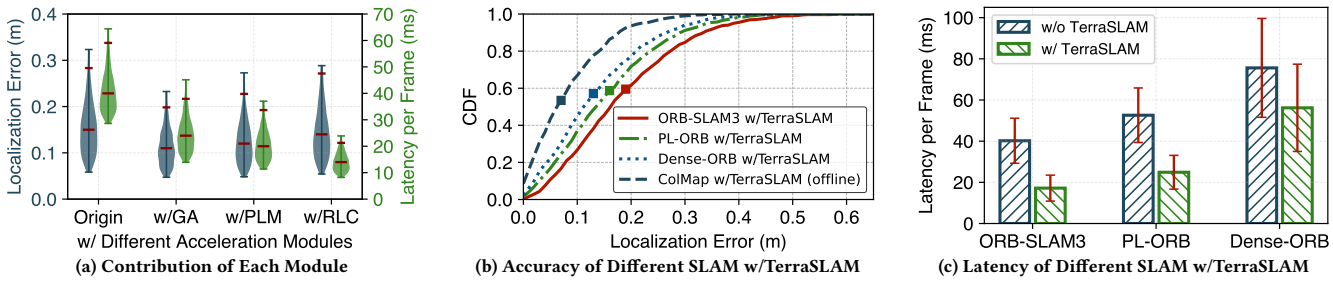
Matrix	F-A (Aug.)	F-A (Nov.)	U-A (Aug.)	U-A (Nov.)
$\boldsymbol{T}_{\text{rough}}$	0.57		0.44	
$\boldsymbol{T}_{\text{fine}}$	0.033	0.045	0.018	0.031

7.5 Ablation Study

7.5.1 Performance of SWA. In the F-A and U-A, we evaluate model alignment in August and November, respectively. We calculate the average point alignment error, derived by dividing the total residual of Eq.1 by the number of points in the generated SLAM point clouds. The offline part of SWA, responsible for calculating the initial $\boldsymbol{T}_{\text{rough}}$, is computed only once in Aug., with subsequent online refinements of $\boldsymbol{T}_{\text{fine}}$ based on $\boldsymbol{T}_{\text{rough}}$. As shown in Table 4, model alignment based on $\boldsymbol{T}_{\text{rough}}$ yields an average alignment error of $57cm$ for F-A and $44cm$ for U-A. After online fine-tuning, the error is reduced to $3.3cm$ and $1.8cm$ in Aug. and to $4.5cm$ and $3.1cm$ in November. The degradation observed in November is attributed to seasonal vegetation changes, causing accuracy discrepancies between the real-world and the GIS model (shown in Table 1). Focusing on robust features like buildings and roads in SWA ensures alignment stays within an acceptable range, supporting accurate localization.

7.5.2 Performance of Hierarchical SLAM Acceleration. The hierarchical acceleration module (§5) in TerraSLAM includes three sub-modules: *GPU-accelerated VO* (GA), *progressive local mapping* (PLM), and *recursive loop closing* (RLC), each enhancing a specific SLAM component. We begin with basic ORB-SLAM3, incrementally adding GA, PLM, and RLC, transforming it into the full TerraSLAM system. We document the system's accuracy and latency improvements following each module's integration.

Experimental results in Fig.14(a) show GA integration reducing average system latency from $40.3ms$ to $24.6ms$, improving accuracy from $0.15m$ to $0.11m$ through more efficient ORB feature processing. The introduction of PLC further reduced latency to $19.8ms$ by optimizing over $w/2$ time windows, slightly increasing accuracy to $0.12m$. RLC subsequently cut P99 latency from $31.2ms$ to



(a) Contribution of Each Module (b) Accuracy of Different SLAM w/TerraSLAM (c) Latency of Different SLAM w/TerraSLAM

Figure 14: Ablation and Portability Study

22.4ms, a 28% drop, by streamlining time-intensive loop closure detection. The final accuracy is maintained at 0.14m. Currently, our PLM and RLC's optimization is based on $w/2$. This parameter can be adjusted according to specific scenario requirements to achieve a better accuracy-efficiency tradeoff.

7.6 Portability of TerraSLAM

We evaluate TerraSLAM's adaptability by incorporating it into three distinct SLAM systems: (i) PL-ORB, which enhances the ORB-SLAM framework by integrating point and line features for VO; (ii) Dense-ORB, which optimizes dense point cloud maps rather than traditional sparse ones in LM; and (iii) ColMap, unlike the former two systems, requires global optimization and dense mapping for each frame and operates offline, typically providing camera pose inputs for NeRF[78], 3D Gaussian Splitting[79], etc.. As demonstrated in Fig.14(b), integrating TerraSLAM enables these systems to output global coordinates, with ColMap, Dense-ORB, and PL-ORB achieving average accuracies of 0.07m, 0.13m, and 0.16m, outperforming baseline (0.21m) by 66.7%, 38.2%, and 23.8%, respectively.

Fig.14(c) shows that integrating TerraSLAM on three real-time systems reduced average frame latencies to 17.2ms (baseline ORB-SLAM3), 24.9ms (PL-ORB), and 56.2ms (Dense-ORB), marking decreases of 57.2%, 53.2%, and 25.6% respectively compared to those without TerraSLAM. Overall, TerraSLAM could improve operational efficiency across different systems, demonstrating portability. However, it has not yet been optimized for additional visual features (as in PL-ORB) or complex optimization algorithms (as in Dense-ORB), resulting in less pronounced improvements than ORB-SLAM3. Future work may focus on extending optimizations to accommodate diverse features and algorithms.

8 Limitation and Discussion

We discuss the limitations and future directions to broaden the applicability of TerraSLAM.

3D GIS Model Availability. Currently, fine-grained 3D GIS models remain relatively limited. However, we anticipate significant growth in their availability due to rapid advancements in platforms like ArcGIS[36], Google Earth[35], Apple Map[67], and Microsoft Bing Map[80], which have greatly expanded GIS accessibility. Public datasets such as OpenTopography [76] also offer extensive global 3D point clouds, providing alternative GIS sources for improved usability. Additionally, pre-built GIS models are already common in industrial and military settings. Our work has proposed a well-defined API (described in §6) to leverage GIS models from diverse sources, positioning our system to readily benefit as GIS model coverage continues to expand.

3D GIS Model Quality. The accuracy and effectiveness of TerraSLAM closely depend on the GIS models' quality, precision, freshness, and real-world relevance. Our current experiments provide initial evidence that TerraSLAM can still outperform DGPS even when the GIS models are outdated or imperfect (§7.4.3). Future work includes (i) developing robust matching algorithms that leverage semantic information and advanced alignment methods to handle outdated or incomplete GIS models, and (ii) designing joint optimization algorithms to balance errors from SLAM point clouds and GIS inaccuracies. Another promising future direction is enabling TerraSLAM to leverage lighter-weight, up-to-date, and widely available image datasets, such as Google Street View[54] or OpenStreetMap[53] for global reference.

9 Conclusion

We have presented the design and implementation of TerraSLAM, the first system that transforms visual SLAM's localization results from local to global coordinate systems, positioning it as a viable alternative to GPS. We show 3D GIS models can act as a bridge to link the SLAM and GPS coordinate systems. At the core of TerraSLAM are two plug-in modules, *semantic-based world alignment* and *hierarchical SLAM acceleration*, that work hand-in-hand to enhance TerraSLAM's global localization accuracy and efficiency. Extensive evaluations in real-world drone localization scenarios demonstrate its superior performance.

Acknowledgments

We thank the anonymous reviewers and our shepherd for their thoughtful and constructive feedback that improved the presentation of this paper. This material is based upon work supported by the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W519TC-23-C-0003 and by the National Science Foundation under grant number CNS-2106862. The content of the information does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred. This work was done in the CMU Living Edge Lab, which is supported by Intel, Arm, Vodafone, Deutsche Telekom, CableLabs, Crown Castle, InterDigital, Seagate, Microsoft, the VMware University Research Fund, IAI, and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

References

[1] Guochang Xu and Yan Xu. 2007. *GPS: Theory, Algorithms and Applications*. Springer.

[2] Yuanxi Yang, Weiguang Gao, Shuren Guo, Yue Mao, and Yufei Yang. 2019. Introduction to BeiDou-3 navigation satellite system. *Navigation* 66, 1 (2019), 7–18.

[3] Peter Steigenberger and Oliver Montenbruck. 2017. Galileo status: orbits, clocks, and positioning. *GPS solutions* 21, 2 (2017), 319–331.

[4] Alfred Leick. 2015. *GPS satellite surveying*. Wiley.

[5] Melinda Laituri and Kris Kodrich. 2008. On line disaster response community: People as sensors of high magnitude disasters using internet GIS. *Sensors* 8, 5 (2008), 3037–3055.

[6] Wan Rahiman and Zafariq Zainal. 2013. An overview of development GPS navigation for autonomous car. In *2013 IEEE 8th Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, 1112–1118.

[7] Yunhao Liu, Zheng Yang, Xiaoping Wang, and Lirong Jian. 2010. Location, localization, and localizability. *Journal of computer science and technology* 25 (2010), 274–297.

[8] Weiwu Pang, Chunyu Xia, Branden Leong, Fawad Ahmad, Jeongyeup Paek, and Ramesh Govindan. 2023. UbiPose: Towards Ubiquitous Outdoor AR Pose Tracking using Aerial Meshes. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*. 1–16.

[9] Yuze He, Li Ma, Zhehao Jiang, Yi Tang, and Guoliang Xing. 2021. VI-eye: semantic-based 3D point cloud registration for infrastructure-assisted autonomous driving. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*. 573–586.

[10] Jay Farrell and Tony Givargis. 2000. Differential GPS reference station algorithm-design and analysis. *IEEE transactions on control systems technology* 8, 3 (2000), 519–531.

[11] Chris Rizos. 2007. Alternatives to current GPS-RTK services and some implications for CORS infrastructure and operations. *GPS solutions* 11 (2007), 151–158.

[12] Space News. 2024. America at Risk: High Impact GPS Jamming & Spoofing from Space. https://spacenews.com/america-risk-high-impact-gps-jamming-spoofing-from-space/. Accessed: 2024-12-08.

[13] Tyler Nighswander, Brent Ledvina, Jonathan Diamond, Robert Brumley, and David Brumley. 2012. GPS software attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security*. 450–461.

[14] Lei Yang, Yekui Chen, Xiang-Yang Li, Chaowei Xiao, Mo Li, and Yunhao Liu. 2014. Tagoram: Real-time tracking of mobile RFID tags to high precision using COTS devices. In *Proceedings of the 20th annual international conference on Mobile computing and networking (ACM MobiCom)*. 237–248.

[15] Guoxuan Chi, Zheng Yang, Jingao Xu, Chenshu Wu, Jialin Zhang, Jianzhe Liang, and Yunhao Liu. 2022. Wi-drone: wi-fi-based 6-DoF tracking for indoor drone flight control. In *Proceedings of the 20th annual international conference on mobile systems, applications and services (ACM MobiSys)*. 56–68.

[16] Jingao Xu, Guoxuan Chi, Zheng Yang, Danyang Li, Qian Zhang, Qiang Ma, and Xin Miao. 2021. FollowUpAR: Enabling follow-up effects in mobile AR applications. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (ACM MobiSys)*. 1–13.

[17] Danyang Li, Jingao Xu, Zheng Yang, Qian Zhang, Qiang Ma, Li Zhang, and Pengpeng Chen. 2022. Motion inspires notion: Self-supervised visual-LiDAR fusion for environment depth estimation. In *Proceedings of the 20th annual international conference on mobile systems, applications and services (ACM MobiSys)*. 114–127.

[18] Raul Mur-Artal and Juan D Tardós. 2017. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics* 33, 5 (2017), 1255–1262.

[19] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. 2021. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics* 37, 6 (2021), 1874–1890.

[20] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. 2017. Robust visual SLAM with point and line features. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1775–1782.

[21] Nicola Angelo Famiglietti, Gianpaolo Cecere, Carmine Grasso, Antonino Memmolo, and Annamaria Vicari. 2021. A test on the potential of a low cost unmanned aerial vehicle RTK/PPK solution for precision positioning. *Sensors* 21, 11 (2021), 3882.

[22] Jue Wang and Dina Katabi. 2013. Dude, where's my card? RFID positioning that works with multipath and non-line of sight. In *Proceedings of the ACM Special Interest Group on Data Communication (ACM SIGCOMM)*. 51–62.

[23] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. 2018. Through-wall human pose estimation using radio signals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7356–7365.

[24] Zheng Yang, Zimu Zhou, and Yunhao Liu. 2013. From RSSI to CSI: Indoor localization via channel response. *ACM Computing Surveys (CSUR)* 46, 2 (2013), 1–32.

[25] Chris Xiaoxuan Lu, Stefano Rosa, Peijun Zhao, Bing Wang, Changhao Chen, John A Stankovic, Niki Trigoni, and Andrew Markham. 2020. See through smoke: robust indoor mapping with low-cost mmwave radar. In *Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services (ACM MobiSys)*. 14–27.

[26] Jia Zhang, Rui Xi, Yuan He, Yimiao Sun, Xiuzhen Guo, Weiguo Wang, Xin Na, Yunhao Liu, Zhenguo Shi, and Tao Gu. 2023. A survey of mmWave-based human sensing: Technology, platforms and applications. *IEEE Communications Surveys & Tutorials* (2023).

[27] Yusra Alkendi, Lakmal Seneviratne, and Yahya Zweiri. 2021. State of the art in vision-based localization techniques for autonomous navigation systems. *IEEE Access* 9 (2021), 76847–76874.

[28] Jinrui Zhang, Huan Yang, Ju Ren, Deyu Zhang, Bangwen He, Ting Cao, Yuanchun Li, Yaoxue Zhang, and Yunxin Liu. 2022. MobiDepth: Real-time depth estimation using on-device dual cameras. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking (ACM MobiCom)*. 528–541.

[29] Zhiyuan Xie, Xiaomin Ouyang, Li Pan, Wenrui Lu, Guoliang Xing, and Xiaoming Liu. 2023. Mozart: A Mobile ToF System for Sensing in the Dark through Phase Manipulation. In *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services (ACM MobiSys)*. 163–176.

[30] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics* 32, 6 (2016), 1309–1332.

[31] Ali J Ben Ali, Marziye Kouroshli, Sofiya Semenova, Zakieh Sadat Hashemifar, Steven Y Ko, and Karthik Dantu. 2022. Edge-SLAM: Edge-assisted visual simultaneous localization and mapping. *ACM Transactions on Embedded Computing Systems* 22, 1 (2022), 1–31.

[32] Jingao Xu, Hao Cao, Zheng Yang, Longfei Shangguan, Jialin Zhang, Xiaowu He, and Yunhao Liu. 2022. {SwarmMap}: Scaling up real-time collaborative visual {SLAM} at the edge. In *19th USENIX Symposium on Networked Systems Design and Implementation (USENIX NSDI)*. 977–993.

[33] David J Maguire. 1991. An overview and definition of GIS. *Geographical information systems: Principles and applications* 1, 1 (1991), 9–20.

[34] Siyka Zlatanova, Alias Rahman, and Morakot Pilouk. 2002. 3D GIS: current status and perspectives. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences* 34, 4 (2002), 66–71.

[35] Google. 2024. Google Earth. https://earth.google.com/web/. Accessed: 2024-12-08.

[36] Esri. 2024. ArcGIS. https://www.arcgis.com/index.html. Accessed: 2024-12-08.

[37] Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 145–152.

[38] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. 2015. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE transactions on pattern analysis and machine intelligence* 38, 11 (2015), 2241–2254.

[39] Emerson Sie, Zikun Liu, and Deepak Vasisht. 2023. Batmobility: Towards flying without seeing for autonomous drones. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–16.

[40] Zhuoqun Cheng, Richard West, and Craig Einstein. 2018. End-to-end analysis and design of a drone flight controller. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37, 11 (2018), 2404–2415.

[41] Jeffrey J Early and Adam M Sykulski. 2020. Smoothing and interpolating noisy GPS data with smoothing splines. *Journal of Atmospheric and Oceanic Technology* 37, 3 (2020), 449–465.

[42] Zheng Yang, Chenshu Wu, Zimu Zhou, Xinglin Zhang, Xu Wang, and Yunhao Liu. 2015. Mobility increases localizability: A survey on wireless indoor localization using inertial sensors. *ACM Computing Surveys (Csur)* 47, 3 (2015), 1–34.

[43] Huatao Xu, Pengfei Zhou, Rui Tan, Mo Li, and Guobin Shen. 2021. Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems (ACM SenSys)*. 220–233.

[44] Sheng Shen, He Wang, and Romit Roy Choudhury. 2016. I am a smartwatch and i can track my user's arm. In *Proceedings of the 14th annual international conference on Mobile systems, applications, and services (ACM MobiSys)*. 85–96.

[45] Jingzhi Hu, Tianyue Zheng, Zhe Chen, Hongbo Wang, and Jun Luo. 2023. MUSE-Fi: Contactless muti-person sensing exploiting near-field Wi-Fi channel variation. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*. 1–15.

[46] Yunzhong Chen, Jiadi Yu, Yingying Chen, Linghe Kong, Yanmin Zhu, and Yi-Chao Chen. 2024. RFSpy: Eavesdropping on Online Conversations with Out-of-Vocabulary Words by Sensing Metal Coil Vibration of Headsets Leveraging RFID. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services (ACM MobiSys)*. 169–182.

[47] Dongfang Guo, Yuting Wu, Yimin Dai, Pengfei Zhou, Xin Lou, and Rui Tan. 2024. Invisible Optical Adversarial Stripes on Traffic Sign against Autonomous Vehicles. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services (ACM MobiSys)*. 534–546.

[48] Kun Qian, Zhaoyuan He, and Xinyu Zhang. 2020. 3D point cloud generation with millimeter-wave radar. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 4 (2020), 1–23.

[49] Yuze He, Chen Bian, Jingfei Xia, Shuyao Shi, Zhenyu Yan, Qun Song, and Guoliang Xing. 2023. Vi-map: Infrastructure-assisted real-time hd mapping for autonomous driving. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*. 1–15.

[50] Yang Lou, Yi Zhu, Qun Song, Rui Tan, Chunming Qiao, Wei-Bin Lee, and Jianping Wang. 2024. A First Physical-World Trajectory Prediction Attack via LiDAR-induced Deceptions in Autonomous Driving. *arXiv preprint arXiv:2406.11707* (2024).

[51] Fangqiang Ding, Xiangyu Wen, Yunzhou Zhu, Yiming Li, and Chris Xiaoxuan Lu. 2024. RadarOcc: Robust 3D Occupancy Prediction with 4D Imaging Radar. *arXiv preprint arXiv:2405.14014* (2024).

[52] Guoxuan Chi, Zheng Yang, Chenshu Wu, Jingao Xu, Yuchong Gao, Yunhao Liu, and Tony Xiao Han. 2024. RF-diffusion: Radio signal generation via time-frequency diffusion. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*.

[53] OpenStreetMap. 2025. OpenStreetMap. https://www.openstreetmap.org/. Accessed: 2025-03-18.

[54] Google. 2025. Google Street View. https://www.google.com/streetview/. Accessed: 2025-03-18.

[55] Dániel Kiss-Illés, Cristina Barrado, and Esther Salamí. 2019. GPS-SLAM: An augmentation of the ORB-SLAM algorithm. *Sensors* 19, 22 (2019), 4973.

[56] Google. 2025. Global localization with VPS. https://developers.google.com/ar/develop/geospatial. Accessed: 2025-03-18.

[57] Jakob Engel, Thomas Schöps, and Daniel Cremers. 2014. LSD-SLAM: Large-scale direct monocular SLAM. In *European conference on computer vision*. Springer, 834–849.

[58] Georg Klein and David Murray. 2007. Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 225–234.

[59] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. 2011. DTAM: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*. IEEE, 2320–2327.

[60] Tong Qin, Peiliang Li, and Shaojie Shen. 2018. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE transactions on robotics* 34, 4 (2018), 1004–1020.

[61] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. 2015. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE transactions on robotics* 31, 5 (2015), 1147–1163.

[62] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. 2000. Bundle adjustment—a modern synthesis. In *Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*. Springer, 298–372.

[63] Danyang Li, Yishujie Zhao, Jingao Xu, Shengkai Zhang, Longfei Shangguan, Qiang Ma, Xuan Ding, and Zheng Yang. 2024. Reshaping Edge-Assisted Visual SLAM by Embracing On-Chip Intelligence. *IEEE Transactions on Mobile Computing* (2024).

[64] Patrik Schmuck and Margarita Chli. 2019. CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams. *Journal of Field Robotics* 36, 4 (2019), 763–781.

[65] Danyang Li, Yishujie Zhao, Jingao Xu, Shengkai Zhang, Longfei Shangguan, and Zheng Yang. 2024. EdgeSLAM2: Rethinking edge-assisted visual SLAM with on-chip intelligence. In *Proceedings of the IEEE INFOCOM*.

[66] Sagar Jha, Youjie Li, Shadi Noghabi, Vaishnavi Ranganathan, Peeyush Kumar, Andrew Nelson, Michael Toelle, Sudipta Sinha, Ranveer Chandra, and Anirudh Badam. 2021. Visage: Enabling timely analytics for drone imagery. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom)*. 789–803.

[67] Apple Inc. 2024. Apple Maps. https://www.apple.com/maps/. Accessed: 2024-12-08.

[68] Yifeng Zhou, Henry Leung, and Martin Blanchette. 1999. Sensor alignment with earth-centered earth-fixed (ECEF) coordinate system. *IEEE Transactions on Aerospace and Electronic systems* 35, 2 (1999), 410–418.

[69] James A Slater and Stephen Malys. 1998. Wgs 84—past, present and future. In *Advances in Positioning and Reference Frames: IAG Scientific Assembly Rio de Janeiro, Brazil, September 3–9, 1997*. Springer, 1–7.

[70] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. 652–660.

[71] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv preprint arXiv:1706.02413* (2017).

[72] Liuhao Ge, Zhou Ren, and Junsong Yuan. 2018. Point-to-point regression pointnet for 3d hand pose estimation. In *Proceedings of the European conference on computer vision (ECCV)*. 475–491.

[73] Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

[74] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. 2020. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155* (2020).

[75] Open3D Team. 2024. ICP Registration. https://www.open3d.org/docs/0.7.0/tutorial/Basic/icp_registration.html. Accessed: 2024-12-08.

[76] OpenTopography. 2025. OpenTopography Dataset. https://opentopography.org/. Accessed: 2025-03-15.

[77] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

[78] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

[79] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

[80] Bing Maps. 2024. Bing Maps. https://www.bing.com/maps. Accessed: 2024-12-08.