RGB-D SLAM with Smartphone LiDAR

Henry Lewis, Altar Karataş, and V. P. B. Vadrevu
Supervisor: Yue Pan

Mobile Sensing & Robotics · M.Sc. in Geodetic Engineering · University of Bonn

February 22, 2026

***Abstract*—Modern smartphone camera clusters increasingly include LiDAR sensors for improved auto-focus in low-light conditions. Leveraging this sensor for robust mapping and localization is challenging in real-world environments due to the low spatial resolution of the depth data. The culmination of our Mobile Sensing & Robotics project, HandySLAM, attempts to overcome this challenge through the incorporation of a metric depth-inference model, resulting in a robust RGB-D SLAM pipeline that produces scale-accurate scene reconstructions. The use of upscaled depth maps in both the mapping and reconstruction stages allows our pipeline to achieve centimeter-level accuracy. Experimental results on the ScanNet++ dataset validate this claim in both trajectory estimation and scene reconstruction.**

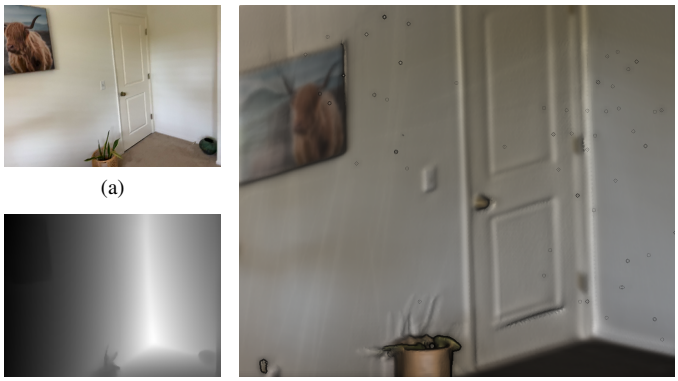
I. Introduction *(by Altar)*

SCENE reconstruction is a key goal for mobile robotics and augmented reality applications. The ubiquity of smartphones makes them a potentially ideal platform for mapping tasks. Until recently; however, smartphones had only monocular cameras, limiting their use in scale-accurate pose estimation. The integration of LiDAR scanners into smartphone camera clusters (notably, the iPhone Pro series) is becoming increasingly common. Such devices are capable of measuring metric depth alongside monocular color imagery, which is critical for scale-accurate reconstruction.

Previously, the dense mapping of a scene with metric scale required expensive, ad-hoc devices such as terrestrial laser scanners or mechanical LiDARs. Although the hardware is becoming increasingly common, the data produced is often low-resolution and noisy, making it difficult to use for precise 3D reconstruction.

A core challenge is the difference in resolution between the camera and the depth sensor. A modern smartphone camera captures detailed color images at high resolutions (often 1920×1440), but the accompanying LiDAR sensor typically produces a sparse depth map at a fixed low-resolution (256×192). Naive upscaling approaches, such as bilinear interpolation, result in the blurring of sharp object edges (see section IV-D). Visual features along these edges have incorrect depth values, which leads to localization drift and noisy reconstructions.

To address these challenges, we propose HandySLAM, a pipeline designed to improve the quality of mobile scanning in indoor environments. Instead of using standard resizing methods, we integrate a learning-based model called PromptDA [12] to infer higher-resolution depth data before tracking with ORB-SLAM3 [4]. By using the color imagery to guide the upscaling, PromptDA preserves true object boundaries and recovers geometry that is absent in the raw depth map. This provides the SLAM system with accurate, dense geometric constraints, allowing for robust tracking even in difficult indoor environments.



(a) (b) (c)

Fig. 1. HandySLAM receives color (a) and depth imagery (b) in series from the sensor platform (here, an iPhone 13 Pro), producing a scale-accurate reconstruction (c). The smartphone internally rectifies and aligns depth maps (256×192) w.r.t. the corresponding RGB frame (1920×1440). Despite the low-resolution depth, dense reconstruction is possible through depth inference.

The main contribution of this paper is a robust RGB-D SLAM pipeline for LiDAR-enabled smartphones that integrates learning-based depth upscaling to achieve high-fidelity, dense scene reconstructions. Figure 1 presents a sample of the pipeline's inputs and outputs. The HandySLAM repository is located at https://github.com/hankotanks/HandySLAM.

In this work, we make the following claims:

1) Our pipeline generates scene reconstructions with centimeter-level accuracy.
2) The integration of upscaled depth maps enables robust trajectory estimation in low-texture indoor environments.
3) The system generalizes effectively across diverse indoor topologies, as validated on the ScanNet++ dataset.
4) Inferring high-resolution depth improves the efficacy of trajectory estimation and density of scene reconstruction.

II. Related Works

A. Dense Mapping Systems Using Portable Devices *(by Altar)*

3D reconstruction from consumer-grade sensors was pioneered by KinectFusion [14], which leveraged Truncated Signed Distance Fields (TSDFs) for real-time volumetric integration. Subsequent works like BundleFusion [5] introduced

global consistency optimization to mitigate drift in large-scale environments.

The use of smartphones as mapping platforms necessitates commercial odometry frameworks (iPhones, for example, require ARKit). Often, computational efficiency and ease-of-use is prioritized over data fidelity, which complicates mapping.

While open-source solutions such as RTAB-Map [11] enable dense mapping on handheld devices, they remain constrained by the input resolution. When applied to sparse mobile LiDAR data, traditional fusion pipelines often struggle to recover fine structural details due to the inherently sparse depth measurements, necessitating densification or depth completion before integration into visual SLAM frameworks [18].

B. Visual and RGB-D SLAM (by Altar)

Robust trajectory estimation is critical for handheld scanning, where rapid motion often causes tracking loss. While dense SLAM systems like ElasticFusion [19] attempt to track camera pose directly against the surface model, they are computationally expensive and sensitive to geometric noise. Other map-centric methods like Gaussian Splatting SLAM [13], NICE-SLAM [23] and ReFusion [15] all suffer from the similar limitations, and are additionally sensitive to fast camera motion. The use of a handheld smartphone as a mapping platform guarantees the occurrence of rapid movements, which precludes the use of these map-centric methods. Instead, feature-based approaches are preferred for their stability.

ORB-SLAM3 [4] serves as a baseline in this domain, specifically for pure RGB-D imagery. Its multi-map Atlas system is particularly effective for our application; it allows the system to survive tracking failure by merging disparate maps once feature overlap is sufficient. However, ORB-SLAM3 is fundamentally a sparse system; it maintains a covisibility graph of discrete landmarks rather than a dense surface representation. Our work leverages this tracking stability while offloading the surface reconstruction to a dedicated dense mapping backend.

C. Depth Estimation (by Bhargava)

The resolution discrepancy between high-fidelity RGB cameras and sparse depth sensors is a persistent challenge in mobile robotics. Standard upscaling techniques, such as bilinear interpolation, fail to preserve high-frequency geometric details. Early "depth completion" methods, such as NLSPN [16], utilized Convolutional Neural Networks (CNNs) to propagate valid depth measurements to neighboring pixels based on image guidance.

More recently, foundation models trained on large-scale datasets have demonstrated strong zero-shot performance. Depth Anything [20] showed that relative depth with sharp boundaries can be estimated from single images. Building on this, Lin et al. proposed PromptDA [12], which conditions the foundation model on sparse metric points. This formulation allows the network to inject metric scale into the estimated depth map while preserving the semantic and geometric edges from the RGB domain, providing a high-fidelity input for dense SLAM.

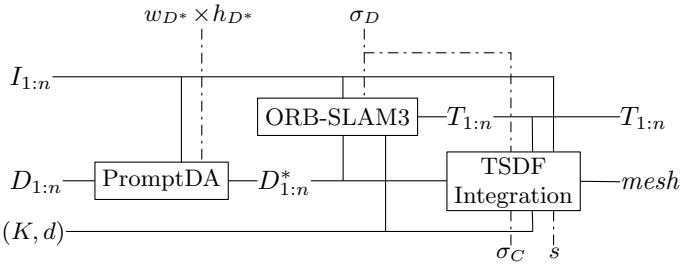


Fig. 2. Overview of the pipeline architecture and the process of mesh creation given an arbitrary scene $(I_{1:n}, D_{1:n}, K, d)$. Dashed lines denote parameters relevant to each stage (inferred resolution $w_{D^*} \times h_{D^*}$; depth threshold σ_D; confidence threshold σ_C (on scenes captured with the Stray Scanner); TSDF voxel length s). The three main pipeline stages are detailed in section III.

Other depth estimation approaches only support relative depth inferencing: Marigold [10] and MiDaS [17]. This makes them unsuitable for both localization and scene reconstruction. Both Metric3D v2 [9] and Depth Pro [3] provide metric results, but are outperformed by PromptDA.

III. Methodology *(by Hank)*

The scene reconstruction and trajectory estimation process can be conceptually divided into three stages (see Figure 2) given a scene consisting of a series of color images $I_{1:n}$, depth maps $D_{1:n}$, camera intrinsics K and distortion parameters d:

1) Raw depth maps $D_{1:n}$ from the smartphone are upscaled from their native resolution using a depth inferencing model. Both ScanNet++ scenes (recorded with an iPhone 13 Pro) and those captured over the course of this project (iPhone 15 Pro) provide depth images at a 256×192 resolution. PromptDA, the selected model, synthesizes high-resolution color imagery (1920×1440) and this low-resolution depth imagery to produce upscaled, 1008×756 depth maps $D^*_{1:n}$ for each frame [12].
2) ORB-SLAM3 receives the scene with upscaled depth $(I_{1:n}, D^*_{1:n}, K, d)$ and internally rescales both $I_{1:n}$ and K to match the resolution of $D_{1:n}$ [4]. Localization and ORB keypoint mapping proceeds frame-by-frame. Once the pose graph is finalized and all loop closures are resolved, relative pose propagation[1] is used to extract poses $\mathbf{T}_{1:n}$ from ORB-SLAM3's internal map.
3) Finally, each frame of RGB-D imagery and its associated pose is integrated into a TSDF volume using Open3D's `ScalableTSDFVolume` [22]. Prior to integration, the RGB-D image is filtered based on depth and confidence thresholds: σ_D and σ_C.

This pipeline produces a TSDF-derived mesh and a trajectory $\mathbf{T}_{1:n}$, where each $\mathbf{T}_i$ represents the sensor's 6DOF pose during the capture of (I_i, D_i).

HandySLAM supports data from two sources: ScanNet++ [21] and the Stray Scanner [2]. The former was used for

[1]ORB-SLAM3 only maintains keyframe (KF) nodes in the graph. At insertion time, non-KF nodes store their relative transform to the nearest KF node After graph optimization is complete, the poses can be retrieved by applying this relative transform to the final KF pose. The approach used in this project is based on the implementation of `ORB_SLAM3::System::SaveTrajectoryTUM`.

the quantitative analysis presented in section IV, while qualitative evaluation during development used the latter. Both datasets provide the following per-frame information, which is variously used through the three stages of the HandySLAM pipeline:

- RGB imagery I (1920×1440)
- Native resolution depth map[2] D (256×192),
- Ground truth camera pose[3] ${}^{w}\mathbf{T}_{c}$.

Each scene contains a description of camera intrinsics K. Additionally, the ScanNet++ dataset provides camera distortion parameters $d = \begin{bmatrix} k1 & k2 & p1 & p2. \end{bmatrix}^T$, while these values are unknown in the context of the Stray Scanner. A scene is described as a combination of its per-frame imagery and the parameters of the sensor: $(I_{1:n}, D_{1:n}, K, d)$.

A. Depth Inferencing

Raw depth imagery is upscaled using PromptDA [12], a depth inference model that utilizes an encoder-decoder architecture similar to Depth Anything [20]. Unlike the foundation model it's based on, PromptDA utilizes an aligned, low-resolution depth map in it's decoder stage to preserve metric scale in the output depth map. This aspect allows depth maps upscaled by PromptDA to be used for RGB-D SLAM and scene reconstruction.

For each frame, The depth inference process $(I, D) \longrightarrow D^*$ produces a 1008×756 depth map from a 1920×1440 RGB image and a 256×192 native depth map. The resulting depth map is still smaller than the input color imagery. However, this is a limitation of PromptDA, and still represents a large increase in available depth information for later pipeline stages.

The reliability of PromptDA's inferences suffers at long-range in the context of smartphone RGB-D. For this reason, upscaled depth maps are thresholded σ_D in the later stages of the pipeline.

B. SLAM

ORB-SLAM3 is a SLAM system for robust pose tracking and mapping. As frames are processed by ORB-SLAM3, the transformation from the previous frame to the current is computed using RANSAC over the frames' extracted ORB features [4].

Feature points are chosen using the FAST corner detection method. If the number of points extracted with the initial intensity threshold `iniThFAST` is less than `nFeatures`, the process is repeated with `minThFAST`. If the number of feature points produced is still insufficient, tracking has been lost.

Otherwise, the camera pose which maximizes feature point agreement across both frames is selected. Frames with high covisibility are redundant and likely to be removed by the local mapping thread, while those with low covisibility are promoted to keyframes (KFs). Upon promotion, a DBoW2 descriptor is computed for that node [7]. KF nodes are permanent in the pose graph, and are crucial for loop closures.

Resolving a potential loop closure requires two sets of three adjacent KF nodes to have a sufficiently small Hamming distance between their DBoW2 place descriptors[4]. If this criteria is met, a loop closure occurs, leading to a global adjustment within the current map.

ORB-SLAM3's multi-map Atlas is central to its inclusion in the HandySLAM pipeline. When tracking and subsequent relocalization fails, a new map is generated with its first KF nodes occurring after the tracking loss. The previous map is deactivated, but remains in memory.

Since a single DBoW2 database is shared across all maps in the Atlas, loop closure and map merging are effectively the same process. The key difference is that the former occurs between KF nodes of the same map, while the latter occurs between the active map and another in the Atlas. Merging an arbitrary map M_i with the active map in the Atlas $A = \{M_1, M_2, ..., M_{active}\}$ is equivalent to the removal of the stale map (1) and its unification with the M_{active} (2).

$$A = A \setminus M_i \tag{1}$$

$$M_{active} = M_i \cap M_{active} \tag{2}$$

The low-resolution and short range of native smartphone-derived depth maps increases the likelihood of tracking loss (even after upscaling with PromptDA). Automatic map merging provides recovery opportunities and enables the successful localization of challenging scenes containing fast and discontinuous camera motion.

With respect to the experimental results in section IV, SLAM has succeeded if and only if $|A| = 1$ after the final frame of imagery has been processed.

The ToF LiDAR sensors used in this project (IMX590 and IMX591 for the iPhone 13 Pro and iPhone 15 Pro, respectively) have an effective range of 5 meters[5]. The accuracy of readings from these sensors decreases as depth approaches this limit. Discarding depth values above a threshold σ_D improves the accuracy of the camera trajectory and reduces the number of outliers in the scene's reconstruction.

C. Obtaining Trajectory with Relative Pose Propagation

Non-keyframe nodes are ephemeral in the pose graph. They are maintained only until the next KF is identified, after which they are eligible for removal by ORB-SLAM3's local mapping thread [4]. Their pose is invalidated upon global bundle adjustment, which occurs during loop closure resolution and the merging of the active map with another in the Atlas.

[2]The 'native' depth map accessible using Apple's ARKit API does not represent the raw LiDAR measurements. Internally, sensor measurements are recorded at 24×24 resolution. They are internally upscaled using a closed-source machine learning approach [1]. In addition, the depth map returned by the API is redistorted w.r.t. the camera's intrinsics and aligned with imagery bounds.

[3]"Ground truth" in this context refers to the internal iPhone odometry, which is computed using a closed-source VIO method.

[4]Querying nodes in the map with high-similarity is trivial due to the hierarchical DBoW2 database. The 'tree' is traversed from root to leaf, choosing the intermediate nodes that minimize the Hamming distance between the query and the available leaves.

[5]This maximum depth was determined experimentally based on the maximum depth readings that could be produced from Apple ARKit and the Stray Scanner app. Documentation for these sensors (produced by Sony) is not publicly available.

In order to obtain a complete trajectory (spanning all frames, regardless of their KF status), the transformation ${}^{r}\mathbf{T}_i$ between each non-KF node F_i and the nearest KF K_r is recorded as they are processed. After SLAM is complete, the camera pose can be expressed in world coordinates by applying this transform to the reference KFs final pose:

$$ {}^{w}\mathbf{T}_i = {}^{w}\mathbf{T}_r \cdot {}^{r}\mathbf{T}_i \tag{3} $$

However, a KF can be removed by the local mapping thread if a majority of its feature points are contained in at least three other KFs. If this occurs, the removed KF's transformation to its replacement is recorded. The accumulation of these transformations is described as the 'spanning tree.'

This tree guarantees that for every non-KF F_i, there is some chain of KF-to-KF transforms $K_{r_0} \longrightarrow K_{r_1} \longrightarrow ... \longrightarrow K_{r_n}$ that ends with a globally consistent camera pose. Thus, in cases where F_i's reference KF has been removed from the map, its pose in world coordinates is found by traversing the spanning tree:

$$ {}^{w}\mathbf{T}_r = \prod_{j=0}^{n-1} {}^{r_j}\mathbf{T}_{r_{j+1}} \tag{4} $$

The complete trajectory is found by computing the reference KF's pose for each frame (4), then applying the relative transform from that KF to the frame (3).

D. Scene Reconstruction

RGB-D imagery and the camera poses $\mathbf{T}_{1:n}$ are combined to produce a mesh using Open3D's `ScalableTSDFVolume` [22]. The low accuracy of smartphone LiDAR necessitates masking of the RGB-D imagery.

Depth values between the upper bound σ_D and the maximum sensor range (5 meters) are ignored in the reconstruction process (5).

$$ \begin{bmatrix} I_{ij} \\ D^*_{ij} \end{bmatrix}_M = \mathbf{1}_M\{D^*_{ij} < \sigma_D\} \begin{bmatrix} I_{ij} \\ D^*_{ij} \end{bmatrix} \tag{5} $$

For each masked frame[6], all voxels in the camera's field of view are unprojected to the image plane (see appendix A), averaged with the frame's depth values, then reprojected onto the volume. As each frame is integrated, the signed distance values of the voxels become increasingly accurate.

Once integration has completed, a mesh can be extracted from the volume using marching cubes. Compared to naive projection of RGB imagery using depth map values, TSDF integration produces smoother geometry and less noise due to the aforementioned incremental averaging.

IV. Experiments *(by everyone)*

To evaluate the proposed pipeline, we utilize the ScanNet++ dataset [21], which provides high-fidelity 3D reconstructions of indoor environments. Unlike earlier datasets, ScanNet++ captures scenes with sub-millimeter precision using professional terrestrial laser scanners (Riegl VZ-400i). Crucially for our work, it includes synchronized RGB-D streams recorded with an iPhone 13 Pro, providing a realistic baseline for consumer-grade LiDAR data.

Two quantitative evaluations are performed in this section using ScanNet++:

- Trajectory drift between the iPhone's internally-computed trajectory and the HandySLAM-derived camera poses.
- Geometric accuracy of HandySLAM's scene reconstruction meshes against the ground truth meshes.

A. Selected Scenes for Evaluation (by Hank)

Eight scenes were used for analysis of the trajectory and reconstruction accuracy (referred to elsewhere by the beginning of their ID). Their frame counts and a short description of their contained geometry is presented in Table I.

TABLE I
Selected Scene Characteristics

ID	Frames	Description
281bc17764	7,122	Office
6b40d1a939	5,491	Bedroom
8a20d62ac0	3,685	Office; large bookshelf
49a82360aa	11,710	Living room
a4e227f506	6,180	Large lounge
c413b34238	7,895	Office; two workstations
e9e16b6043	3,207	Office; treadmill
e050c15a8d	9,508	Meeting room & offices

Due to the short range of the LiDAR scanner, interior scenes were selected for evaluation. Pipeline performance in outdoor environments was not benchmarked. Without IMU-integration, it is unlikely that SLAM in sparse outdoor environments would be possible or accurate using smartphone LiDAR.

B. Parameters (by Hank)

ORB-SLAM3's parameters were tuned with respect to the low spatial resolution of the depth maps. `nFeatures` was set to 2500 to overcome feature clustering and ensure that sufficient extracted features coincide with valid depth values. Since more initial candidate are produced, RANSAC produces a result with a higher inlier count, which results in more stable tracking.

`iniThFAST` and `minThFAST` were set to 12 and 5 respectively. Lowering these thresholds allows weaker intensity gradients to produce feature points. Combined with the higher feature count, these parameters improve the robustness of tracking in challenging environments[7].

ORB-SLAM3 uses an image pyramid for robust feature detection at different scales. Between the 8 levels of the pyramid (parameterized by `nLevels`, imagery is downscaled by a factor of 1.2. (`scaleFactor`). During ORB feature extraction, FAST corners are detected at all levels of this

[6]Scenes captured with the Stray Scanner are additionally filtered using the confidence map provided by Apple ARKit; however, ScanNet++ does not include this data. This combined masking function is defined as: $\mathbf{1}_M\{D^*_{ij} < \sigma_D \cap C_{ij} < \sigma_C\}$.

[7]Such as those found in the ScanNet++ dataset, where the camera frequently pans over featureless geometry.

pyramid. When descriptors are computed for these points, the patch size is scaled with respect to the pyramid level. This produces scale invariance across the feature set; future frames which contain the same geometry at different distances have analogous features with high similarity (although likely at different pyramid levels).

A distance threshold σ_D of 4 meters was used to mask the RGB-D imagery before SLAM and scene reconstruction. Depth values between $[4, 5)$ have a much larger receptive field as compared to readings close to the sensor. As a result, they contribute disproportionately to outliers in the scene reconstruction.

C. Trajectory Evaluation (by Altar)

To quantify the localization performance, Absolute Pose Error (APE) was computed for the selected scenes. Since the coordinate systems of the SLAM pipeline and the ground truth are initialized independently, we performed a $\mathrm{SE}(3)$ Umeyama alignment to recover the correct scale, rotation, and translation parameters before comparison.

TABLE II
ABSOLUTE POSE ERROR (RMSE) ON SCANNET++ SCENES

ID	**RMSE** [cm]	**Max** [cm]
281bc17764	11.1	57.8
6b40d1a939	11.3	50.3
8a20d62ac0	2.8	8.2
49a82360aa	15	51.7
a4e227f506	5.8	13.7
c413b34238	8.9	48.7
e9e16b6043	2.7	10.3
e050c15a8d	5.3	20.3

Evaluation utilized `evo`, a Python package for trajectory comparison [8]. The primary metric used is the Root Mean Square Error (RMSE) of the APE's translation, as it provides a robust measure of overall tracking stability.

The results, summarized in Table II, validate our second claim: the pipeline maintains robust tracking even without inertial data, relying solely on the geometric constraints provided by the upscaled depth maps.

1) Best Case: The system demonstrated its highest accuracy on scene's `e9e` and `8a2`, achieving 2.7 cm and 2.8 cm RMSE, respectively. Both scenes depict relatively small office environments with distinct geometric features in the room peripheries. The former contains a treadmill and plants; the latter: bookshelves. The lack of unadorned walls likely contributed to the high number of stable feature matches.

The maximum error remained below 9 cm, indicating that the scale drift was effectively constrained by the inferred depth maps.

2) Limiting Case: The highest error was observed in scene `49a`, which resulted in an RMSE of 15 cm. This scene contains over $11,700$ frames—more than three times the length of the best-performing scene. The error spike to 51 cm corresponds to feature-poor region where tracking became temporarily unstable.

In this scene, high-drift regions correspond to a small neighborhoods of the pose graph left sparsely-connected after a map merge. Their effect is local, and does not significantly degrade the usability of the trajectory for mapping tasks.

3) Drift Analysis: To verify that the system does not suffer from unbounded drift, we analyzed the error evolution over time. As illustrated in Figure 3 for all scenes, the error oscillates rather than growing monotonically. While drift naturally accumulates during exploratory phases (represented by the peaks), it decreases when the system revisits known areas or optimizes the pose graph. This confirms that the upscaled depth maps provide sufficiently distinct geometric constraints to support long-term consistency.

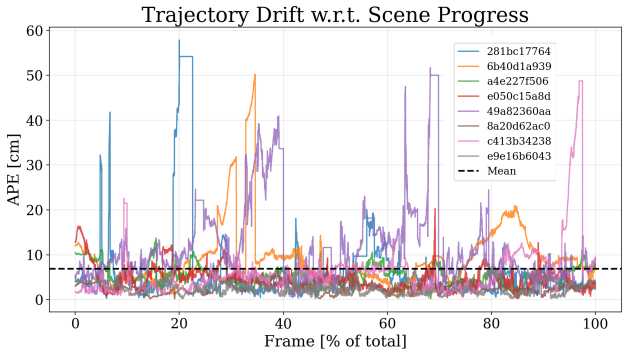


Fig. 3. Absolute Pose Error (APE) over time for all scenes. The error does not grow unbounded but oscillates, indicating that the backend successfully corrects accumulated drift through local bundle adjustment.

D. Ablation Study (by Hank)

Depth inferences given by PromptDA perform better than naively upscaling depth maps. Of the eight evaluated scenes, only five successfully produced a continuous trajectory using bilinearly upscaled depth maps. Those which succeeded had significantly higher trajectory drift than trajectories produced using inferred depth maps (see Table III).

TABLE III
MEAN TRAJECTORY DRIFT W.R.T UPSCALING APPROACH

ID	**Naive** [cm]	**PromptDA** [cm]
281bc17764	-	6.13
6b40d1a939	8.01	8.99
8a20d62ac0	6.93	2.63
49a82360aa	-	11.86
a4e227f506	5.59	5.33
c413b34238	-	6.15
e9e16b6043	2.49	2.39
e050c15a8d	8.28	4.63

The successful localization of multiple scenes with PromptDA preprocessing, which failed with naive upscaling, indicates that the inference stage of the pipeline is valuable; without it, HandySLAM's robustness suffers. The variance of the trajectory offsets is far larger in the bilinear case (see Figure 4).

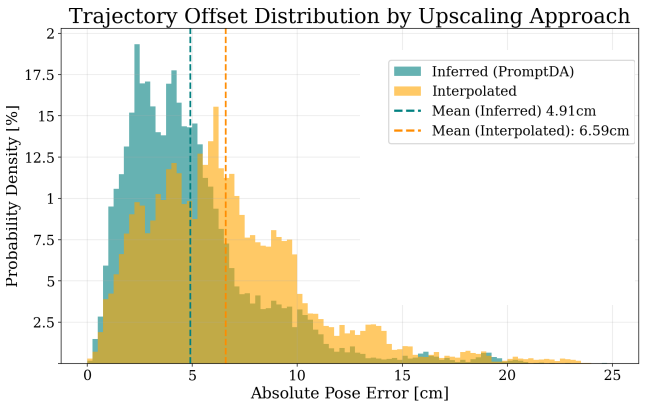


Fig. 4. The distribution of trajectory offsets across the five scenes which successfully localized. The variance of the trajectories derived from inferred depth maps is much lower than those using the bilinearly interpolated maps. Additionally, the mean drift is >1.5 cm closer to the ground truth.

E. Quality of Scene Reconstruction (by Bhargava)

To assess the geometric fidelity of the final reconstruction, we computed the Cloud-to-Mesh (C2M) distance between a uniform sampling of the TSDF-derived mesh and the ground truth mesh. Many scenes, notably `8a2`, `49a` and `6b4`, contained large, unobstructed windows. As a result, our volumetric reconstructions frequently contained exterior geometry that was not represented in the ground truth mesh. To ensure that the distances were computed over the same geometric extent, the sampled point cloud was cropped to the bounding box of the corresponding ground truth mesh.

The results, summarized in Table IV, demonstrate that HandySLAM achieves centimeter-level accuracy in all but the most challenging ScanNet++ scenes. Expectedly, `e9e` represents the best case with an average error of 2.05 cm. This corresponds to its low positional error (discussed in section IV-C). Exacerbated by the ground truth's overlarge extent, `a4e` represents the worst case, with an RMSE of 12.75 cm.

TABLE IV
RECONSTRUCTION ACCURACY (C2M DISTANCE)

ID	**RMSE** [cm]	**Mean** [cm]
`281bc17764`	7.63	5.19
`6b40d1a939`	6.49	4.59
`8a20d62ac0`	7.82	4.39
`49a82360aa`	12.56	8.93
`a4e227f506`	12.75	7.92
`c413b34238`	8.63	5.42
`e9e16b6043`	3.99	2.05
`e050c15a8d`	7.26	4.57

Figures 5 and 6 present visualizations of the reconstructed geometry. Our pipeline resolves thin features such as chair legs, computer monitors, or shelves; this would not be possible with naively upscaled depth maps. This visual quality aligns with the quantitative C2M results, where RMSE remains below the validation threshold for the majority of scenes.

V. CONCLUSION *(by Bhargava)*

This paper introduced HandySLAM, a pipeline for robust indoor RGB-D SLAM using data collected from LiDAR-capable smartphones. Experiments demonstrate that trajectories produced with the pipeline have a mean deviation of <9 cm from the internal iPhone 13 Pro solution. Similarly, scene reconstructions achieved centimeter-level accuracy w.r.t. ScanNet++ ground truth meshes. Further, the experimental results presented in section IV-D highlight the necessity of intelligent depth upscaling in the context of RGB-D SLAM with low-quality, commercial sensors.

In short, HandySLAM provides a well-structured and robust SLAM pipeline for smartphone sensors, and serves as a foundation for future work.

(a) (b)

Fig. 5. Overhead view of `281` ($\sigma_D = 4$ m, $s = 1$ cm), demonstrating coherence of reconstructed geometry. Additional views depict a close of up a desk (a) in the bottom-left of the room, and an office chair (b) in the top-left.

VI. FUTURE WORK *(by Bhargava)*

The ARKit API provides distorted RGB-D imagery; radial and tangential distortion coefficients are needed for rectification. Despite the smartphone's prior calculation of these parameters, they are not accessible through the API. Although ScanNet++ scenes contain `COLMAP` calibration results, self-collected data requires manual calibration. Much of the calibration backend (based on Kalibr) has been implemented, but it lacks proper integration in the HandySLAM pipeline [6]. For this reason, scene reconstruction quality from Stray Scanner data is degraded w.r.t. ScanNet++.

IMU-integration is likely to improve tracking robustness and pose accuracy as compared to the current visual-only approach. This extension is also partially implemented in the HandySLAM pipeline, but the lack of raw IMU data in the ScanNet++ dataset impeded progress.

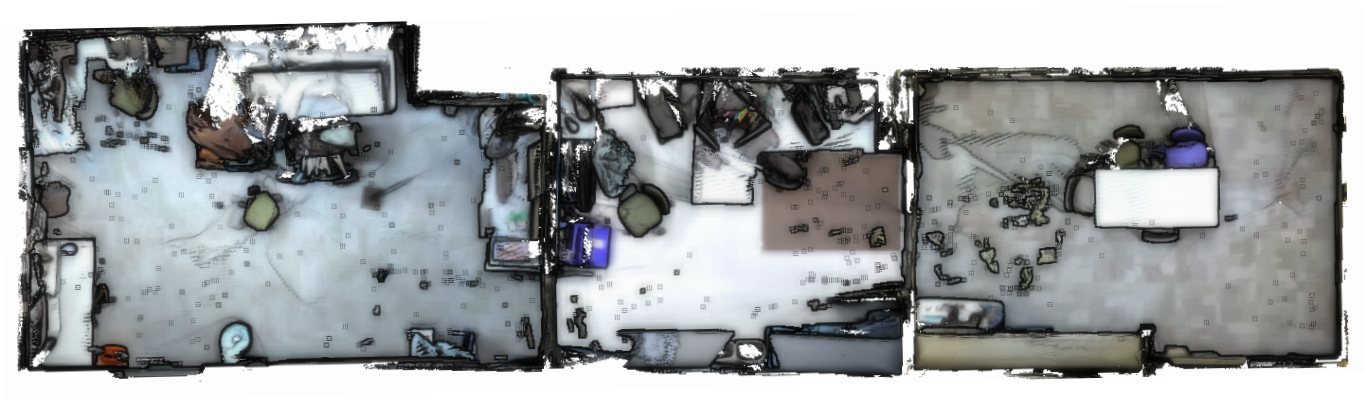


Fig. 6. Overhead view of `e05` reconstruction ($\sigma_D = 4$ m, $s = 1$ cm). This scene is the largest (and longest) in the evaluation set; it comprises three rooms. The volumetric reconstruction is accurate within 5 cm of the ground truth. Although `e05` has the highest frame count, its large extent results in missing geometry.

APPENDIX A
VOXEL UNPROJECTION TO IMAGE COORDINATES *(by Hank)*

To determine whether a voxel within a TSDF volume is within the camera's viewing frustum at a given pose ${}^{c}\mathbf{T}_{w}$, the voxel center must be unprojected from its world position p_w to its 2D coordinates on the image plane p_i using the camera's intrinsics K.

$$\tilde{p}_c = {}^{c}\mathbf{T}_{w} \cdot \tilde{p}_w$$
$$\tilde{p}_i = K \cdot \begin{bmatrix} \frac{x_c}{z_c} & \frac{y_c}{z_c} & 1 \end{bmatrix}^T$$

If $z_c < 0$, the voxel is behind the camera and must be skipped. The voxel is within the camera's field of view if and only if its unprojected image coordinates fall within the depth map's boundaries ($w_D \times h_D$): $p_i \in [0, w_D) \times [0, h_D)$.

REFERENCES

[1] Apple Inc. *Explore ARKit 4*. Apple Worldwide Developers Conference (WWDC 2020). 2020. URL: https://developer.apple.com/videos/play/wwdc2020/10611/ (visited on 02/18/2026).

[2] Kenneth Blomqvist. *Stray Scanner*. Version 1.3. 2021. URL: https://github.com/strayrobots/scanner.

[3] Aleksei Bochkovskii et al. "Depth Pro: Sharp Monocular Metric Depth in Less Than a Second". In: *International Conference on Learning Representations*. 2025. URL: https://arxiv.org/abs/2410.02073.

[4] Carlos Campos et al. "ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM". In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890.

[5] Angela Dai et al. "BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration". In: *ACM Transactions on Graphics 2017 (TOG)* (2017).

[6] Paul Furgale, Joern Rehder, and Roland Siegwart. "Unified temporal and spatial calibration for multi-sensor systems". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2013, pp. 1280–1286. DOI: 10.1109/IROS.2013.6696514.

[7] Dorian Galvez-López and Juan D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197. DOI: 10.1109/TRO.2012.2197158.

[8] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo. 2017.

[9] Mu Hu et al. "Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024).

[10] Bingxin Ke et al. *Marigold: Affordable Adaptation of Diffusion-Based Image Generators for Image Analysis*. 2025. arXiv: 2505.09358 [cs.CV].

[11] Mathieu Labbé and François Michaud. "Memory management for real-time appearance-based loop closure detection". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 1271–1276. DOI: 10.1109/IROS.2011.6094602.

[12] Haotong Lin et al. "Prompting Depth Anything for 4K Resolution Accurate Metric Depth Estimation". In: 2024.

[13] Hidenobu Matsuki et al. *Gaussian Splatting SLAM*. 2024. arXiv: 2312.06741 [cs.CV]. URL: https://arxiv.org/abs/2312.06741.

[14] Richard A. Newcombe et al. "KinectFusion: Real-time dense surface mapping and tracking". In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. 2011, pp. 127–136. DOI: 10.1109/ISMAR.2011.6092378.

[15] E. Palazzolo et al. "ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals". In: *arXiv* (2019). URL: https://arxiv.org/abs/1905.02082.

[16] Jinsun Park et al. *Non-Local Spatial Propagation Network for Depth Completion*. 2020. arXiv: 2007.10042 [cs.CV]. URL: https://arxiv.org/abs/2007.10042.

[17] René Ranftl et al. "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-

Dataset Transfer". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.3 (2022).

[18] Florian Sauerbeck et al. "RGB-L: Enhancing Indirect Visual SLAM using LiDAR-based Dense Depth Maps". In: *arXiv preprint arXiv:2212.02085* (2022).

[19] Thomas Whelan et al. "ElasticFusion: Dense SLAM Without A Pose Graph". In: *Robotics: Science and Systems*. 2015. URL: https://api.semanticscholar.org/CorpusID:16413702.

[20] Lihe Yang et al. *Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data*. 2024. arXiv: 2401.10891 [cs.CV]. URL: https://arxiv.org/abs/2401.10891.

[21] Chandan Yeshwanth et al. "ScanNet++: A High-Fidelity Dataset of 3D Indoor Scenes". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. 2023.

[22] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[23] Zihan Zhu et al. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.