Visual-Inertial 3D Reconstruction in Challenging Industrial Environments

Eyas Taifour
etaifour@stanford.edu

Zihan (Monica) Geng
zihangen@stanford.edu

Taarush Grover
taarush@stanford.edu

Abstract

VGGT [27] reconstructs the 3D point maps of most scenes accurately but degrades on the grayscale, low-texture, high-contrast images typical of construction sites. We study the factors driving this degradation and provide interventions that reduce reconstruction error without further training on new images and without supplying LiDAR as model input. We decouple the reconstruction effort into a camera-pose problem and a depth one. For pose, we design and train from scratch a side network that conditions VGGT's camera tokens on inertial measurements through cross-attention and regresses 6*-DOF poses. This removes VGGT's catastrophic failure on the peripheral cameras and drives relative translation error (RTE) to* 3 *cm, well below the* 13.2 *cm of an IMU-only baseline, though that baseline attains the lower relative rotation error (RRE) of* $1.7°$ *against our mean* $7.6°$*, a gap we attribute to the injection point of our intervention rather than the inertial signal. For depth, we implement BetterDepth [8], a diffusion method that refines coarse depth maps, which lowers per-frame Chamfer* L_2 *from VGGT's* $0.98\,m^2$ *to* $0.75\,m^2$*. We report this as promising rather than conclusive evidence since our synthesized ground-truth surrogate carries apparent residual artifacts that bound how far the diffusion refiner can be trusted.*

1. Introduction

Digital twins of active construction sites are necessary to support schedule management, operational hazard monitoring, cost optimization, and other functions. At the core of any such twin is an accurate representation of the site geometry, typically a reconstructed point cloud.

VGGT reframes how this reconstruction is performed: its multi-task learning objective enables it to jointly infer camera parameters, depth maps, point maps and 3D point tracks in a single forward pass. In comparison to classical methods which relied on long optimization processes, VGGT dramatically decreases this reconstruction time, paving the way for new applications in AR/VR, Robotics, Localization and Mapping.

Yet despite its strong multi-task performance, VGGT degrades on samples that deviate from its training dataset. Active construction site datasets are a prime example: they are characterized by lack of visual features and colors, which together prevent VGGT from accurately registering correspondences across multiple views, consequently leading to poorly generated poses. Additionally, the monocular depth maps generated by VGGT suffer from a smoothing effect that destroys the fine-grained details required for accurate point-cloud reconstruction.

These limitations motivate us to seek methods that can reduce VGGT reconstruction error, without finetuning on construction site images, and without supplying depth cues (LiDAR) as input. We break down the reconstruction problem into two subproblems, and define individual intervention for each as follows:

Poses accuracy: The input is a set of images and their cameras' accompanying inertial measurements. We build a fusion adapter composed of an IMU embedding mechanism, and a series of Transformer Encoders/Decoders that together query VGGT camera tokens and use our IMU keys/values to output new pose estimates as quaternions and translation vectors. We finally convert these vectors to $SE(3)$ matrices for visualization and evaluation.

Depth quality: The input is a set of images, which we process in two networks: a frozen VGGT produces coarse depth maps, which a refiner (BetterDepth) enhances using conditional latent diffusion, to output refined depth maps.

1.1. VGGT Background

Traditional 3D reconstruction is a multi-step pipeline composed of feature extraction, feature matching, iterative optimization, and monocular depth estimation. Each stage defines a self-contained objective, and chaining them accumulates both complexity and computational cost.

VGGT reduces this pipeline to a single forward pass. Its input is a set of S uncalibrated RGB images $I \in \mathbb{R}^{3\times H\times W}$ that together observe the same scene from C cameras. VGGT maps each image I_i to four outputs:

Camera parameters $g_i = (\mathbf{q}, \mathbf{p}, \mathbf{f}) \in \mathbb{R}^9$, the concatenation of a quaternion $\mathbf{q} \in \mathbb{R}^4$, a translation $\mathbf{p} \in \mathbb{R}^3$, and a field of view $\mathbf{f} \in \mathbb{R}^2$. VGGT places the principal point

at the image center, so g_i fully describes the i-th camera's extrinsics and intrinsics. Note that g_i is a 9-vector, not an element of $SE(3)$; the corresponding pose $T_i \in SE(3)$ is recovered by mapping $\mathbf{q}$ to a rotation matrix.

Depth maps, which assigns each pixel $\mathbf{y}$ a depth $D_i(\mathbf{y}) \in \mathbb{R}^+$ as seen from the i-th camera. A DPT [18] (Dense Prediction Transformer) head decodes multi-scale features from layers 4, 11, 17 and 23 into depth maps and confidence scores.

Point maps, which assigns each pixel $\mathbf{y}$ a 3D point $\mathbf{P}_i(\mathbf{y}) \in \mathbb{R}^3$ in the coordinate system of the first camera, taken as the world reference frame.

Keypoint tracking, which is delegated to a separate CoTracker [7] head. It is trained jointly with VGGT but is omitted from our project since our point cloud is assembled by unprojecting depth using (Eq. 1); and this head's output vectors provide neither pose nor depth information.

Each image is turned to patches then tokenized through a DINOv2 [13] backbone, then appended with a per-image camera token. The tokens pass through 24 stages of self-attention, alternating frame-wise attention (tokens of the same image) and global attention (all tokens of all images) at each stage. These representations are then routed to prediction heads according to their type: camera tokens downsampled to $g \in \mathbb{R}^9$ by four layers, while dense outputs (depth maps, point clouds, tracking maps) are produced by a DPT head over the image tokens. Additionally, the DPT head outputs confidence masks. VGGT is trained end-to-end with a multi-task regression loss on large posed multi-view datasets, real and synthetic, composed of images and scale-invariant depth.

1.2. Problem definition

A point cloud that has been reconstructed from multiple monocular depth maps is only as accurate as (a) the camera poses that register the depth maps into a common world frame, (b) the quality of the depth maps that place geometry along each ray, and (c) the scale of these depth maps in absolute units.

We focus on the first two aspects and treat each as a separate subproblem. The decoupling allows us to uniquely define the features we compute, the layers we intervene on, and the learning objective.

- **Camera pose**: We build an inertial odometry encoder that embeds IMU samples into a feature that describes the rig's displacement and change in orientation over a window of frames. Because inertial measurements depend on motion rather than scene appearance, this feature carries pose information regardless of visual cues; we inject it into the latent space of VGGT's camera tokens and read out per-camera $\mathbb{R}^7$ vectors, which we transform to a 6-DOF pose representation. The task we set is to reduce relative rotation and translation errors (RRE/RTE).
- **Depth maps**: For a pixel $\mathbf{y}$ in view (t, c) with depth $D_{(t,c)}(\mathbf{y})$, intrinsics $\mathbf{K}_c$, and extrinsics $(\mathbf{R}_{(t,c)}, \mathbf{p}_{(t,c)})$, the back-projected world point is:

$$\mathbf{X}_{(t,c)}(\mathbf{y}) = \mathbf{R}_{(t,c)}^{\top}\big(D_{(t,c)}(\mathbf{y})\,\mathbf{K}_c^{-1}\tilde{\mathbf{y}} - \mathbf{p}_{(t,c)}\big)\,, \quad (1)$$

  where $\tilde{\mathbf{y}}$ is the homogeneous pixel coordinate [25]. The task we set is to reduce the Chamfer L_2 distance of the generated depth maps.

2. Related Work

3D reconstruction is classically performed by Structure-from-Motion pipelines such as COLMAP [24], which rely on iterative optimization to arrange point positions and camera poses simultaneously. Recent advancements in deep learning have introduced models that can perform 3D reconstruction in a single forward pass. This feed-forward approach (colloquially known as 3R) was pioneered by DUSt3R [30], which predicts dense 3D point maps from image pairs. MASt3R [11] improved dense correspondence and presented a strategy that can handle thousands of images. These models have culminated in VGGT, which employs a single transformer architecture to compute a multi-task objective (1.1) to recover 3D attributes in a single pass.

LiDAR-VGGT [28] extends VGGT by fusing LiDAR geometry at inference to correct scale and pose failures in difficult scenes. It is the closest prior work to ours, though it relies on LiDAR as a model input, whereas our work only uses images and IMU samples.

Fisheye3R [4] proposes an adaptation framework that can extend VGGT to fisheye imagery while remaining backwards compatible. This strategy is not accompanied by an implementation and was one of our initial project considerations, but required large-scale data and compute, which were difficult to secure.

Visual Inertial Odometry estimates a camera's trajectory from image sequences and inertial measurements. Classical VIO solutions such as VINS-Mono [16] pre-integrate measurements between keyframes to produce relative motion constraints, which are usually used in an optimization problem in SLAM solutions.

Although VIO has undergone a rapid evolution driven by feed-forward networks, none of the major 3R models (VGGT [27], DUSt3R [30], MASt3R [11], π^3 [31], CUT3R [29], Fast3R [32]) incorporate inertial measurements, except for MASt3R-Fusion [35], which integrates MASt3R's point map regression with IMU and GNSS data in a tightly coupled factor graph SLAM. Our work evaluates the addition of an IMU encoder that produces tokens integrated into the visual transformer via cross-attention. Unlike MASt3R-Fusion, our work does not rely on pose graphs.

Point cloud registration is a set of methods that estimate the rigid transform that best aligns two sets of 3D points. RANSAC [5] provides a robust framework for estimating this transform in the presence of outliers by iteratively fitting a model to random subsets of correspondences. ICP [1] then refines the alignment by iteratively minimizing point-to-point distances between the two clouds. KISS-ICP [26] introduced a method that simplifies this pipeline into a minimal, robust formulation that works reliably across sensor types and environments without per-dataset tuning, making it well suited for the LiDAR sequences in our dataset.

Monocular depth estimation is the attempt to predict an image's pixel depth by relying on a single view. MiDaS [19] introduced affine-invariant depth training to handle scale ambiguity across datasets, enabling models to generalize to new scenes without metric supervision. DepthAnything [33] builds on this foundation and scales it to large unlabeled datasets, producing relative depth maps that are accurate in structure but lack metric scale. UniDepth [15] addresses this by conditioning on camera intrinsics, allowing it to predict metric depth directly from a single image.

DPT established the use of vision transformers for this task, producing dense depth predictions by aggregating features across multiple scales. Since VGGT employs DPT, it produces depth maps that exhibit similar limitations, such as scale-invariance and blurred depth boundaries.

BetterDepth [8] is a diffusion-based depth enhancement approach that refines existing depth predictions. It's a conditional latent model that takes a coarse depth prediction from any feed-forward network and refines it via iterative denoising, which produces sharper edges and better fine-grained structure. BetterDepth is a strategy built on Marigold [9], which leverages the rich visual knowledge stored in Stable Diffusion [21] to enhance depth maps.

3. Data

The Hilti 2023 SLAM Challenge dataset [12] benchmarks SLAM in construction sites and exposes difficulties rarely seen in the curated scenes that dominate public benchmarks. It contains video key frames of building interiors during active construction; dominated by concrete and plasterboard with repetitive, texture-less, visually ambiguous structure; lighting is uneven and poor with many high-contrast regions; and the frames are monochrome. These characteristics make the Hilti dataset perfect for the study of enhancing pose and depth estimates in the absence of high-fidelity image sources.

The dataset comprises twelve sessions collected at three sites, each recorded from a handheld rig carried by an operator through the work zone. The rig integrates five global-shutter, monochrome fisheye cameras, an embedded 6-DOF IMU, alongside a LiDAR 360° sensor. The devices' clocks are hardware-synchronized but they run at different rates and their timestamps never coincide: the IMU streams at 399.38 Hz, the LiDAR at 10 Hz, and the cameras at 40 Hz. Per-sample timing deviation is negligible for the IMU and cameras ($\sigma_{\text{IMU}} = 0.0048$ ms, $\sigma_{\text{camera}} = 0.0003$ ms) and larger for the LiDAR ($\sigma_{\text{LiDAR}} = 0.12082$ ms), which we found correlates with sudden jerks. We've opted to disregard this correlation in an attempt not to leak LiDAR data to our input.

Each session is recorded in a ROS 1 Noetic bag file composed of timestamped messages across the seven sensor topics, which we extract to disk. Calibration files are also provided and contain information about camera intrinsics, lens distortion coefficients, and device-to-rig-origin transforms. In total, the twelve sessions amount to $\approx$ 33 minutes of data: $520{,}000$ images (H=$540, W$=720), $780{,}000$ IMU samples, and $20{,}000$ LiDAR sweeps. We split the total dataset to 80% training, 20% validation, and keep the data time-ordered.

3.1. Preprocessing

Images: VGGT was trained on perspective images and therefore degrades on Hilti's wide field-of-view ones. We therefore undistort the images to rectilinear pinhole viewpoints and accept the loss of peripheral coverage. We duplicate the single monochromatic channel to RGB, apply standard ImageNet normalization, and center-crop each image to H'=$532, W'$=714.

Since the rig is handheld and the data is acquired at active construction sites, we implemented a pipeline to detect and remove humans from images and their accompanying point clouds in LiDAR. We used a two-stage human-presence filtering pipeline. First, Grounded-SAM2 [20] was applied to the camera images with prompts for people, limbs, helmets, and safety vests to produce a binary human mask with a 4-pixel boundary. Then, using the camera-to-LiDAR transformation matrix, we project the LiDAR points into the camera frames and remove points inside the corresponding masks. Manual inspection of the sequence images reveals the only dynamic objects in the scene are construction workers (see Fig. 9).

IMU: We estimate the gyro bias and gravity vector from the first session's stationary period ($\approx 10s$) and subtract them, rotate samples into the cam_0 frame via the provided extrinsics, and standardize each channel using μ, σ calculated once from the training split.

LiDAR preprocessing: We pre-process LiDAR twice to support the requirements of each subproblem as follows:

a. Pose supervision from LiDAR: Supervising the pose estimation network requires a camera-to-world pose $T^*_{(t,c)} \in SE(3)$ for every camera c at each frame timestamp τ_t, which we derive from the LiDAR in two stages. **(a)** We

calculate the relative transform $\Delta T \in SE(3)$ between consecutive LiDAR sweeps by point-cloud registration: scans are voxel-downsampled, described with FPFH [22] features, coarsely aligned by RANSAC over mutually filtered correspondences, and refined by point-to-point ICP. Each ΔT is cached once and reused across overlapping windows. **(b)** We interpolate new relative transforms of the translation vector $\mathbf{p}$ linearly and the rotation matrix $\mathbf{R}$ by SLERP along the $SO(3)$ geodesic so the result stays on the manifold. We finally rebase the LiDAR-interpolated motion onto each camera coordinate by chaining back to cam_0.

b. Dense point maps for depth estimation: Similar to the previous section, we pre-process the LiDAR point clouds using point cloud registration to create a global point cloud of the construction site. For each camera timestamp t, we collect the 20 LiDAR sweeps preceding the timestamp as well as 20 succeeding it. We use RANSAC and ICP to build an aggregate point cloud that covers a range of 40 sweeps, centered around the sweep closest in time to the camera frame. We tessellate the resulting point cloud to generate approximate surface normals, and rely on a simple ray cast renderer to determine occlusions. We delete the occluded points and rasterize the remaining ones to generate a synthetic depth map.

4. Methods

We freeze VGGT and cache to disk the feature representations of the Hilti images, which we further process twice to address each subproblem individually. Although different architecturally, each method follows the same recipe: After the initial network is built, we evaluate its capacity to overfit, followed by a search for optimal hyperparameter settings. After training, we input invalid and out-of-distribution tensors to the networks (*e.g.* all zeros, noisy IMUs, noisy depth maps) to evaluate the network's response. We log quantitative training artifacts using a mix of tensorboard and WandB, and qualitative ones (*e.g.* depth maps, confidence scores of pre-selected evaluation samples) on disk to inform our experiments. Each network is trained individually.

4.1. IMU cross-attention fusion adapter

During pre-processing, we built a camera-to-world transform $T_{(t,c)}$ for every camera $c \in \{0, \dots, C-1\},\ C = 5$, at each frame $t \in \{1, \dots, n\}$ of an n-frame window (n=4). Using our initial settings, this represents a total of 20 images taken by 5 cameras per training sample. We anchor this window so that the origin is cam_0 at the first frame, $T_{(1,0)} = \mathbf{I}$, and predict the remaining poses. We task the network to generate $(n \cdot C) - 1$ poses. We represent poses as a $\mathbb{R}^7$ vector of a 6-DOF pose composed of a quaternion and a translation vector to support our multi-task loss function.

Tokenization: We build a side network f_θ, which is an adapter that conditions frozen VGGT camera tokens on inertial embeddings via cross-attention and outputs per-camera poses (Fig. 1). Its inputs are:

- The VGGT camera token $\mathbf{h}_{(t,c)} \in \mathbb{R}^{d_{\text{vggt}}}$, one per camera per frame ($n \cdot C$ tokens total), with d_{vggt}=2048;
- The IMU sequence $\mathbf{u}_t \in \mathbb{R}^{K \times 6}$, where K is a hyperparameter that defines the number of IMU samples preceding an image. We set K=40 which provides 100 ms of accelerometer/gyroscope readings preceding τ_t. The IMU is rig-mounted, so a single $\mathbf{u}_t$ matrix is shared by all C cameras of a frame.

We tokenize $\mathbf{u}_t$ in two complementary ways, both projecting into the IMU embedding space d_{imu}=512 (chosen after hyperparameter search):

- *Segment tokens.* Rather than train on the raw stream, we split the ordered K samples into M=8 contiguous chunks of K/M samples, flatten each $\mathbb{R}^{(K/M)\times 6}$ chunk, and map it with a shared two-layer MLP to a token $\mathbf{z}_m \in \mathbb{R}^{d_{\text{imu}}}$, $m \in \{1, \dots, M\}$.
- *Pre-integration token.* Inspired by VINS-Mono, we summarize the net motion over the window into a single vector $\mathbf{z}^{\text{pre}} \in \mathbb{R}^{d_{\text{imu}}}$. We experiment with a two-layers 1D CNN (kernel 3, dilations $1, 2$) over the $(K, 6)$ stream followed by global average pooling and a linear projection, and with $2\times$ Transformer blocks where we use a [CLS] token, as done in BERT [2] and ViT [3] to aggregate the sequence's tokens into a single feature.

Inertial memory encoder: We stack the two outputs above into a matrix we call the inertial memory $\mathbf{Z}_t = [\mathbf{z}_1; \dots; \mathbf{z}_M; \mathbf{z}^{\text{pre}}_{M+1}] \in \mathbb{R}^{(M+1)\times d_{\text{imu}}}$ and encode it with a self-attending Transformer E (2 blocks, 8 heads, FFN width $4 \cdot d_{\text{imu}}$). Attention is restricted to a single session's IMU set, so $\mathbf{Z}_t$ never mixes inertial content across sessions.

IMU-conditioned camera token decoder: We project each image's inferred VGGT-camera token to the side dimension d_{imu} using a single-layer MLP and use it as the query of a cross-attention decoder D (2 pre-norm blocks) whose keys and values are the encoded inertial tokens $E(\mathbf{Z}_t)$. An individual session's inertial context u conditions all C camera queries, and each camera's query determines how that shared motion evidence corrects its own pose.

Pose head: A four-layer MLP head $\mathbb{R}^{d_{\text{imu}}} \to \mathbb{R}^7$ maps the conditioned token to a raw 7-vector; its first four entries are ℓ_2-normalized to a unit quaternion $\mathbf{q}_{(t,c)}$ and the last three are the translation $\mathbf{p}_{(t,c)}$, which together define $T_{(t,c)}$. We considered inputting the inertial features at every one of VGGT's 24 stages (akin to a 'ladder' decoder) rather than only at the final tokens, but kept the single-tap design, which trains an order of magnitude faster and leaves VGGT's forward pass untouched and cacheable.

Uncertainty-aware multi-task loss: Translation and ro-

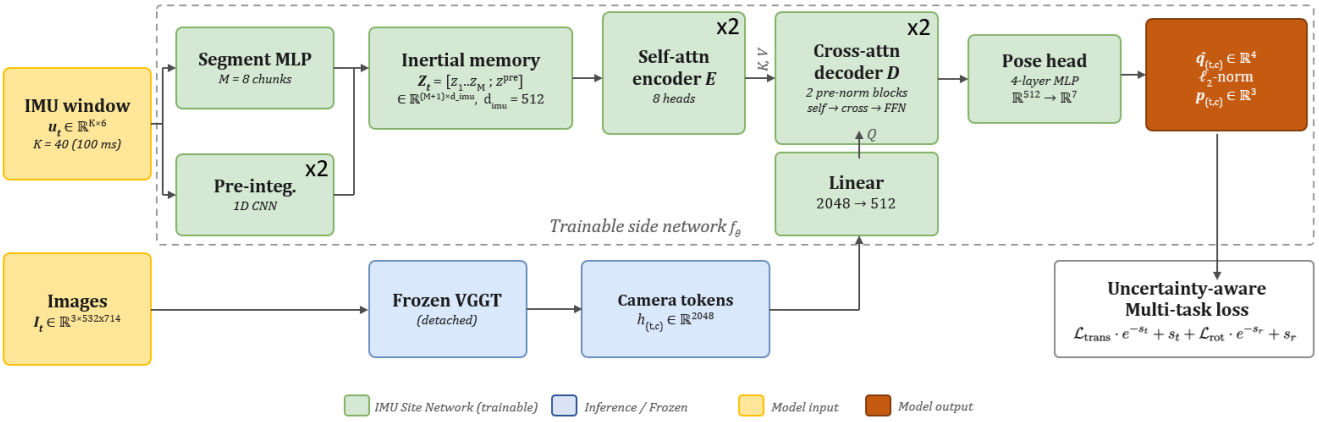

Figure 1: Pose estimation network (Side Network) implementation

tation live in different units; we therefore supervise them separately and balance them by learned homoscedastic uncertainty [10] rather than a hand-tuned weight. For each non-anchor pose, we use the quaternion geodesic angle and the Euclidean translation error:

$$\mathcal{L}_{\text{rot}} = 2\arccos\big(|\langle q_{(t,c)}, q^*_{(t,c)}\rangle|\big) \tag{2}$$

$$\mathcal{L}_{\text{trans}} = \|p_{(t,c)} - p^*_{(t,c)}\|_2 \tag{3}$$

where $q^*_{(t,c)}$ and $p^*_{(t,c)}$ are the ground-truth pose $T^*_{(t,c)}$ and the anchor $(1,0)$ is excluded. The loss function forces to learn per-task log-variances s_r, s_t (initialized at 0):

$$\mathcal{L}_{total} = e^{-s_r}\,\mathcal{L}_{\text{rot}} + s_r + e^{-s_t}\,\mathcal{L}_{\text{trans}} + s_t \tag{4}$$

The exponent's negative signs force a high-uncertainty task to be down-weighted. The addition terms $+s_{\{t,r\}}$ act as regularization terms: they are added to prevent the network from selecting very large values of s_r, s_t in a bid to drive the loss to values near zero.

Rig Constraints: We do not provide the model with any prior about the rig's constraints, except for forcing the model to anchor its results so that the predicted transformation of the first shot taken by the first camera is I_4, according to how VGGT works, which explains why our task is to predict $(n \cdot C) - 1$ poses, not $(n \cdot C)$. We do this by subtracting $\hat{p}_{(1,0)}$ from all predicted translation vectors $\hat{p}$. Therefore, VGGT is unaware of the relationship between the cameras, and treats the $n \cdot C$ images as if they were taken using the same camera at different positions.

4.2. Depth Maps enhancements

To enhance the depth maps generated by VGGT, we implement BetterDepth by following their strategy (Fig. 2). We base our work on existing Marigold checkpoints which we retrieve from Huggingface (`prs-eth/marigold-depth-v1-1`) and train the network on the Hilti data. We select the learning rate and schedule from the sweep reported in Sec. 5.

BetterDepth refines a coarse depth prediction (the VGGT-produced depth map) with a latent-diffusion model: a frozen Stable Diffusion VAE encoder encodes the image, the coarse depth map, and the ground-truth depth into latents z_r, z_c, z_0 respectively, and a U-Net is trained to denoise the target latent while conditioned on z_r and z_c. We freeze the VAE and use CLIP [17] to encode an empty prompt. We condition on the frozen VGGT depth as the coarse input, so the refiner stays an adapter outside the backbone, similar in spirit to the IMU side network. Because VGGT depth is relative, we rescale it to metric units by the median $D_{\text{gt}}/D_{\text{vggt}}$ over the training split.

The U-Net consumes 8 latent channels $[\,z_r \mid z_t\,]$; we widen its first convolution to 12 channels $[\,z_r \mid z_c \mid z_t\,]$ to admit the coarse depth. Since the convolution is linear in its channels, we keep the pretrained behavior at initialization by copying the image and noise weights into the slots that receive image and noise data and zero-initializing the new $4:8$ channels, so the coarse latent contributes nothing until training adapts it. This is the zero-initialization used by ControlNet [34]. We ablate this choice against Xavier [6]. We adopt zero-initialization, as it allows refinement to begin from VGGT's depth rather than a corrupted one, and it is the setting we use. We train only the U-Net, with a v-prediction loss [23], and globally pre-align the coarse depth to the ground truth by a per-frame scale and shift before normalization. We treat each image as an independent monocular problem (discarding the synchronized camera frames and only consuming each image individually) and hold the poses fixed.

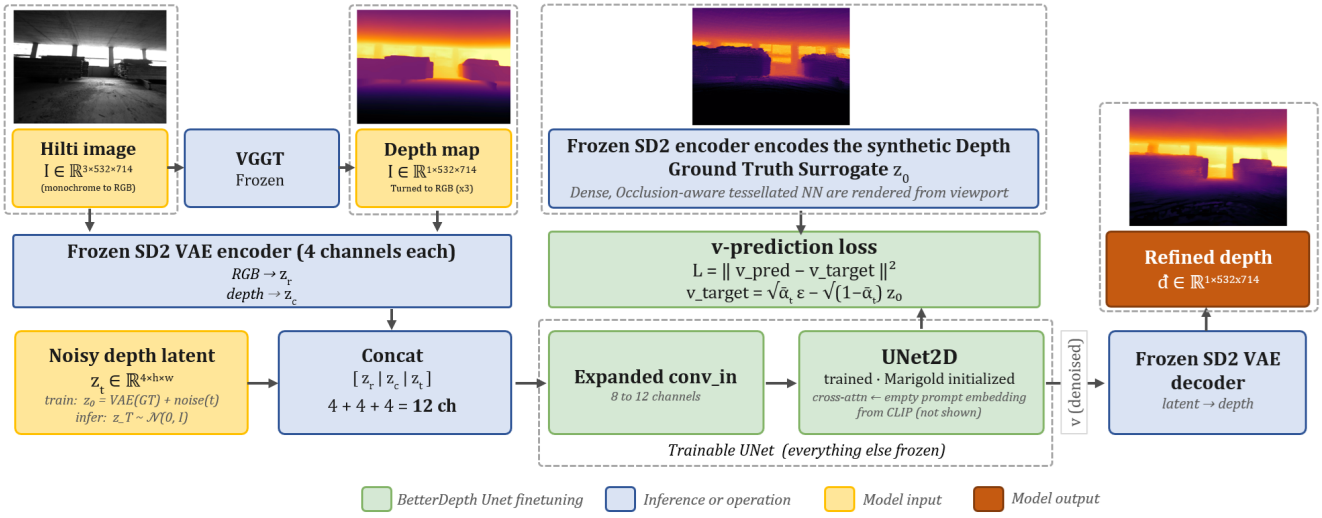

Figure 2: BetterDepth Implementation highlighting the network architecture and qualitative result for a single sample. More results can be found in Fig. 6.

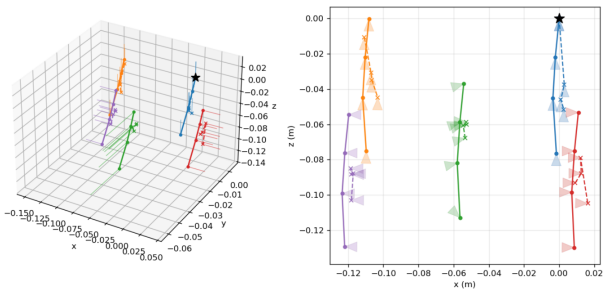

Figure 3: **Rig trajectory of a random sample in the validation set**. 3D and side (XZ) view. *Legend* solid = GT. dashed = predicted using side-Network (VGGT+IMU). $\star$ = anchor $(\text{cam}_0, \text{frame}_0)$.

5. Experiments

5.1. Inertial-based pose adjustment

We organize our pose experiments around two questions that follow from the decoupling mentioned in the problem section. First, does conditioning frozen VGGT camera tokens on inertial evidence repair the poses VGGT predicts on its own? Second, how much of the pose can inertial data recover with no visual backbone at all?

To answer these questions, we run two sets of experiments. First, we freeze VGGT, train the side network on our training set for nine epochs with AdamW with a cosine scheduler that warms up for 500 steps and a learning rate of 10^{-4}. We then evaluate on the held-out set of site1 composed of 171 time-ordered windows ($n = 4,\ C = 5$), IMU buckets $K = 40$, spanning 100 ms.

We observe a bimodal failure in VGGT's rotation: cam_0 and cam_1 have initial estimates close to ground truth, but the 3 remaining cameras do not. VGGT's training recipe is order invariant: temporal position of frames is never taken into consideration during training, nor is the model aware of multi-camera setups. A single VGGT inference job should only contain images of the same dimensions, taken by a camera with consistent intrinsics.

On the forward-facing camera cam_0 the initial rotation error is $0.76°$ and on its stereo partner cam_1 it is $6.16°$, while on the three peripheral cameras (cam_2 up, cam_3 right, cam_4 left) it exceeds $100°$. The two forward cameras share a field of view and give VGGT the correspondences required for epipolar geometry, whereas the peripheral cameras observe disjoint surfaces and leave their orientation effectively unconstrained, especially due to our preprocessing method which involved un-distorting the fisheye images. We validated the absence of meaningful trackable features by querying the tracker head for pixel pairs across pixels in different cameras, and found no visual correlation when any of the cameras $cam_{2,3,4}$'s pixels are queried against $cam_{0,1}$. Conditioning the camera tokens on inertial evidence lowers the mean error on those three cameras into the 9–14° range and holds the forward pair near $1.2°$. The side network which conditions the camera tokens achieves a mean RRE of $7.63°$ across all cameras. Per-camera results are provided in Table 1.

We find that the improvement introduced by the IMU side network is attributed to several factors, such as the data imbalance (with only a few high acceleration / sharp turn windows) making it effective to regress to the mean.

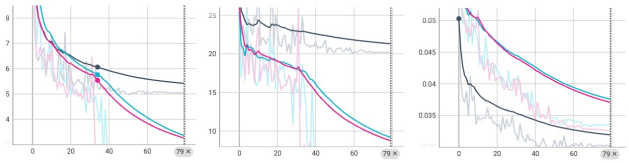

Figure 4: **Validation results of the side network**. **Left:** Rotation error across all cameras in degrees. **Center:** Rotation error for cam_3 in degrees (different scale). **Right:** Translation error across all cameras in meters. X axis = validation step

	cam0	cam1	cam2	cam3	cam4
VGGT	**0.76**	6.16	119.71	102.84	102.01
Ours	1.17	**1.22**	**9.47**	**12.50**	**13.79**
VGGT	**0.022**	0.104	0.835	0.186	0.395
Ours	0.036	**0.028**	**0.032**	**0.030**	**0.029**

Table 1: Side Network per-camera mean RRE (upper section) and RTE (lower section) on the validation set. VGGT exceeds $100°$ on the three peripheral cameras (cam2–cam4).

This symptom was clear in our analysis where we found the per-camera standard deviation on $cam_{2,3,4}$ remains 33-42°, where the large motions are the ones our network misses. Additionally, translation flattens to a near-uniform ≈ 3 cm across all five cameras, with a validation translation error settling at 3.1-3.6 cm within few epochs without subsequent movement. That value coincides with the precision of the interpolated ground truth, therefore the network has converged to the window-mean displacement to within what the supervision can resolve (see Fig. 3). Plotting $||t_{gt}||$ against $||t_{pred}||$ reveals this symptom, and highlights the network's inability to identify the rare occurrences that deviate from the mean. Therefore, we believe the network is simply overfitting, and the 3 cm residual error is a description of the supervision's resolution rather than a demonstrated property of the model (see Fig. 7).

Consistent with a single task loss, the homoscedastic weights stay symmetric throughout ($e^{-s_r} \approx e^{-s_t} \approx 1.46$), so the objective never re-prioritizes one task over the other.

We finally remove the visual backbone altogether and ask how far inertial data alone carries the pose. We train our side network by disconnecting it from VGGT, where we simply remove the decoder, and double the amount of encoder blocks, resulting in a network that purely processes IMU samples and produces poses, without relying on visual cues of any sort. The results are described in Table 2. This IMU-only network reaches a validation rotation error of $1.70°$, below VGGT's peripheral-camera errors and

	Side network (VGGT + cross-attn)	Backbone-free (IMU-only)
Uses VGGT	yes	no
Val RRE (°)	7.63	**1.70**
Val RTE (m)	**0.032**	0.132

Table 2: Inertial encoders comparison. The backbone-free predictor attains the lower rotation error but the highest translation error.

the $7.6°$ of the token-conditioning network, making it better performing for the rotation task for the side cameras. Translation does not follow: the validation RTE stays near 0.13 m, and drifts upward over training $0.126 \rightarrow 0.132\,m$.

5.2. Reconstruction error attribution

We ran a no-training attribution experiment to ask where the residual Chamfer-L_2 error actually lives. Let $\mathcal{C}(\mathcal{P}, \mathcal{Q})$ denote the symmetric voxel Chamfer distance (voxel size $v = 0.05$ m) between a predicted cloud $\mathcal{P}$ and the interpolated ground truth $\mathcal{Q}$:

$$\mathcal{C}(\mathcal{P}, \mathcal{Q}) = \frac{1}{|\mathcal{P}|} \sum_{\mathbf{p} \in \mathcal{P}} \min_{\mathbf{q} \in \mathcal{Q}} \|\mathbf{p} - \mathbf{q}\|^2 + \frac{1}{|\mathcal{Q}|} \sum_{\mathbf{q} \in \mathcal{Q}} \min_{\mathbf{p} \in \mathcal{P}} \|\mathbf{q} - \mathbf{p}\|^2 \quad (5)$$

To isolate the pose contribution we hold VGGT's depth fixed and substitute its predicted poses with our interpolated ground-truth poses, defining:

$$\chi_{\text{vggt}} = \mathcal{C}\big(\mathcal{P}(D^{\text{vggt}}, \hat{g}^{\text{vggt}}), \mathcal{Q}\big) \quad (6)$$

$$\chi_{\text{floor}} = \mathcal{C}\big(\mathcal{P}(D^{\text{vggt}}, g^{\text{gt}}), \mathcal{Q}\big) \quad (7)$$

so that χ_{floor} is the best Chamfer-L_2 distance attainable from VGGT depth with perfect poses (projected from ground truth poses). This acts as a depth-limited floor, and enables us to calculate $\chi_{\text{vggt}} - \chi_{\text{floor}}$, which is the entire pose-attributable error. We evaluate over an 80-window subsample of the time-ordered validation split.

The results highlight the weight depth maps have on point map reconstruction. Substituting perfect poses lowers mean Chamfer-L_2 only from $\chi_{\text{vggt}} = 52.54$ to $\chi_{\text{floor}} = 50.75$, a gap of 3.4%. The remaining 96.6% are attributed to depth. This explains the muted reconstruction response when attempting to solve pose alone.

We note that full scene point-cloud Chamfer distance can be difficult to interpret for monocular depth given that one camera can only observe a small part of the LiDAR sweep. Therefore, we performed a full-scene depth oracle experiment as a diagnosis rather than a final depth solution.

In this diagnosis, camera poses were fixed to LiDAR-derived GT trajectory and only depth map used for backprojection was varied. For the scale sweep, each VGGT depth map D is replaced with $D' = \lambda D$ before backprojection.

For each variant, we reconstructed point clouds from predicted depth maps and compared them against LiDAR references using Chamfer-L2 and F-score@0.1. Per window visualizations showed that Chamfer values can hide different failure modes by having over-extended predicted clouds that may reduce nearest-neighbor distance to LiDAR points without matching to the room geometry.

5.3. Diffusion-refined depth maps

Trained refinement on synthetic GT: We train the BetterDepth U-Net on our synthetic dense depth with the frozen VGGT depth as the source coarse condition. We add 4 channels to the U-Net to accommodate the VGGT-produced depth map z_c, and experiment with different initializations. We found Xavier adds noise into the conditioning path from the first step, and the refiner degrades VGGT's depth; zero-initializing them starts refinement from the pretrained prior, after which training improves the depth instead of destroying it. We therefore settle to zero these channels at the start, and sweep the learning rate $\{5{\times}10^{-7},\ 10^{-6},\ 2{\times}10^{-6}\}$ and the scheduler (constant vs. cosine). We observe that every run lowers the loss: the per-epoch v-prediction objective falls from ≈ 0.32 to ≈ 0.24 over 14 epochs, and our held-out BetterDepth chamfer-L_2 metric trends downward from the untrained value of 0.977 to a best of 0.75 at lr $= 2{\times}10^{-6}$ (-23%). Using a cosine scheduler and smaller learning rate both underperformed (Fig. 5).

These reductions indicate the efficacy of BetterDepth, but in our case are measured against an imperfect target: our supervision is rasterized from registered LiDAR and still carries reconstruction artifacts: occlusion-culling residue and near-blob "dark spots" which our pipeline suppresses but does not eliminate (see Fig. 6 for examples). A refiner trained to match this target necessarily learns part of its artifacts, which is why we report the loss reductions as evidence that diffusion-based refinement is a promising direction for VGGT depth on Hilti, not as a demonstration of metrically improved depth. We believe this question can only be settled with a dataset that supplies true RGB-D ground truth, or a better synthesis process. We explore recovering the value of the residual scaler λ by building a small MLP which we define in Sec. 5.4.

5.4. Recovering scale through residual refinement

To further increase depth accuracy, we explored whether we could learn a depth scale factor from the Hilti images. For each cam_0 frame, we align VGGT depth to the camera-visible LiDAR depth and train a small residual MLP, with sparse LiDAR supervision during training, to predict the corrections of VGGT depth. This is done using only image and VGGT-derived features like pixel location, aligned VGGT depth, image intensity, and local image-depth gradients. We experimented with training

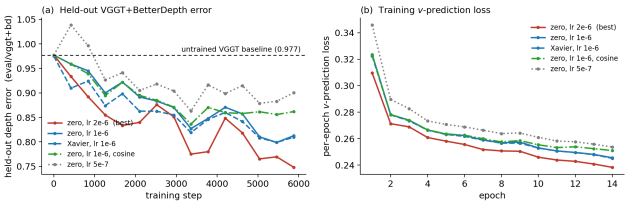

Figure 5: **BetterDepth training sweep. (a)** Held-out depth-evaluation error vs. training step; all runs fall below the untrained baseline (horizontal dashed, 0.977). **(b)** Per-epoch v-prediction loss; although every configuration converges, we aborted training due to the issues our synthesized ground truth surrogate introduced (see Fig. 6).

a small MLP with a single hidden layer on cam_0 frames from `site1_handheld_1` and evaluated on 20 held-out frames. We swept small MLP variants over hidden dimension and learning rate, comparing against simpler ridge residual baselines. All residual MLPs improved over the aligned baseline VGGT. The best result was with the 256-dimensional hidden layer and the learning rate 10^{-3}. This residual refiner reduced the MAE from 2.48m to 2.28m, AbsRel from 0.286 to 0.246, and RMSE from 5.95m to 5.73m. MAE was improved in $16/20$ held-out frames. The modest gains by the residual refiner suggest that Hilti-specific training can correct part of VGGT's depth error (see Fig. 8 for more details).

6. Conclusion

Conditioning frozen camera tokens on inertial evidence does remove VGGT's bimodal failure and reduces peripheral cameras errors dramatically, but the gain reflects the supervision and the network regresses to window-mean motion, missing the rare high-acceleration windows. The backbone-free variant attains a lower rotation error, which since both networks share the same objective isolates the injection point and suggests that late cross-attention into frozen camera tokens is the wrong place to inject inertial evidence.

The majority of the error comes from depth, which BetterDepth lowered across every ablation we ran; we stop short of calling this conclusive only because our synthetic supervision carries rasterization artifacts that BetterDepth partly learns. An inertial fusion adapter trained at earlier layers in tandem with VGGT's training objective would have permitted ablations to validate the token's expressivity (in terms of supporting other VGGT features), hence early fusion strategies would have been preferred. We expect that operating on the fisheye imagery, training on more inertial data, and better RGB-D ground truths would have prevented the failures we observed.

References

[1] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] A. Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[4] R. Duan, E. Hong, D. Zhao, E. Turner, A. Wong, and Y. Zhou. Fisheye3r: Adapting unified 3d feed-forward foundation models to fisheye lenses. *arXiv preprint arXiv:2603.28896*, 2026.

[5] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[6] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[7] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker: It is better to track together. In *Proc. ECCV*, 2024.

[8] B. Ke, A. Obukhov, S. Wang, P. Tang, K. Schindler, O. Sorkine-Hornung, and C. Schroers. Betterdepth: Plug-and-play diffusion refiner for zero-shot monocular depth estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[9] B. Ke, K. Qu, T. Wang, N. Metzger, S. Huang, B. Li, A. Obukhov, and K. Schindler. Marigold: Affordable adaptation of diffusion-based image generators for image analysis, 2025.

[10] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, 2018.

[11] V. Leroy, Y. Cabon, and J. Revaud. Grounding image matching in 3d with mast3r. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[12] A. D. Nair, J. Kindle, P. Levchev, and D. Scaramuzza. Hilti SLAM challenge 2023: Benchmarking single + multi-session SLAM across sensor constellations in construction, 2024.

[13] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[14] A. Paliwal. feynman, may 2026.

[15] L. Piccinelli, Y.-H. Yang, C. Sakaridis, M. Segu, S. Li, L. V. Gool, and F. Yu. Unidepth: Universal monocular metric depth estimation, 2024.

[16] T. Qin, P. Li, and S. Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. 2017.

[17] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[18] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[19] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022.

[20] T. Ren and S. Shen. Grounded sam 2: Ground and track anything in videos. `https://github.com/IDEA-Research/Grounded-SAM-2`, 2024.

[21] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

[22] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

[23] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. In *ICLR*, 2022.

[24] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] A. Torralba, P. Isola, and W. T. Freeman. *Foundations of Computer Vision*. Adaptive Computation and Machine Learning series. The MIT Press, 2024.

[26] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. Stachniss. Kiss-icp: In defense of point-to-point icp – simple, accurate, and robust registration if done the right way. 2022.

[27] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[28] L. Wang, L. Guo, Z. Xu, Q. Wang, F. Gao, and X. Chen. Lidar-vggt: Cross-modal coarse-to-fine fusion for globally consistent and metric-scale dense mapping, 2025.

[29] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[30] S. Wang, V. Leroy, Y. Cabon, T. Morfitis, V. Sekkat, J.-Y. Chen, M. Cord, P. Perez, and C. Schmid. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20716–20726, 2024.

[31] Y. Wang, J. Zhou, H. Zhu, W. Chang, Y. Zhou, Z. Li, J. Chen, J. Pang, C. Shen, and T. He. π^3: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025.

[32] J. Yang, A. Sax, K. J. Liang, M. Henaff, H. Tang, A. Cao, J. Chai, F. Meier, and M. Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2025.

[33] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao. Depth anything v2. *arXiv:2406.09414*, 2024.

[34] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. In *ICCV*, 2023.

[35] Y. Zhou, X. Li, S. Li, Z. Yan, C. Xia, and S. Feng. Mast3r-fusion: Integrating feed-forward visual model with imu, gnss for high-functionality slam, 2025.

7. Contributions & Acknowledgements

E.T. developed the ROS extraction tools, and developed, trained and evaluated the IMU-camera fusion network, the barebone IMU network, and the BetterDepth U-Net network. He also authored this report.

Z.G. defined the early project direction and suggested the dataset. She developed the dataloader, including the RANSAC+ICP pose pipeline, fisheye undistortion, point cloud registration mechanisms, and window design, along with visualization tools. She contributed to experiments and evaluations, authored the related work section, and built the poster.

T.G. developed the human-presence pipeline, LiDAR filtering pipeline, and conducted exploratory depth and reconstruction analyses. He developed, trained, and evaluated the LiDAR-supervised residual depth-refinement MLP (Sec. 5.4). He performed a multi-sweep LiDAR supervision ablation. He also contributed to experiments, evaluations and the related work section.

7.1. Generative AI

Throughout the project, the network design and implementation, the experiment design, research questions, interpretation of what the results meant, as well as authoring of the report remained the team's own original work.

We used a mix of Claude Opus 4.7, Opus 4.8, and ChatGPT 5.5 to increase our productivity across three areas:

- **Conversational**: to build deeper understanding and check reasoning. These conversations were used to develop intuition for unfamiliar topics related to multi-view geometry, fisheye imagery, point cloud registration, loss-specific questions (inquiring about negative loss in Kendall, and about convergence/divergence of task weights), as well as infrastructure-related matters such as multi-processing, caching, and LaTEX related questions.
- **Coding agent** Helped troubleshoot code we have written, such as the implementations of the ROS 1 bag extraction library, IMU bias estimation and correction, the RANSAC+ICP pose pipeline, Synthetic RGB-D ground truth tessellation and raycast rendering, the implementation and evaluation of Grounded-SAM2 image masking, conversion of our initial scripts to be hyperparameters-sweepable with WandB, logging and caching features to disk, opencv fisheye undistortion, coordinate-transformation support libraries, quaternion to $SE(3)$ conversion and vice-versa, and build some visualization-assisting tools in matplotlib and open3d.
- **Literature Review**: We used Feynman [14] solely to conduct literature review of topics such as "Feed-forward 3D reconstruction SOTA", "Feed-forward and VIO latest papers", "Review of distillation-based methods for depth map refinement".

7.2. Python libraries

Library	Version
NumPy	2.4.4
Pandas	2.3.3
PyTorch	2.11.0
VGGT	commit: a288dd0
SAM2	commit:2b90b9f
Open3d	0.19.0
OpenCV-python	4.13.0
KISS-ICP	1.3.0
SciPy	1.17.1
PyYAML	6.0.3
TQDM	4.67.3
Transformers	5.5.4
Matplotlib	3.10.8
Rosbags	0.11.3

Table 3: Libraries used in the project

8. Appendix

Figure 6: **Depth maps comparison**. The original VGGT depth maps (2nd col.) have more fidelity to the scene (1st col.) than our synthesized ground truth (4th column). The BetterDepth refined results (3rd col.) show the artifacts generated by our Ground Truth proxy.

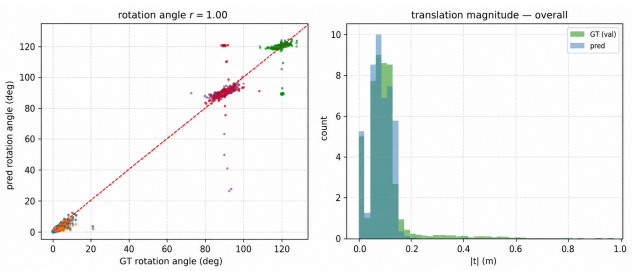

Figure 7: Rotation and translation error distribution. **Left** Rotation The majority of the relative rotation lie at $0°, 90°$, and $120°$ for $\text{cam}_{0,1}$, $\text{cam}_{3,4}$ and cam_2 respectively. The network clearly overfits due to the dominance of stationary points in the training set and fails to detect sudden variations. **Right** Distribution of translation vectors according to their magnitudes. The network successfully predicts movements up to 0.2 m but fails to capture long-tail, infrequent changes of up to 0.5 m.

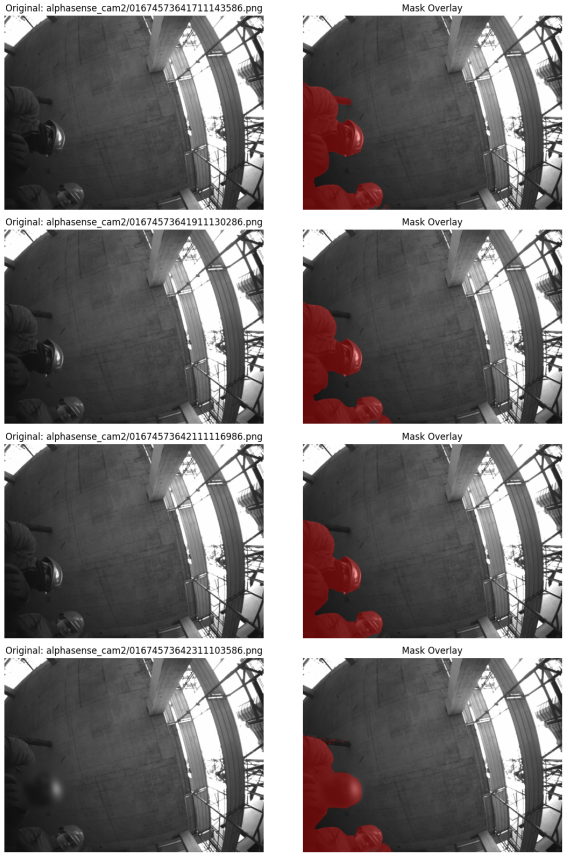

Figure 9: **Qualitative human-presence masking**, Representative original cam_2 frames alongside their Grounded-SAM2 mask overlays. Red regions are human-occupied pixels used to filter LiDAR points.

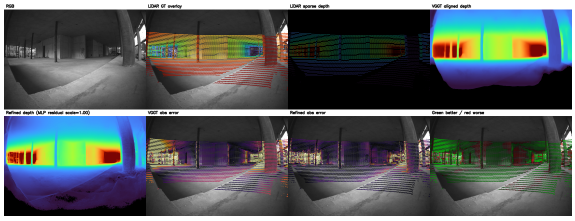

Figure 8: **Held-out Qualitative results from the Hilti-specific residual refiner**, grayscale input, camera-visible LiDAR supervision, sparse LiDAR depth, and aligned VGGT prediction. Bottom: refined depth, error maps, and per-pixel comparison where green shows improvement and red indicates degradation. Ground truth here is the sparse LiDAR reference, with potential calibration errors and occlusions. For this frame, MAE reduces from 1.6m to 1.00m, and across 20 held-out frames tested, MAE reduces from 2.48m to 2.28m.