

VGGT-SLAM 2.0: Real-time Dense Feed-forward Scene Reconstruction

Dominic Maggio¹ and Luca Carlone^{1*}

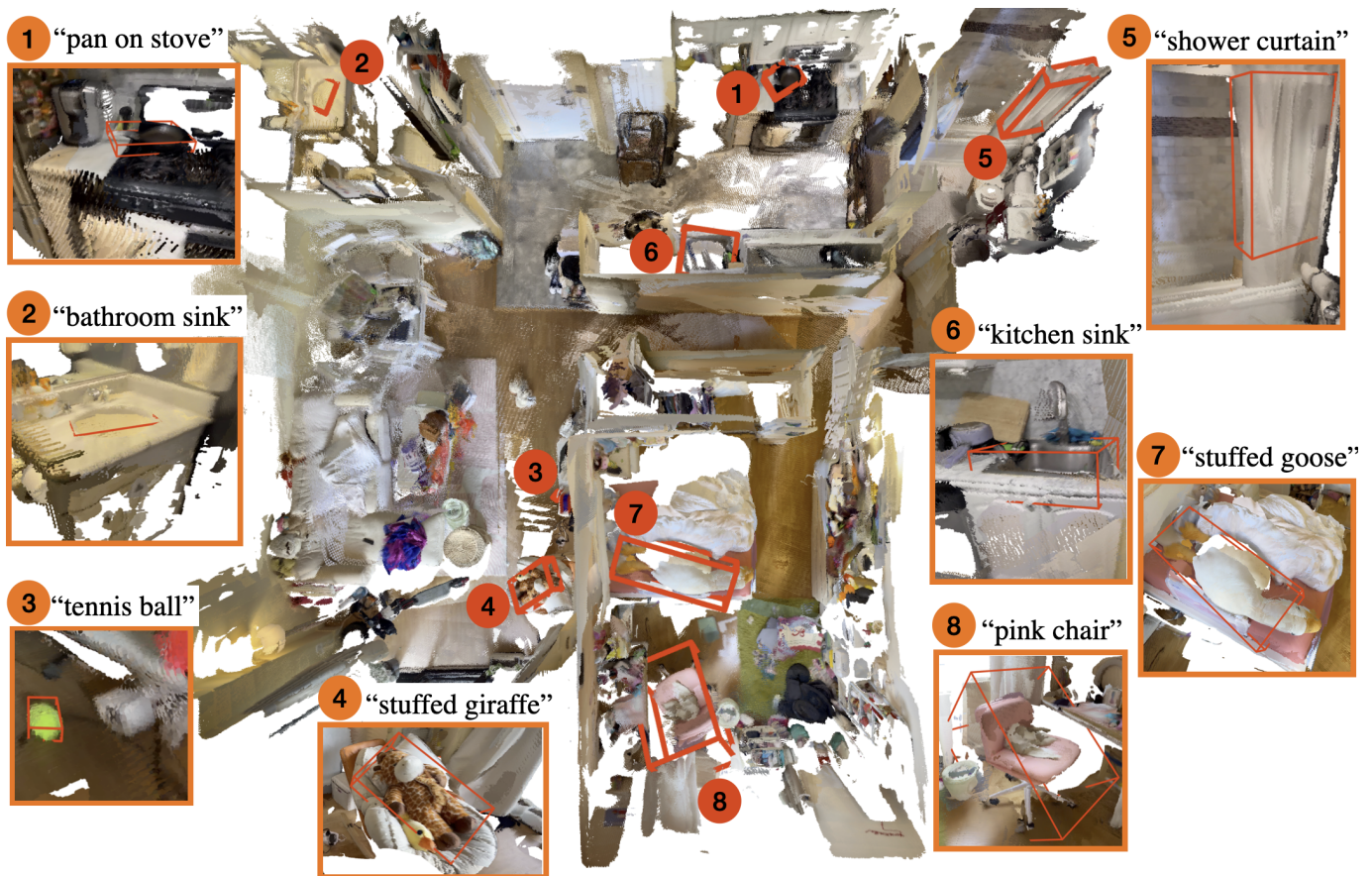


Fig. 1. A VGGT-SLAM 2.0 map of an apartment scene including a living room, kitchen, bedroom, 2 bathrooms, and closet built using uncalibrated RGB images collected with an iPhone. After the map is constructed, it is easy to use it for retrieval of open-set objects. Here, eight example open-set object queries are included, showing the resulting 3D bounding box produced by VGGT-SLAM 2.0 given the provided text query.

Abstract—We present VGGT-SLAM 2.0, a real-time RGB feed-forward SLAM system which substantially improves upon VGGT-SLAM for incrementally aligning submaps created from VGGT. Firstly, we remove high-dimensional 15-degree-of-freedom drift and planar degeneracy from VGGT-SLAM by creating a new factor graph design while still addressing the reconstruction ambiguity of VGGT given unknown camera intrinsics. Secondly, by studying the attention layers of VGGT, we show that one of the layers is well suited to assist in

image retrieval verification for free without additional training, which enables both rejecting false positive matches and allows for completing more loop closures. Finally, we conduct a suite of experiments which includes showing VGGT-SLAM 2.0 can easily be adapted for open-set object detection and demonstrating real-time performance while running online onboard a ground robot using a Jetson Thor. We test in environments ranging from cluttered indoor apartments and office scenes to a 4,200 square foot barn, and we also demonstrate VGGT-SLAM 2.0 achieves the highest accuracy on the TUM dataset with about 23 percent less pose error than VGGT-SLAM. Code is available at <https://github.com/MIT-SPARK/VGGT-SLAM>.

¹Laboratory for Information & Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA. {drmaggio, lcarlone}@mit.edu

*Luca holds concurrent appointments as a faculty at the Massachusetts Institute of Technology and as an Amazon Scholar. This paper describes work performed at MIT and is not associated with Amazon. This work was supported in part by the NSF Graduate Research Fellowship Program under Grant 2141064, the ARL DCIST program, and the ONR RAPID program.

I. INTRODUCTION

Recently, the foundational task in robotics and computer vision of using images from a camera to simultaneously create a 3D reconstruction of a scene and localize the camera has seen a paradigm shift from using classical multi-view geometry and

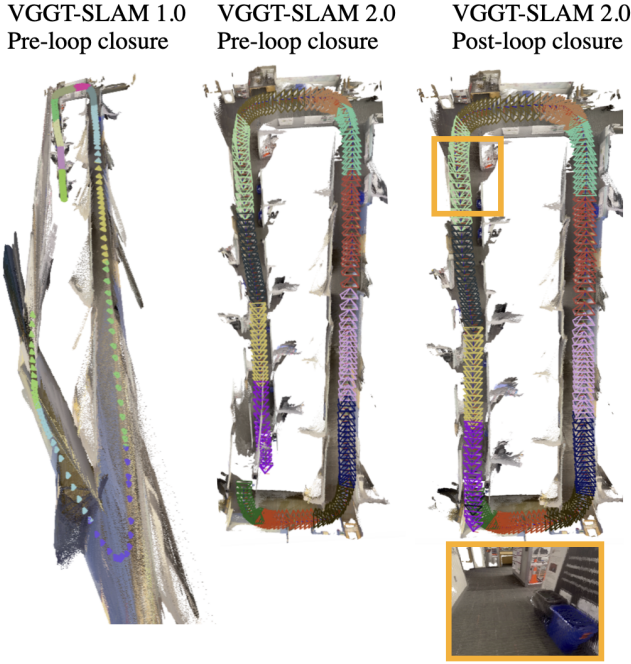


Fig. 2. Office loop dataset from [5]. Left: VGGT-SLAM [5] before detected loop closure showing high-dimensional drift. Middle: VGGT-SLAM 2.0 before detected loop closure showing substantially reduced drift. Right: VGGT-SLAM 2.0 after detected loop closure

optimization techniques [1] towards building on top of feed-forward geometric foundation models [2, 3, 4] to develop SLAM systems [5, 6, 7, 8, 9]. These new hybrid SLAM systems, which combine both geometric foundation models with tools from classical SLAM, produce a much simpler SLAM system that is both easier to use and maintain. They also produce dense RGB point clouds maps while not requiring camera calibration be known beforehand.

One recent work, VGGT-SLAM [5], shows that even though VGGT [4] is unable to directly process potentially thousands of frames as typically done in SLAM¹, it can be extended to become a SLAM system by creating and aligning submaps from VGGT where neighboring submaps share an *overlapping frame*. The overlapping frame is a common image between submaps which is used for alignment. Furthermore, VGGT-SLAM shows that simply aligning the submaps with a similarity transformation (rotation, translation, and scale) is not always sufficient since the VGGT submaps — which rely on uncalibrated camera images — sometimes have a higher dimensional projective ambiguity. VGGT-SLAM thus optimizes a 15-degree-of-freedom (DoF) alignment on the $SL(4)$ manifold which compensates for submap distortions.

However, there are several key limitations which hinder the deployment of VGGT-SLAM which are addressed in this paper. The first is the higher dimensional 15-DoF alignment introduces rapid drift between loop closures which can severely warp a scene before loop closures (Fig. 2) and is not always recoverable even with loop closures. Solving for the 15-

DoF homography is also degenerate in planar scenes (since a minimum solver requires 5 non-coplanar points) which can cause divergence in common scenarios such as a camera looking only at a wall or at a flat floor.

Secondly, the factor graph in VGGT-SLAM only attempts to estimate the homography between submaps instead of correcting keyframe-level errors. This means that any potential translation and rotation error from VGGT is handled sub-optimally during global optimization from loop closures.

Thirdly, image retrieval for loop closure in a VGGT mapping system has largely been treated as an independent task using only information from a separate retrieval network. VGGT-SLAM and VGGT-Long [8] for example both use SALAD [10] to retrieve frames while not leveraging information inside the VGGT layers to verify the validity of the retrieved frame.

Contributions. To address these challenges, we develop VGGT-SLAM 2.0 which provides the following contributions:

- Our first contribution (Section IV-A) is to remove the issue of high-dimensional drift and planar degeneracy of VGGT submap alignment by directly enforcing that the two overlapping frames in submap alignment must have the same position, rotation, and calibration, and solve for a consistent scale factor.
- Our second contribution (Sections IV-A and IV-C) is a new factor graph structure where all keyframes are nodes and the graph contains *intra* edges connecting keyframes inside a submap and *inter* edges connecting overlapping frames between submaps.
- Our third contribution (Section IV-B) is to study the attention layers of VGGT to demonstrate that, without any fine-tuning, one layer of VGGT is particularly well suited to verify the validity of a potential retrieved frame for loop closures. This helps prevent false positives (including in challenging environments such as an office floor with similar instances of desk cubicles), and in turn enables establishing more loop closures.
- Finally, to show the applicability of VGGT-SLAM we conduct a suite of experiments showing: (1) simple yet effective integration of open-set object detection into the geometric VGGT-SLAM map, (2) real-time mapping on a Jetson Thor running online on board a ground robot, (3) reconstruction of environments ranging from cluttered indoor scenes to a 4,200 square foot barn to outdoor driving sequences from the Kitti dataset, and (4) the best accuracy of recent learning-based methods on the TUM RGB-D dataset with a 23% error decrease from VGGT-SLAM.

II. RELATED WORKS

Classical mapping techniques. Classical scene reconstruction methods rely on tracking and associating features [11, 12] across multiple frames and then performing backend optimization to regress camera poses and scene geometry [13, 14, 15]. This has been used in multiple sparse [16, 17] and dense [18, 19] real-time SLAM systems. Learning-based

¹VGGT is limited to about 60 frames on an RTX 4090 with 24 GB due to memory constraints.

methods have also been incorporated in SLAM systems such as for optical flow [20] and for neural scene representations [21]. Several works also perform classical optimization of projective alignment on the $SL(3)$ group [22], and [23] discusses synchronization on the $SL(4)$ group.

Feed-forward scene reconstruction. Recently, following the seminal works of DUST3R [2] and MAST3R [3], which are trained to take in a pair of uncalibrated monocular images and produce a 3D scene reconstruction, there has been a paradigm shift towards using Geometric foundation models (GFM) for SLAM. Since the two aforementioned works can only process a pair of images, multiple works have begun developing approaches to process larger numbers of frames. MAST3R-SFM [24] builds on MAST3R to perform global optimization over multiple images but quickly grows computationally expensive as the number of frames increases. MAST3R-SLAM [6] is the first work to demonstrate real-time SLAM with a GFM. MAST3R-Fusion [9] incorporates IMU and GNSS as additional sensor measurements through classical optimization.

Several works have developed GFMs to directly process multiple images [25, 26, 27, 4] with some also modifying the networks to also enable Gaussian Splatting [28] reconstructions [29]. However, all of these works are restricted by GPU memory usage to a relatively small number of frames. To extend VGGT [4] to a large number of frames, VGGT-SLAM [5] is the first work to incrementally create and align submaps with VGGT. VGGT-Long [8] aligns VGGT submaps for large-scale outdoor self-driving scenes. Recently, SING3R-SLAM [30] creates submaps and fuses them into a global Gaussian Splatting map and ViSTA-SLAM [7] uses a lightweight model for two-view association combined with a $Sim(3)$ pose graph. Additionally, MegaSaM [31] uses deep visual SLAM to handle reconstruction for dynamic scenes and TTT3R [32] uses Test-Time Training to extend CUT3R [25] to longer sequences. Concurrently, TALO [33] uses a sparse set of control points and Thin Plate Spline deformations fields to align submaps created by a GFM.

Attention layer analysis. Perception Encoder [34] conducts a study of the layers of a vision encoder for downstream tasks. The layers of DUST3R have recently been studied in [35], and [36] shows how an understanding of DUST3R’s layers can be used to enable dynamic object detection. Concurrently, recent works have begun exploring the layers of VGGT for tasks such as noise suppression [37] and geometric interpretation of the layers [38], but so far no works have studied the layers of VGGT for image retrieval verification.

III. NOTATION AND PRELIMINARIES

Notation. We represent matrices and vectors with bold letters (*e.g.*, $\mathbf{C}$) and sets using capital calligraphic fonts (*e.g.*, $\mathcal{C}$). To create a submap, $\mathcal{S}_i$, we pass a set of frames $\mathcal{I}$ to VGGT, and use the output camera calibrations, $\mathcal{K}$; poses, $\mathcal{T}$; depth maps, $\mathcal{D}$; and corresponding depth confidence maps, $\mathcal{C}$. All submaps have n keyframes, except for the loop closure submaps (Section IV-C) which have 2 frames.

Preliminaries. VGGT estimates 3D points which are defined w.r.t. the first camera of a submap. 3D points can be computed from either the point maps or by transforming the estimated depth by estimates of the calibrations and poses. In this paper, we will instead define 3D points w.r.t. their respective camera frame. Thus, the points $\mathbf{X}$ for frame i (*i.e.*, $\mathbf{X}_i$) are computed through back projection using only $\mathbf{K}_i^{-1}$ and $\mathbf{D}_i$, where $\mathbf{X}_i \in \mathbb{R}^{3 \times h \times w}$ for image of size $h \times w$.

The 4×4 homography matrix which expresses the relationship between corresponding points $\mathbf{X}_i^a, \mathbf{X}_j^b \in \mathbb{R}^3$ is:

$$\mathbf{X}_i^a = \mathbf{H}_j^i \mathbf{X}_j^b, \quad (1)$$

where we use overloaded notation such that $\mathbf{X}$ is in homogeneous coordinates when multiplied by a homography. The full 4×4 homography matrix,

$$\mathbf{H}_i = \begin{bmatrix} \mathbf{K}\mathbf{R} & \mathbf{t} \\ \mathbf{v}^T & s \end{bmatrix}, \quad (2)$$

has 15 degrees of freedom: 3 for translation ($\mathbf{t}$), 3 for rotation ($\mathbf{R}$), 1 for scale (s), 5 for affine (*i.e.*, camera calibration, $\mathbf{K}$), and 3 for projective components ($\mathbf{v}$) [39]. Since the homography matrix is only up to scale, optimizing the residuals between homography matrices can be done by mapping each homography to the $SL(4)$ manifold which consist of all 4×4 matrices with unit determinant, allowing for factor graph optimization on the $SL(4)$ manifold [5]. More common manifolds used in robotics such as $SE(3)$ and $Sim(3)$ are subsets of $SL(4)$.

IV. VGGT-SLAM 2.0

Our objective is to incrementally align submaps produced by VGGT into a globally consistent map that recovers the true geometry of the scene (up to a similarity transform ambiguity as we do not estimate the true scene scale). To do this, in Section IV-A we create a factor graph where every node corresponds to a keyframe in a submap and edges describe the relationship between keyframes. Consecutive submaps share an image such that the first image of a new submap is the same as the last image of the prior submap. We refer to the two submaps’ respective estimate of the frame of this image as the *overlapping frames*. In Section IV-B we show how the attention layers of VGGT can be used for image retrieval verification, that is, given a queried frame and a retrieved frame which are predicted to have overlap by a third-party image retrieval method (in our case SALAD [10]), we can use the attention layers to provide assurance that VGGT identifies overlap in the images. Finally, in Section IV-C we show how our new factor graph structure and image retrieval verification are combined to enable global optimization with loop closures.

A. Relative frame alignment

For each keyframe, we want to estimate a homography $\mathbf{H}_i$ which transforms the VGGT reconstruction (which may have submap-specific distortions) from the local camera frame to a globally consistent map. For edges inside a submap (*intra edges*), the edges will only have non-identity rotation

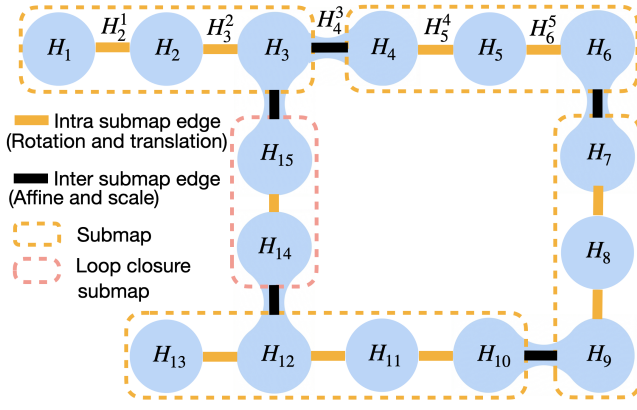


Fig. 3. Factor graph structure of VGGT-SLAM 2.0 showing edges inside and between submaps where all keyframes are nodes and loop closure submaps consist of two frames. Nodes connected with an inter submap edge such as H_3 and H_4 are an example of overlapping nodes (meaning the camera image associated with these nodes is identical). Here, for the loop closure submap which is made up of nodes H_{14} and H_{15} , H_{12} represents the node of a queried frame which was matched with the node H_3 of the retrieved frame.

and translation (*i.e.*, $SE(3)$) variables since projective distortion is consistent inside a submap, which means we do not have to apply affine correction to each keyframe separately. Optimizing these edges will help correct potential drift in VGGT’s translation and rotation estimates. For edges between overlapping frames (*inter edges*) there will only be non-identity calibration (*i.e.*, affine) and scale variables which will be used to make each submap’s estimate of the scale and calibration of the overlapping frame be identical. Since both of these transformations are subsets of $SL(4)$, we use $SL(4)$ nodes for factor graph optimization. A visualization of the factor graph structure showing different edge types can be seen in Fig. 3. Importantly, given that there is error in VGGT’s estimate of calibration, we do not know what the true camera calibration is for the overlapping frames — only that they must have the same calibration.

Intra submap frame alignment. For two keyframes i and j inside a submap, the homography $\mathbf{H}_j^i$ from Eq. (2) that transfers correspondences between point clouds $\mathbf{X}_i$ and $\mathbf{X}_j$ is simply a rotation and translation which we can get directly from the VGGT outputted poses as:

$$\mathbf{H}_j^i = \mathbf{T}_i^{-1} \mathbf{T}_j. \quad (3)$$

Inter submap frame alignment. Now, we have to find the alignment between the overlapping frames i and j in overlapping submaps. For example, nodes H_3 and H_4 in Fig. 3 correspond to the same keyframe processed in the first and second submap. Importantly, instead of solving for a 15-DoF transformation by optimizing an alignment over the 3D points corresponding to the two overlapping frames as was done in VGGT-SLAM, here we recognize that the overlapping frames by construction must have the same translation, rotation, and camera calibration. Thus, we enforce that the relative translation and rotation between the overlapping frames is zero and determine the transformation needed to align their calibration. We then solve for the scale factor, s .

The homography in Eq. (2) then simplifies to:

$$\mathbf{H}_j^i = \begin{bmatrix} \mathbf{K}_i^{-1} \mathbf{K}_j & 0 \\ \mathbf{0}^T & s \end{bmatrix}, \quad (4)$$

where the calibration components, $\mathbf{K}_i$ and $\mathbf{K}_j$, of the homography come from the VGGT estimated calibrations. Importantly, even if the calibrations $\mathbf{K}_i$ and $\mathbf{K}_j$ do not represent the true camera calibration (which can occur if VGGT incorrectly estimates camera calibration for the current submap), this enforces their calibrations to be identical. We know they must be identical because the overlapping frames represent the same camera. Solving for the scale factor is the only time we directly use the VGGT 3D points. To find s , we leverage two ideas. The first is that since the overlapping frames are identical, corresponding 3D points between them are trivially known. The second is that unlike [5], with our new problem formulation, we can easily estimate relative scale separately from other parameters of the homography. The 3D points of each image are defined not w.r.t. the first frame of a submap but rather to their own camera frame. Therefore, corresponding points seen in overlapping keyframes (*e.g.*, H_3 and H_4 from Fig. 3) must have the same scale. To estimate the scale, we warp the point clouds $\mathbf{X}_i$ and $\mathbf{X}_j$ to have the same calibration using $\mathbf{K}_i^{-1}$ and $\mathbf{K}_j$ and then estimate the scale factor as the median ratio between corresponding 3D point distances for all points with sufficient depth confidence $\mathcal{C}$.

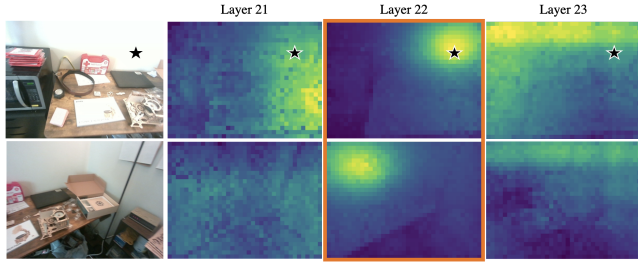
B. Image retrieval verification with VGGT

In this section, we demonstrate that by studying the attention layers of VGGT, we can easily extract information to determine if VGGT is able to find overlap between input images. While we find our analysis holds for an arbitrary number of frames passed to VGGT, since our objective in this section is image retrieval verification, we focus on the scenario where VGGT is given two images (a query and a retrieved frame). In particular, we reveal that layer 22 in VGGT consistently pinpoints whether VGGT predicts correspondences between the two images. In Fig. 4 we visualize the attention matrix of a selected key token (shown with a black star), with respect to all other query tokens. Layer 22 consistently shows a spotlight-like attention map which shows larger attention values at the correspond location of the image. This is not clearly present in the other layers, including the nearby layers (21 and 23) which are shown in Fig. 4 for comparison. This spotlight affects also works on low texture regions of a scene as demonstrated by the first example of Fig. 4 which places the query token on a plain white wall.

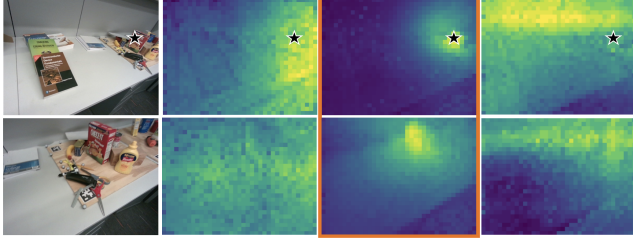
To leverage this visual observation, we construct a match score in Eq. (6) between two images which captures if layer 22 predicts that VGGT can reconstruct the two images.

$$\gamma_t = \max_{q \in Q^{(2)}} \left(\frac{\text{Softmax}(Q^{(2)} K^{(1)\top})}{\max_{q \in Q^{(1)}} \text{Softmax}(Q^{(1)} K^{(1)\top})} \right) \quad (5)$$

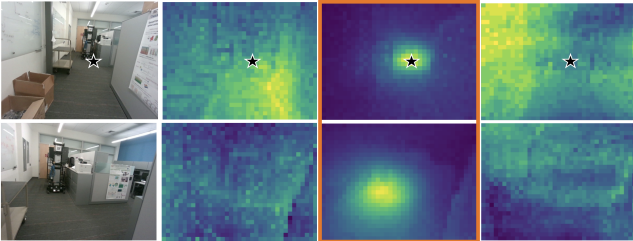
$$\alpha_{match} = \text{Mean}_{\text{top 25\%}}(\{\gamma_t\}) \quad (6)$$



(a) Example images from Clio [40] apartment scene. Key token placed on a plain wall.



(b) Example images from Clio [40] cubicle scene. Key token placed on a box.

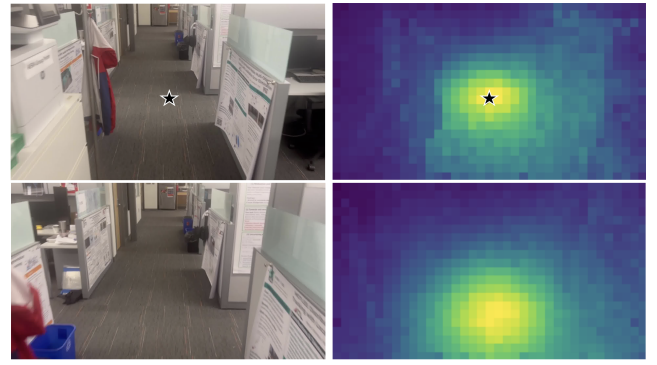


(c) Example images from Clio [40] office scene. Key token placed on the floor.

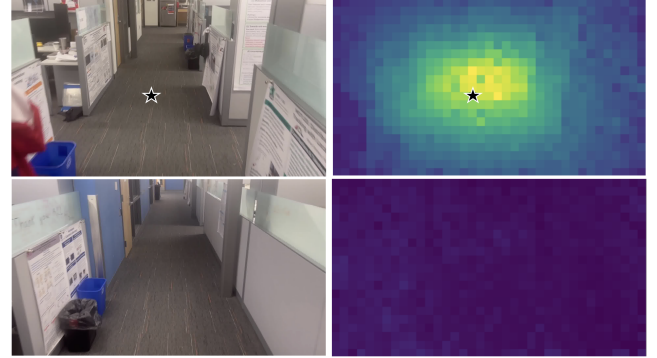
Fig. 4. Attention matrix of query tokens for pairs of images with respect to a selected key token (identified with a black star) using tokens from layers 21, 22, and 23. Layer 22 shows a clear spotlight of attention between corresponding parts of the image pairs.

Here, $Q^{(1)}$ and $K^{(1)}$ represents all query and key tokens (averaged across all heads) for image 1, and $Q^{(2)}$ is likewise all query tokens for image 2. Intuitively, Eq. (5) computes the attention scores of all query tokens of image 2 w.r.t. the key tokens of image 1 and normalizes by the corresponding attention scores of the query tokens of image 1 w.r.t. the key tokens of image 1, where we use the query tokens that yield the highest attention for each key token. Since even a correctly retrieved pair of images will usually only have partial overlap, Eq. (6) computes the final match score (α_{match}) by taking the mean of the top 25% of all ratios.

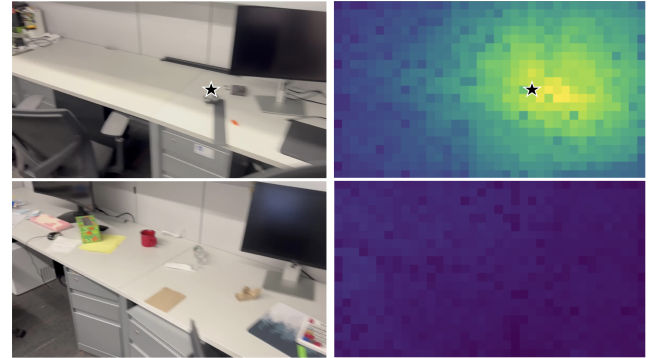
A visual demonstration of using layer 22 to verify image retrieval is shown in Fig. 5. Here we show the match score and attention map for a sample key token for three cases of query and retrieved frame: both frames have overlap, both frames do not have overlap, and both frames are from similarly looking office cubicles but do not have overlap. In this last case, SALAD produced a false positive match with high confidence while our match score accurately shows the frames do not overlap.



(a) Queried and retrieved frames that have overlap. Our estimated match score: 1.026.



(b) Queried and retrieved frames that do not have overlap. Our estimated match score: 0.491.



(c) Queried and retrieved frames that do not have overlap, but SALAD incorrectly estimated a match between them. Our estimated match score: 0.550.

Fig. 5. Examples using layer 22 to verify a match between queried and retrieved frames. We visualize attention maps between a selected key token and all query tokens revealing high attention for matching frames and low attention for non-matching frames. This is captured for all pairs of key and query tokens using our match score produced with Eq. (6).

C. Loop closures and global optimization

Similar to VGGT-SLAM, we perform image retrieval using SALAD [10] to check for possible loop closures between frames in the current submap and frames in prior submaps (ignoring the most recent prior submap since it will always have overlap with the current submap). We differ from VGGT-SLAM in two key ways. Firstly, instead of adding the retrieved frames into the current submap, we create a small submap of two frames (the loop closure submap shown in Figure 3).

This enables greater flexibility as VGGT can be run on the current batch of frames before performing image retrieval. This two frame submap consists of the retrieved frame from a prior submap and the corresponding query frame in the current submap. This two-frame loop closure submap is passed to VGGT and an Inter submap edge is then added between the frame of the loop closure submap and its respective overlapping frame, as shown in Figure 3.

Secondly, and most importantly, we use information from the attention layers of VGGT to estimate whether the retrieved candidate from SALAD is actually a correct match for which VGGT is able to produce a valid estimate as described in Section IV-B. This allows us to both relax the image retrieval threshold from SALAD to get more potential loop closure candidates and reject false positives in challenging environments. Given a SALAD loop closure candidate, we run the procedure in Section IV-B on the queried and retrieved frame and only add a loop closure edge if the procedure estimates a match between the two frames.

Given the loop closure constraints, we run global optimization which follows VGGT-SLAM and is optimized on the $SL(4)$ manifold using GTSAM [41] without robust losses. The $SL(4)$ optimization solver has since been added to the official GTSAM release, making installation simpler with just a pip install.

Recovering the global reconstruction. Now, given that we have the full set of homographies, we can use them to recover transformed poses and points for our map reconstruction.

We can recover projection matrices $\mathbf{P}_i$ which transform points from the global frame, w , to the camera frame corresponding to image I_i as

$$\mathbf{P}_i = \mathbf{K}_i^{3 \times 4} (\mathbf{H}_i^w)^{-1} \quad (7)$$

Note that differently from VGGT-SLAM, since we have defined points to be relative to their respective frames and not the first frame of a submap, the right-hand side of Eq. (7) only has a homography multiplied with a camera calibration (padded with a zeros vector to be 3×4) from VGGT instead of additionally being multiplied by a pose from VGGT. Global poses are found by decomposing $\mathbf{P}_i$ into calibration and pose, and global 3D points are recovered using Eq. (1) with $\mathbf{H}_i^w$.

V. EXPERIMENTS

A. Experimental setup

To show that VGGT-SLAM 2.0 does not require careful fine-tuning, we keep parameters constant across all quantitative experiments and also use the same parameter used by VGGT-SLAM for minimum disparity (50 pixels) between keyframes and for confidence threshold to filter VGGT points using $\mathcal{C}$ (25%). Unless otherwise mentioned, all experiments use a SALAD threshold of 0.95 (more relaxed than the 0.80 threshold used in [5]) with a retrieval verification threshold of 0.85 for α_{match} .

B. Pose estimation evaluation

Following the standard pose evaluation protocol which has been used in recent feed-forward SLAM papers such as [5, 42], we evaluate VGGT-SLAM 2.0 on the TUM RGB-D benchmark in Table I. For fairness, we use submaps of size 32 frames for VGGT-SLAM 2.0 as that is the submap size used by VGGT-SLAM in [5]. VGGT-SLAM 2.0 achieves the best overall average pose error of 4.1 cm out of all uncalibrated baselines which is approximately 23% lower than VGGT-SLAM and 22% lower than recent ViSTA-SLAM.

To provide quantitative pose evaluations on additional scenes, we summarize results from [50] in Table II, showing VGGT-SLAM 2.0 achieves substantially improved accuracy on ScanNet++ [51], ScanNetV2 [52], and Waymo [53].

C. Loop closure verification evaluation

We demonstrate that our image retrieval verification method which was presented in Section IV-B can help achieve more loop closures while preventing false positives. In Table III we compare the number of loop closures found by running VGGT-SLAM 2.0 on the three scenes of the Clio [40] datasets using a SALAD threshold of 0.80 (which was the value used in [5]) without using retrieval verification against the case where we relax the value to 0.95 and use the proposed retrieval verification. The office scene without verification led to false positive loop closures causing the reconstruction to diverge even with a SALAD threshold of 0.80 due to challenging similar looking areas such as the example from Fig. 5c. Meanwhile, using verification led to more loop closures and no false positives, even on the office scene. We show the beneficial effect this has on pose accuracy in Table IV, where running loop closure verification boosts VGGT-SLAM 2.0's accuracy on all three Clio scenes. We also remark that Table IV highlights that even without loop closure verification, VGGT-SLAM 2.0 still substantially surpasses VGGT-SLAM pose accuracy on all three Clio scenes as VGGT-SLAM diverges due to high dimensional drift.

To provide an additional experiment on another public dataset, in Table V we compare the Recall@1 score using two retrieval methods (SALAD [10] and NetVLAD [54]) with and without our verification. We use the LaMAR [55] HGE phone validation scene and label a retrieved frame as correct if it is within 25 m of the queried frame per standard practice [54]. In the case with verification, we take the top five retrieved frames from SALAD and NetVLAD respectively and score them using our α_{match} from Eq. (6) and return the one with the highest α_{match} to compute the Recall@1 score. We observe our verification procedure improves the results of both methods.

In order to further demonstrate that layer 22 is the optimal layer for our proposed image retrieval verification, in Fig. 6 we show an ablation using each of the 24 layers of VGGT on the LaMAR HGE phone validation scene (using 15 m for retrieval criteria). Surprisingly, while layer 22 performs the best, the worst performing layers are its neighbors, layers 21 and 23. We also remark that while layer 20 shows a close secondary

		360	desk	desk2	floor	plant	room	rpy	teddy	xyz	avg
Calibrated	ORB-SLAM3 [43]	X	0.017	0.210	X	0.034	X	X	X	0.009	-
	DeepV2D [44]	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
	DeepFactors [45]	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
	DPV-SLAM [46]	0.112	0.018	0.029	0.057	0.021	0.330	0.030	0.084	0.010	0.076
	DPV-SLAM++ [46]	0.132	0.018	0.029	0.050	0.022	0.096	0.032	0.098	0.010	0.054
	GO-SLAM [47]	0.089	0.016	0.028	0.025	0.026	0.052	0.019	0.048	0.010	0.035
	DROID-SLAM [20]	0.111	0.018	0.042	0.021	0.016	0.049	0.026	0.048	0.012	0.038
Uncalibrated	MAS3R-SLAM [42]	0.049	0.016	0.024	0.025	0.020	0.061	0.027	0.041	0.009	0.030
	CUT3R [25]	0.174	0.592	0.546	0.662	0.467	0.911	0.051	0.845	0.129	0.486
	SLAM3R [48]	0.211	0.861	0.967	0.790	0.755	1.013	0.063	0.986	0.185	0.648
	VGGT-Long [8]	0.063	0.059	0.045	0.108	0.057	0.172	0.056	0.137	0.051	0.083
	MapAnything [27]	0.084	0.058	0.067	0.072	0.049	0.209	0.045	0.059	0.025	0.074
	DROID-SLAM [20]	0.202	0.032	0.091	0.064	0.045	0.918	0.056	0.045	0.012	0.158
	MAS3R-SLAM [42]	0.070	0.035	0.055	0.056	0.035	0.118	0.041	0.114	0.020	0.060
	ViSTA-SLAM [7]	0.104	0.030	0.030	0.070	0.052	0.067	0.023	0.080	0.015	0.052
	VGGT-SLAM Sim(3) [5]	0.123	0.040	0.055	0.254	0.022	0.088	0.041	0.032	0.016	0.074
	VGGT-SLAM SL(4) [5]	0.071	0.025	0.040	0.141	0.023	0.102	0.030	0.034	0.014	0.053
	VGGT-SLAM 2.0	0.050	0.025	0.029	0.102	0.026	0.063	0.026	0.038	0.014	0.041

TABLE I. ROOT MEAN SQUARE ERROR (RMSE) OF ABSOLUTE TRAJECTORY ERROR (ATE (M)) ON TUM RGB-D [49].

Method	ScanNet++	ScanNetV2 [52]	Waymo [53]
DROID-SLAM [20]	0.623	0.217	7.535
MAS3R-SLAM [42]	0.346	0.106	16.610
VGGT-SLAM [5]	1.081	0.852	13.166
VGGT-SLAM 2.0	0.182	0.073	1.295

TABLE II. AVERAGE ATE RMSE (METERS).

Dataset	Without Retrieval Verification	With Retrieval Verification
Clio Cubicle	2	5
Clio Apartment	0	9
Clio Office	X	4

TABLE III. NUMBER OF LOOP CLOSURES ON THE CLIO DATASETS WITH AND WITHOUT OUR RETRIEVAL VERIFICATION. X DENOTES FALSE POSITIVE LOOP CLOSURES WERE PRESENT.

Method	Cubicle	Apartment	Office
VGGT-SLAM [5]	1.092	2.191	5.816
VGGT-SLAM 2.0 w/o ret. verif.	0.032	0.208	1.287
VGGT-SLAM 2.0 with ret. verif.	0.020	0.078	0.479

TABLE IV. RMSE OF ATE (METERS) ON CLIO SCENES.

	Without Retrieval Verification	With Retrieval Verification
SALAD [10]	88.45	90.13
NetVLAD [54]	85.92	89.08

TABLE V. RECALL@1 SCORES ON LAMAR HGE PHONE DATASET SHOWING IMPROVED RECALL WITH OUR PROPOSED RETRIEVAL VERIFICATION.

performance in Fig. 6, for practical deployment layer 22 is easier to threshold for retrieval verification to prevent false positives.

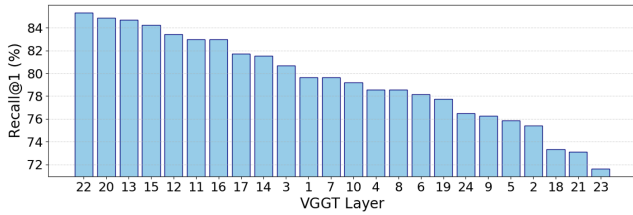


Fig. 6. Recall @1 on LaMAR dataset for all VGGT layers

D. Open-set semantic task evaluation

In this section, to further show the usability of VGGT-SLAM 2.0 for robotics, we demonstrate that VGGT-SLAM 2.0 can easily be modified to enable 3D open-set object detection.

Setup. During mapping, for every keyframe in a submap we compute an image embedding vector using the Perception Encoder [34] CLIP model. To locate a 3D object, the user inputs a text query, from which we compute a CLIP text embedding and locate the corresponding keyframe with the highest cosine similarity. As in [5], we have already stored all keyframes to disk memory so that they can be used during potential loop closures. We retrieve the corresponding

keyframe from memory and pass the frame to SAM 3 [56] along with the text query to perform 2D segmentation. Finally, if SAM 3 detects an object, we identify the points in the VGGT-SLAM 2.0 map which correspond to the masked region and compute a minimum 3D oriented bounding box around the points. This process is summarized in Fig. 7. The total query time from user text input to output 3D bounding box is approximately 0.36 seconds on a 3090 GPU.

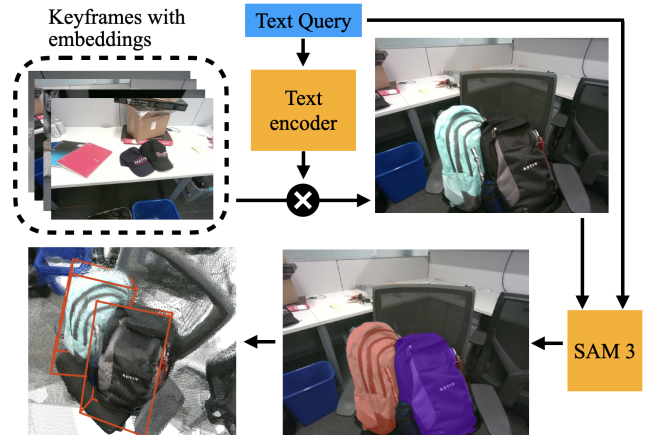


Fig. 7. Open-set object detection pipeline where an input text query, in this example “backpack”, is used to retrieve the best keyframe (which is passed to SAM 3 to perform 2D object segmentation) and finally produce a 3D detection and bounding box of the queried object.

Qualitative Results. We provide visual results of multiple objects detected in the VGGT-SLAM 2.0 map through open-

set querying on the Clio cubicle dataset in Fig. 8 which uses a RealSense D455 camera and in a custom apartment in Fig. 1 using an iPhone camera. We also show a large-scale barn scene in Fig. 9 where a large tractor is queried. These results show impressively crisp and reliable object extraction using a simple open-set query method built on top of VGGT-SLAM 2.0.

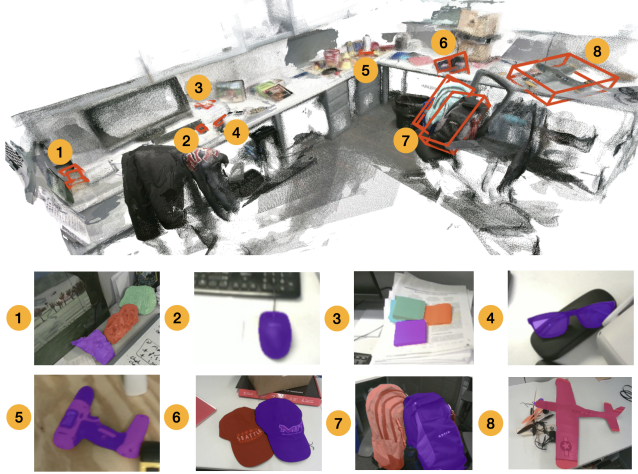


Fig. 8. Example 3D open-set object detection results on the Clio cubicle dataset showing estimated 3D oriented bounding boxes and SAM 3 segmentation masks on the queried keyframe. Text input for the 8 queries are: 1. rock, 2. mouse, 3. sticky notes, 4. glasses, 5. drill, 6. hat, 7. backpack, and 8. plane.

E. Additional qualitative results

In this section we provide additional qualitative results to show VGGT-SLAM 2.0 performing in a range of large environments. In Fig. 9 we show an example of a reconstruction of the inside and outside a 4200 square foot barn using custom data recorded with an iPhone camera. We also show an example of querying a large object (a tractor) with our open-set approach described in Section V-D.

While we design VGGT-SLAM 2.0 primarily for indoor datasets or small scale outdoor scenes, we also include an example from the Kitti [57] dataset in Fig. 10 to provide an example of running on long outdoor sequences to demonstrate improved robustness to long sequences. VGGT-SLAM diverges on the Kitti dataset due to long distances between loop closures causing rapid drift, and vulnerability to planar degeneracies as the majority of VGGT reconstructed points are on a flat road. Both of these scenes (which contain 34 submaps for the barn and 44 for Kitti) are larger than the largest scene shown in [5] which contained only 22 submaps of size 16 frames. Additional visual results are provided in the supplementary video.²

F. Real-time experiments and timing results

The runtime of VGGT-SLAM 2.0 is about 8.4 FPS for a submap of size 16 frames using a 3090 GPU if no open-set CLIP vectors are computed, and 6.3 FPS if open-set object



Fig. 9. VGGT-SLAM 2.0 reconstruction of a 4200 square foot barn with 34 submaps of size 16 frames using images captured from an iPhone. An example open-set query detecting a tractor inside the barn with the query “large tractor” is shown in the bottom figure.

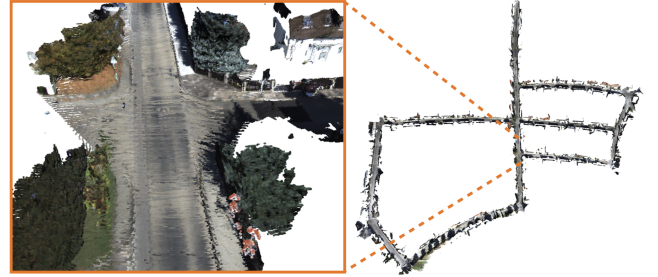


Fig. 10. Reconstruction of Kitti outdoor driving sequence 05 which contains 42 submaps of size 32 frames covering a driving distance over 2.2 km.

detection is enabled. FPS is computed by dividing the total time to handle a submap by the number of unique frames in the submap (which does not include the overlapping frames). A breakdown of the major timing components of VGGT-SLAM 2.0 is shown in Table VI. Over half of the time per submap is used to run VGGT (1248 ms) which means further reduction in computation can benefit from future faster variants of VGGT. Since timing is very hardware specific, we also run VGGT-SLAM and MAST3R-SLAM on the same machine to provide an approximate reference point for comparison. We observe that MAST3R-SLAM runs at a comparable 7.2 FPS and VGGT-SLAM runs at 6.9 FPS.

To demonstrate real-time usage of VGGT-SLAM 2.0 for robotics, we demonstrate running our entire system onboard a Jetson Thor mounted to a Jackal with a RealSense D455 camera. Using submaps of size 4, VGGT-SLAM 2.0 runs at 3.5

²<https://www.youtube.com/watch?v=GBdOvd6p4OU>

Stage	total time per submap (ms)
VGGT Inference	1248 ms
Keyframe Detection	176 ms
Loop Closure Detection	128 ms
Backend Optimization	8 ms
Semantics (optional)	368 ms

TABLE VI. AVERAGE RUNTIME PER SUBMAP IN MILLISECONDS OF THE MAJOR COMPONENTS OF VGGT-SLAM 2.0. TIMING RESULTS REPORTED USING A 3090 GPU FOR SUBMAP SIZE OF 16 FRAMES.

FPS onboard the Jetson Thor. Our supplementary video shows a VGGT-SLAM 2.0 map being created in real-time while the robot explores an office floor. Note that by comparison, VGGT-SLAM is not designed to process a live camera stream, only offline images.

VI. LIMITATIONS

While our proposed approach substantially improves VGGT-SLAM and other state-of-the-art methods, it is not without limitations. Through experiments on a variety of custom scenes, we did observe some cases of unsatisfactory results and include examples in the supplementary video to provide both a holistic evaluation and present possible directions of future improvements. One such experiment is the reconstruction of an empty four-bedroom house which contained keyframes which only saw a plain white wall causing VGGT reconstruction to fail. In this case, VGGT-SLAM 2.0 reconstruction will also diverge; possible future work could incorporate a lost-tracking module to recover from this. Additionally, our backend factor-graph optimization is only over poses and not points. While this greatly simplifies the SLAM pipeline (such as by not requiring tracking correspondences) and is effective in many of our experiments, in some cases it can cause artifacts from misaligned 3D points and in some challenging scenes leads to potentially preventable failure modes. Finally, the first frame of the submap is also unequally influential since not only does VGGT reconstruct all frames w.r.t. to first frame, but we also use the first frame to compute the scale factor of a submap. Possible future work could optimally select the first frame to both improve scale estimation and reduce VGGT failure modes.

VII. CONCLUSION

We have presented VGGT-SLAM 2.0, an improved approach to creating and aligning submaps with VGGT for dense SLAM with an uncalibrated monocular camera. Our new factor graph design removes the high-dimensional drift and planar degeneracies present in VGGT-SLAM while still accounting for the errors in camera calibration estimated from VGGT. We have conducted a study of the attention layers of VGGT and demonstrated how one of the layers can be leveraged for image retrieval verification, enabling more loop closures and protection against false positives. Finally, we have conducted a suite of experiments and demonstrations such as providing an open-set object adaptation of VGGT-SLAM 2.0 and a real time pipeline which can incrementally process a camera stream as a robot explores an environment while running onboard a Jetson Thor. Lastly, we remark that since

our method does not require any training, future improvements to the popular VGGT network such as lighter-weight or more robust variants can easily be plugged in to VGGT-SLAM 2.0.

ACKNOWLEDGEMENT

The authors gratefully thank Gary Maggio, Teresa Maggio, and Roubing Liao for assistance with data collection.

REFERENCES

- [1] L. Carlone, A. Kim, T. Barfoot, D. Cremers, and F. Dellaert, Eds., *SLAM Handbook. From Localization and Mapping to Spatial Intelligence*. Cambridge University Press, 2026.
- [2] S. Wang, V. Leroy, Y. Cabon, B. Chidlovskii, and J. Revaud, “Dust3r: Geometric 3d vision made easy,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 20 697–20 709.
- [3] V. Leroy, Y. Cabon, and J. Revaud, “Grounding image matching in 3d with mast3r,” in *European Conf. on Computer Vision (ECCV)*, 2024.
- [4] J. Wang, M. Chen, N. Karaev, A. Vedaldi, C. Rupprecht, and D. Novotny, “Vggt: Visual geometry grounded transformer,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [5] D. Maggio, H. Lim, and L. Carlone, “VGGT-SLAM: Dense RGB SLAM optimized on the SL(4) manifold,” in *Conf. on Neural Information Processing Systems (NeurIPS)*, 2025, (code).
- [6] R. Murai, E. Dexheimer, and A. J. Davison, “Mast3r-slam: Real-time dense slam with 3d reconstruction priors,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 16 695–16 705.
- [7] G. Zhang, S. Qian, X. Wang, and D. Cremers, “Vista-slam: Visual slam with symmetric two-view association,” *arXiv preprint arXiv:2509.01584*, 2025.
- [8] K. Deng, Z. Ti, J. Xu, J. Yang, and J. Xie, “Vggt-long: Chunk it, loop it, align it—pushing vggt’s limits on kilometer-scale long rgb sequences,” *arXiv preprint arXiv:2507.16443*, 2025.
- [9] 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,” *arXiv preprint arXiv:2509.20757*, 2025.
- [10] S. Izquierdo and J. Civera, “Optimal transport aggregation for visual place recognition,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2024.
- [11] G. Zhang and P. Vela, “Good features to track for visual slam,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [12] S. Baker and I. Matthews, “Lucas-kanade 20 years on: A unifying framework,” *Intl. J. of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.
- [13] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, “3d reconstruction of complex structures with bundle adjustment: an incremental approach,” in

- IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2006, pp. 3055–3061.
- [14] J. L. Schonberger and J.-M. Frahm, “Structure-from-motion revisited,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4104–4113.
 - [15] L. Pan, D. Barath, M. Pollefeys, and J. L. Schönberger, “Global Structure-from-Motion Revisited,” in *European Conf. on Computer Vision (ECCV)*, 2024.
 - [16] A. Davison, I. Reid, N. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun 2007.
 - [17] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.
 - [18] J. Engel, V. Koltun, and D. Cremers, “Direct sparse odometry,” *IEEE Trans. Pattern Anal. Machine Intell.*, 2018.
 - [19] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “Dtm: Dense tracking and mapping in real-time,” in *Intl. Conf. on Computer Vision (ICCV)*. IEEE, 2011, pp. 2320–2327.
 - [20] Z. Teed and J. Deng, “DROID-SLAM: Deep visual SLAM for monocular, stereo, and RGB-d cameras,” in *Advances in Neural Information Processing Systems (NIPS)*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.
 - [21] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, “NICE-SLAM: Neural implicit scalable encoding for slam,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022.
 - [22] S. Lovegrove, “Parametric dense visual SLAM,” Ph.D. dissertation, 2012.
 - [23] R. Madhavan, A. Fusiello, and F. Arrigoni, “Synchronization of projective transformations,” in *European Conf. on Computer Vision (ECCV)*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds. Springer Nature Switzerland, 2025, pp. 18–36.
 - [24] B. P. Duisterhof, L. Zust, P. Weinzaepfel, V. Leroy, Y. Cabon, and J. Revaud, “Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion,” in *2025 International Conference on 3D Vision (3DV)*. IEEE, 2025, pp. 1–10.
 - [25] Q. Wang, Y. Zhang, A. Holynski, A. A. Efros, and A. Kanazawa, “Continuous 3D Perception Model with Persistent State,” *arXiv preprint arXiv:2501.12387*, 2025.
 - [26] H. Wang and L. Agapito, “3D reconstruction with spatial memory,” *arXiv preprint arXiv:2408.16061*, 2024.
 - [27] N. Keetha, N. Müller, J. Schönberger, L. Porzi, Y. Zhang, T. Fischer, A. Knapitsch, D. Zauss, E. Weber, N. Antunes *et al.*, “Mapanything: Universal feed-forward metric 3d reconstruction,” *arXiv preprint arXiv:2509.13414*, 2025.
 - [28] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023.
 - [29] Z. Chen, J. Yang, and H. Yang, “Pref3r: Pose-free feed-forward 3d gaussian splatting from variable-length image sequence,” *arXiv preprint arXiv:2411.16877*, 2024.
 - [30] K. Li, M. Niemeyer, S. Wang, S. Gasperini, N. Navab, and F. Tombari, “Sing3r-slam: Submap-based indoor monocular gaussian slam with 3d reconstruction priors,” *arXiv preprint arXiv:2511.17207*, 2025.
 - [31] Z. Li, R. Tucker, F. Cole, Q. Wang, L. Jin, V. Ye, A. Kanazawa, A. Holynski, and N. Snavely, “Megasam: Accurate, fast and robust structure and motion from casual dynamic videos,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025, pp. 10 486–10 496.
 - [32] X. Chen, Y. Chen, Y. Xiu, A. Geiger, and A. Chen, “Ttt3r: 3d reconstruction as test-time training,” *arXiv preprint arXiv:2509.26645*, 2025.
 - [33] F. Zhang, T. Zhang, K. Khosoussi, Z. Zhang, Z. Huang, and Y. Luo, “Talo: Pushing 3d vision foundation models towards globally consistent online reconstruction,” *arXiv preprint arXiv:2512.02341*, 2025.
 - [34] D. Bolya, P.-Y. Huang, P. Sun, J. H. Cho, A. Madotto, C. Wei, T. Ma, J. Zhi, J. Rajasegaran, H. Rasheed, J. Wang, M. Monteiro, H. Xu, S. Dong, N. Ravi, D. Li, P. Dollár, and C. Feichtenhofer, “Perception encoder: The best visual embeddings are not at the output of the network,” in *Advances in Neural Information Processing Systems 38 (NeurIPS)*, 2025.
 - [35] M. Stry, J. Gaubil, A. Tewari, and V. Sitzmann, “Understanding multi-view transformers,” *arXiv preprint arXiv:2510.24907*, 2025.
 - [36] X. Chen, Y. Chen, Y. Xiu, A. Geiger, and A. Chen, “Easi3r: Estimating disentangled motion from dust3r without training,” *arXiv preprint arXiv:2503.24391*, 2025.
 - [37] J. Han, S. Hong, J. Jung, W. Jang, H. An, Q. Wang, S. Kim, and C. Feng, “Emergent outlier view rejection in visual geometry grounded transformers,” *arXiv preprint arXiv:2512.04012*, 2025.
 - [38] J. Bratulić, S. Mittal, T. Brox, and C. Rupprecht, “On geometric understanding and learned data priors in vgg,” *arXiv preprint arXiv:2512.11508*, 2025.
 - [39] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.
 - [40] D. Maggio, Y. Chang, N. Hughes, M. Trang, D. Griffith, C. Dougherty, E. Cristofalo, L. Schmid, and L. Carlone, “Clio: Real-time task-driven open-set 3D scene graphs,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 9, no. 10, pp. 8921–8928, 2024, (pdf),(video),(web).
 - [41] F. Dellaert and G. Contributors, “borglab/gtsam,” May 2022. [Online]. Available: <https://github.com/borglab/gtsam>
 - [42] R. Murai, E. Dexheimer, and A. J. Davison, “MASt3R-

- SLAM: Real-Time Dense SLAM with 3D Reconstruction Priors,” *arXiv preprint arXiv:2412.12392*, 2024.
- [43] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM,” *IEEE Trans. Robotics*, 2021.
 - [44] Z. Teed and J. Deng, “DEEPV2D: Video to depth with differentiable structure from motion,” *Intl. Conf. on Learning Representations (ICLR)*, 2018.
 - [45] J. Czarnowski, T. Laidlow, R. Clark, and A. Davison, “DeepFactors: Real-time probabilistic dense monocular SLAM,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.
 - [46] L. Lipson, Z. Teed, and J. Deng, “Deep patch visual SLAM,” in *European Conf. on Computer Vision (ECCV)*, 2024, pp. 424–440.
 - [47] Y. Zhang, F. Tosi, S. Mattoccia, and M. Poggi, “GO-SLAM: Global optimization for consistent 3D instant reconstruction,” in *Intl. Conf. on Computer Vision (ICCV)*, 2023, pp. 3727–3737.
 - [48] Y. Liu, S. Dong, S. Wang, Y. Yin, Y. Yang, Q. Fan, and B. Chen, “SLAM3R: Real-time dense scene reconstruction from monocular RGB videos,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.
 - [49] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 573–580.
 - [50] K. Ren, G. Li, C. Jiang, Y. Xu, T. Lu, L. Xu, J. Dong, J. Pang, M. Yu, and B. Dai, “M³: Dense matching meets multi-view foundation models for monocular gaussian splatting slam,” *arXiv preprint arXiv:2603.16844*, 2026.
 - [51] C. Yeshwanth, Y.-C. Liu, M. Nießner, and A. Dai, “ScanNet++: A high-fidelity dataset of 3d indoor scenes,” in *Intl. Conf. on Computer Vision (ICCV)*, 2023, pp. 12–22.
 - [52] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, “ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes,” *arXiv:1702.04405 [cs]*, Apr. 2017.
 - [53] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, “Scalability in perception for autonomous driving: Waymo open dataset,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.
 - [54] T. Cieslewski and D. Scaramuzza, “Efficient decentralized visual place recognition from full-image descriptors,” in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2017, pp. 78–82.
 - [55] P.-E. Sarlin, M. Dusmanu, J. L. Schönberger, P. Speciale, L. Gruber, V. Larsson, O. Miksik, and M. Pollefeys, “LaMAR: Benchmarking Localization and Mapping for Augmented Reality,” in *European Conf. on Computer Vision (ECCV)*, 2022.
 - [56] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, A. Huang, J. Lei, T. Ma, B. Guo, A. Kalla, M. Marks, J. Greer, M. Wang, P. Sun, R. Rädle, T. Afouras, E. Mavroudi, K. Xu, T.-H. Wu, Y. Zhou, L. Momeni, R. Hazra, S. Ding, S. Vaze, F. Porcher, F. Li, S. Li, A. Kamath, H. K. Cheng, P. Dollár, N. Ravi, K. Saenko, P. Zhang, and C. Feichtenhofer, “Sam 3: Segment anything with concepts,” 2025. [Online]. Available: <https://arxiv.org/abs/2511.16719>
 - [57] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012, pp. 3354–3361.