MV-RoMa: From Pairwise Matching into Multi-View Track Reconstruction

Supplementary Material

A. Grouping Images for SfM

In this section, we detail how we partition the full set of images into groups for the SfM pipeline, as in the multi-view reconstruction experiments in Sec. 4.3 and Sec. 4.4 of the main paper. Generally, SfM pipelines are formulated over large, unordered collections of images, whereas **MV-RoMa** processes only a small group at a time: one source image and up to K target images. Given such a group, the model predicts dense correspondences from the source to all target images, which are then aggregated into multi-view tracks and fed into the downstream SfM pipeline for 3D reconstruction (Sec. 3.5). Since it is computationally infeasible to run our model on all possible image groups in a scene, we need to select a set of groups that provide informative tracks for SfM.

Ideally, each group should include images that share sufficient scene overlap so that the model can generate abundant reliable correspondences. In addition, dense correspondences should be available in both directions ($I_a \rightarrow I_b$ and $I_b \rightarrow I_a$) for any image pair, to enable bidirectional consistency check (Sec. 3.5). To satisfy these requirements, we adopt a *two-stage sampling* scheme. In the first stage, we construct an initial set of image groups, under a fixed group budget G_{budget} (*i.e.* the number of groups we are allowed to construct), where images within each group have strong visual overlap. Then, in the second stage, we add a minimal number of extra groups so that every image pair has correspondences in both directions.

A.1. Sampling Procedure

First, we compute a pairwise overlap matrix $O \in [0,1]^{M\times M}$ over all images, where each entry o_{ij} is an *overlap score* between images i and j. We consider two simple strategies to define this score.

Visibility-based overlap. In our default setting, O is obtained from an off-the-shelf pairwise matcher. Let $\mathcal{P}_i$ denote the set of all 2D pixel locations in image i, and let $\mathcal{N}_{ij} \subseteq \mathcal{P}_i$ be the subset of pixels in image i that have a valid match in image j. Valid matches are determined from the matcher output, for example by thresholding a confidence score with a parameter τ_{conf}. We then define

$$o_{ij} = \frac{|\mathcal{N}_{ij}|}{|\mathcal{P}_i|} \tag{1}$$

so that o_{ij} measures the fraction of pixels in image i that have a valid correspondence in image j.

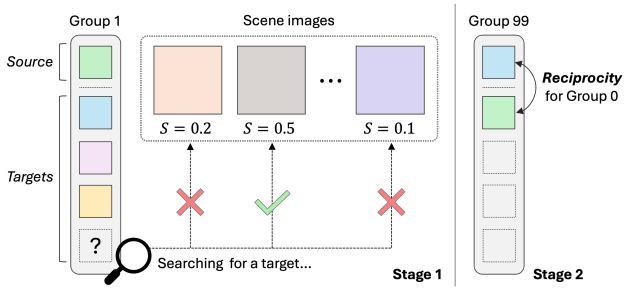


Figure 1. **Group Sampling Procedure.** We use a two-stage strategy: **Stage 1** greedily selects targets based on selection scores, while **Stage 2** generates additional groups to enforce reciprocity.

Feature-based overlap. In the setting where we only rely on image-level features, we instead define O based on global descriptors from image retrieval models [1, 2, 8]. Specifically, let f_i and f_j denote global descriptors for images i and j, respectively. We then define

$$o_{ij} = \frac{\langle f_i, f_j \rangle}{\|f_i\| \, \|f_j\|} \tag{2}$$

i.e. o_{ij} is the cosine similarity between the two descriptors, providing an appearance-based measure of overlap with low computational cost.

Source quotas for balanced sampling. Given the overlap matrix O, we first determine how often each image is allowed to serve as a source. Under a limited group budget G_{budget}, we prioritize using images that exhibit strong overlap with many others as sources, since they are likely to produce more useful matches. At the same time, we aim to prevent certain images from being repeatedly overused while ensuring that all images serve as sources at least once. To this end, we assign each candidate source image i a quota g_i that determines how many groups it can serve as a source for. Let N_i denote the number of high-overlap neighbors of image i above a fixed threshold τ, defined as:

$$N_i = \sum_{j=1}^{M} \mathbb{I}[o_{ij} > \tau] \tag{3}$$

We then set

$$g_i \propto (N_i + 1)^{\beta} \tag{4}$$

with an exponent $0 < \beta < 1$ (we use $\beta = 0.75$ by default). The additive term $+1$ guarantees that every image can act as a source at least once. In practice, we sample source images in proportion to g_i and keep forming groups until we reach the global budget G_{budget}.

Greedy group construction. Once a source image i is selected to construct a group, we add target images incrementally. At each step, we use a *selection score* that (i) favors strong overlap with the source, (ii) encourages coherence with already selected targets, and (iii) discourages repeated use of the same ordered pair (i, j). Let $\mathcal{T}_{\text{cur}}$ denote the current set of target images. For a candidate target image $j \notin \mathcal{T}_{\text{cur}}$, we define the selection score as:

$$\text{Score}(i, j \mid \mathcal{T}_{\text{cur}}) \;=\; \Big(\alpha_{\text{src}}\, o_{ij} + \alpha_{\text{tgt}} \sum_{k \in \mathcal{T}_{\text{cur}}} o_{kj}\Big) \cdot \frac{1}{1 + \lambda\, c_{ij}} \tag{5}$$

where $\alpha_{\text{src}}, \alpha_{\text{tgt}}, \lambda \geq 0$ are scalar hyperparameters, and $c_{ij} \in \mathbb{N}$ counts how many times the directed pair (i, j) has already been selected in previously constructed groups. The linear term in parentheses measures how well j fits the current group: o_{ij} captures alignment with the source, and $\sum_{k \in \mathcal{T}_{\text{cur}}} o_{kj}$ captures alignment with the existing targets, balanced by α_{src} and α_{tgt}. The multiplicative factor $(1 + \lambda c_{ij})^{-1}$ applies a soft penalty to repeated use of the same pair, encouraging the sampler to favor a more diverse set of image pairs. At each step, we evaluate this score for all remaining candidates $j \notin \mathcal{T}_{\text{cur}}$ and select the image with the highest score, until we reach the maximum group size (up to K targets) or no candidates remain.

Group augmentation for reciprocity. To support the bidirectional consistency check in Sec. 3.5, we add a second stage that ensures all image pairs are bidirectional by creating additional groups. Whenever a directed pair (i, j) is created during the first stage (*i.e.*, $i \rightarrow j$), we record that j needs to connect back to i. For each image j that still has such remaining reciprocity requirements, we create additional groups with j as a source image. In these groups, we first fill the target slots with images k that still require reciprocity with j. For remaining target slots, we add further targets chosen using the same selection score as above, but restricting candidates to images that have already been paired with j (either $j \rightarrow k$ or $k \rightarrow j$). This procedure avoids creating new one-sided pairs in the second stage while ensuring that all image pairs have dense correspondences in both directions.

B. Runtime and Efficiency

B.1. MV-RoMa's Runtime

We compare the inference time of MV-RoMa against RoMa at 448×448 resolution on an NVIDIA RTX 6000 Ada GPU. RoMa requires 85 ms for a single source–target pair (1-to-1) with a batch size of 1, which scales to 340 ms when processing five pairs in parallel with a batch size of 5. In contrast, MV-RoMa processes one source image jointly with five target images (1-to-5) in 280 ms.

B.2. SfM Runtime

In our experiments in the main paper, we set the group budget G_{budget} to around $N\sqrt{N}$, ensuring $\mathcal{O}(N\sqrt{N})$ complexity. (N is the number of images in a scene). We use $\sqrt{N}$ to reflect scene scale; all our evaluation scenes contain fewer than 100 images, and for large N, a constant value can replace $\sqrt{N}$. Within each group, matching for track clustering (via UFM) is performed K times (source-to-targets, where K is the number of target images) rather than combinatorially.

Table 1. Runtime comparison of each method for SfM reconstruction on a 20-image scene from the Texture-Poor SfM dataset.

Method	Budget	Match	Select & Filter	COLMAP	Refine	Total
DF-SfM w/ RoMa	-	1.5min	2sec	2.5min	1.5min	5.5min
MV-RoMa (Ours)	Full	3min	3sec	3min	-	6min
	Half	1.5min	2sec	2min	-	3.5min

To demonstrate the efficiency of MV-RoMa, Tab. 1 reports runtimes on the sub-scene *'20bag000'*(containing 20 images) from scene *1000* of the Texture-Poor SfM dataset, comparing MV-RoMa with DF-SfM+RoMa [5] using a single A6000 GPU. For DF-SfM+RoMa, we use exhaustive pairs following DF-SfM's evaluation protocol and additionally apply bidirectional consistency filtering (Sec.3.5.1) for fair comparison. Our *Half Budget* configuration (from 60 to 70 groups with each group containing 1-to-4 pairs), which reduces the group budget from $N\sqrt{N}$ to $\frac{1}{2}N\sqrt{N}$, achieves a total runtime of 3.5 minutes compared to DF-SfM's 5.5 minutes. This is mainly because MV-RoMa does not require track refinement performed in the DF-SfM pipeline, saving post-processing time after COLMAP reconstruction. As for per-group runtime, RoMa takes 0.25 seconds per 1-to-1 pair, while MV-RoMa takes 1.4 seconds per 1-to-4 group (including 0.25 seconds for running UFM per group). Note that the time cost for match selection and filtering (Sec.3.5.1) is negligible.

Table 2. Performance comparison between DF-SfM and MV-RoMa on the scene *1000* of the Texture-Poor SfM dataset.

Method	Budget	Texture-Poor AUC@3° / 5° / 10°
DF-SfM w/ RoMa	-	45.22 / 63.54 / 80.54
MV-RoMa (Ours)	Full	**60.71** / **74.66** / 85.88
	Half	59.97 / 74.60 / **86.23**

To evaluate the relationship between group budget for matching and pose accuracy, we compare camera pose estimation with full and half budgets in scene 1000 from the Texture-Poor SfM. As shown in Tab. 2, halving the budget results in only a marginal drop in accuracy, demonstrating that MV-RoMa's gains mainly stem from superior multi-view track consistency rather than high density of image pairs.

C. Training Data Construction

MegaDepth. For MegaDepth [6], we use the multi-view image groups provided by DF-SfM [5], which already ensure sufficient co-visibility within each group. From each group, we randomly select one source image and four target images.

ScanNet. For ScanNet [3], we build new multi-view groups from the two-view training split of LoFTR [9]. First, we derive candidate source–target pairs from LoFTR's training pairs and compute dense correspondences using ground-truth depth maps, camera poses, and intrinsics via geometric warping and verification. For each pair, we compute the *overlap ratio* as the ratio of the number of valid correspondences to the total number of pixels sampled for correspondence estimation in the source image, and keep only pairs with overlap ratio in the range $[15\%, 80\%]$ to discard both nearly disjoint and almost identical views. Next, to enforce view diversity among target images from the same source image, we compute pairwise overlap between every pair of candidate target images, similar to the overlap ratio applied between source and target. Whenever two targets overlap by more than 70%, we randomly remove one of them. From the remaining candidates, we form training groups by selecting four targets per source, ensuring that each frame appears at most once in the training set. This procedure yields multi-view groups with sufficient geometric overlap for learning, while avoiding redundant or overly similar frame combinations.

D. Analysis of Track Token

D.1. Flexibility Across Matchers

Table 3. Homography estimation on the HPatches dataset using different off-the-shelf matchers to generate track tokens.

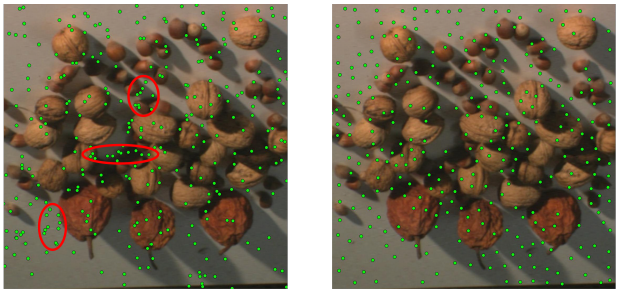
Matcher model	AUC @1px / @3px / @5px	
	DLT	RANSAC
ALIKED [10] + LightGlue [7]	42.9 / 70.3 / 79.1	42.9 / 71.0 / 80.2
UFM [4]	41.8 / 69.9 / 78.8	42.6 / 71.5 / 81.0

As discussed in Sec. 3.3 of the main paper, our framework can use any off-the-shelf matcher model to generate track tokens. To demonstrate this flexibility, we train and evaluate our model using ALIKED [10] and LightGlue [7] as the detector and matcher, respectively, applying the same clustering-based sampling strategy on the resulting matches to construct track tokens. Tab. 3 shows that using ALIKED+LightGlue achieves homography estimation performance comparable to using UFM [4]. This indicates that keypoint-based models can also provide effective geometric priors through track token construction.

D.2. Effect of Clustering-based Sampling

Table 4. Homography estimation on the HPatches dataset, comparing MV-RoMa's performance under clustering-based sampling versus random sampling.

Matcher model	Clustering	Error (median)	AUC @1px / @5px	
			DLT	RANSAC
ALIKED [10] + LG [7]		**0.50**	42.8 / 78.9	**43.1** / 80.1
	✓	**0.50**	**42.9** / **79.1**	42.9 / **80.2**
UFM [4]		0.53	41.5 / **78.8**	42.4 / 80.7
	✓	**0.51**	**41.8** / **78.8**	**42.6** / **81.0**



(a) Random Sampling (b) K-means Clustering

Figure 2. **Random vs Clustering.** **(a)** Random sampling results in spatially uneven distributions with redundant aggregations (red circles). **(b)** Our clustering-based sampling approach generates a spatially uniform distribution of track tokens, ensuring better coverage for feature exchange.

As explained in Sec. 3.3 of the main paper, we construct track tokens by applying clustering-based sampling to the initial matching results. As illustrated in Fig. 2, our k-means clustering yields a more spatially uniform distribution of tracks, whereas random sampling often produces highly aggregated keypoints in certain regions (red circles in Fig. 2.a). Such uneven coverage can leave substantial regions without any tracks, reducing the chance of retrieving information there and leading to suboptimal results. As reported in Tab. 4, clustering-based sampling outperforms random sampling in homography estimation.

D.3. Effect of Spatial Coverage

Table 5. Effect of reduced spatial coverage on homography estimation on HPatches.

Border Masked	AUC @1px / @3px / @5px	
	DLT	RANSAC
5%	41.9 / 69.9 / 78.8	42.5 / 71.3 / 80.7
10%	41.8 / 69.9 / 78.8	42.2 / 71.3 / 80.8
15%	41.4 / 69.9 / 78.9	41.7 / 70.7 / 80.2

To further validate the importance of spatial coverage, we conduct controlled experiments that artificially restrict where track tokens can be generated. As illustrated in

Figure 3. **Spatial Coverage Analysis.** We mask the outer 5%, 10%, and 15% border regions (bright area) while maintaining a fixed total number of track tokens ($T = 512$, only 300 tokens are shown for visualization). The experiment confirms that well-distributed tokens are essential for optimal results.

Fig. 3, we progressively suppress track tokens in the outer 5%, 10%, and 15% border regions, while keeping the total number of track tokens fixed at 512. As reported in Tab. 5, performance degrades as spatial coverage is reduced, confirming that well-distributed track tokens across the entire image are essential for optimal results. This finding highlights the importance of clustering-based sampling, which naturally promotes a more uniform spatial distribution of track tokens.

References

[1] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition, 2016. 1

[2] Gabriele Berton and Carlo Masone. Megaloc: One retrieval to place them all, 2025. 1

[3] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017. 3

[4] Yide Di, Yun Liao, Hao Zhou, Kaijun Zhu, Qing Duan, Junhui Liu, and Mingyu Lu. Ufm: Unified feature matching pre-training with multi-modal image assistants. *PloS one*, 20 (3):e0319051, 2025. 3

[5] Xingyi He, Jiaming Sun, Yifan Wang, Sida Peng, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Detector-free structure from motion. *CVPR*, 2024. 2, 3

[6] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[7] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. Lightglue: Local feature matching at light speed. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 17627–17638, 2023. 3

[8] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation, 2018. 1

[9] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021. 3

[10] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter C. Y. Chen, Qingsong Xu, and Zhengguo Li. Aliked: A lighter keypoint and descriptor extraction network via deformable transformation, 2023. 3