

Detector-free Image Matching with Lightweight Backbone and Feature Filtering

Xiaolong Guo¹, Min Wang^{2*}, Hui Wu¹, Wengang Zhou^{1,2}, Houqiang Li^{1,2}

¹ EEIS Department, University of Science and Technology of China

² IAI-ATCDI Joint R&D Center (Laboratory) on Intelligent Technologies for Comprehensive Transportation, Institute of Artificial Intelligence, Hefei Comprehensive National Science Center

{guoxiong3, wh241300}@mail.ustc.edu.cn, wangmin@iai.ustc.edu.cn, {zhwg, lihq}@ustc.edu.cn,

Abstract

Image Matching aims to establish correspondences between images, which plays a crucial role in camera pose estimation. Recent detector-free approaches have demonstrated remarkable progress in this field; however, they still suffer from low computational efficiency. The primary sources of their high time complexity are as follows: (1) the computationally expensive feature extraction backbone, and (2) the redundant feature updating process. To address these challenges, we propose an effective and efficient approach. Specifically, for issue (1), we design a lightweight backbone while preserving model performance through knowledge distillation and fine-tuning. For issue (2), we introduce a feature filtering module that retains only co-visible features, thereby reducing redundancy. To enhance the number of co-visible features, we leverage repeatability loss by formulating the feature filtering process as a keypoint detection task. Furthermore, to minimize hyperparameter tuning, the filtering thresholds are determined automatically. The proposed feature filtering module facilitates more accurate feature interaction while mitigating the impact of irrelevant match candidates. Extensive experiments on multiple datasets demonstrate that our approach significantly reduces computational cost while achieving slightly higher accuracy in pose estimation.

1. Introduction

Image matching, which aims to find correspondences between two images, is a fundamental task in computer vision. The correspondences are then used for camera pose estimation in applications such as Structure from Motion (SfM) and SLAM. These applications often require real-time computing, so the correspondences are supposed to be generated as fast as possible.

Nowadays, image matching methods can be categorized into detector-based ones [2, 4, 9, 10, 23, 25, 27, 28, 36] and detector-free ones [3, 5, 11–15, 17, 26, 30, 33, 37, 38].

*Corresponding author.

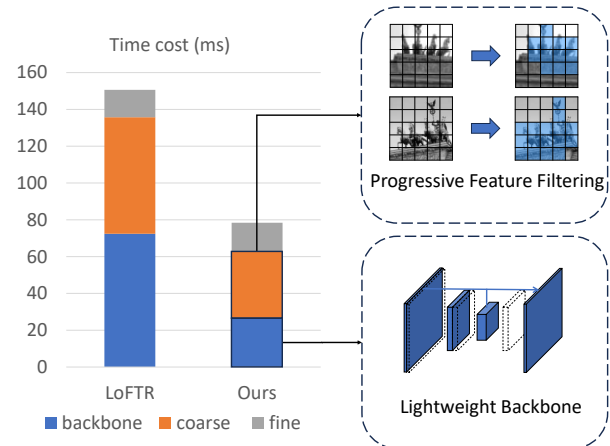


Figure 1. The time cost of each stage in LoFTR [30] and our method. With our lightweight backbone, the time cost of the backbone stage reduces by almost two-thirds. With our feature filtering module, the inference time of coarse matching stage is nearly halved, while the outdoor or indoor pose estimation performance is slightly better, as shown in experiments.

Detector-based methods like SIFT [23] and Superpoint [9] use detectors to generate keypoints and descriptors and conduct matching using mutual nearest neighbors(MNN) or neural matchers like SuperGlue [29] or LightGlue [21]. While having the advantage of fast speed because of sparse matching on keypoints, these methods encounter the keypoint detection error problem that not all useful matches can be identified through keypoints due to the loss of matched points in detection stage. So how to define keypoints is important in detector-based methods. In this regard, SIFT detects scale-invariant keypoints in scale space, while Superpoint learns the keypoints from the corner points of synthetic shapes and homographic adaptation. Both of them could not solve the detection error problem.

In contrast, detector-free methods like LoFTR [30] extract and match the feature densely rather than keypoint detection, thus avoiding the keypoint detection error and achieving better performance in pose estimation accuracy. However, extracting and matching features densely bring a

large amount of computation, which slows down the running speed of the methods and hinders them from a wider range of applications. In existing methods like LoFTR [30], as shown in Fig. 1, the majority of time cost comes from the backbone and coarse matching stage, which are responsible for feature extraction and updating, respectively. For feature extraction, it remains unknown how simple a backbone can be while still being capable of performing the feature extraction we want. This indicates that the backbone of existing methods could be too complex and consume unnecessary time. For feature updating, existing methods update all features even if some features have no corresponding features in another image, meaning they are non-co-visible. Intuitively, updating non-co-visible features provides no benefit to matching. It not only interferes with the updating of useful features from co-visible area but also introduces unnecessary computation and time overhead.

To address these two issues of detector-free methods, we propose an effective and efficient image matching method, which introduces a lighter backbone and a feature filtering module. In the lighter backbone, we reduce the number of channels and layers based on the FPN [19]. To ensure our backbone has a comparable feature extraction capability, we train it in a two-stage manner including distillation and fine-tuning. In distillation, we use LoFTR’s backbone which is already trained as the teacher, and align their output with L2 loss. In fine-tuning, we further train the backbone with other frozen modules end to end. In the feature filtering module, we use classification networks to classify whether the feature patch is co-visible or not after each cross-attention layer of LoFTR, since the features generated by the cross-attention layers contain cross-image information. We also consider the co-visibility scores as keypoint scores in detector-based methods, so we additionally use keypoint losses to learn the network. For instance, we use repeatability loss from R2D2 [27] to improve the scores to facilitate that the corresponding feature patches have similar scores and can be preserved together. In inference, to avoid setting N hyper-parameters for N feature filtering layer’s thresholds, we design dynamic thresholds for each layer by using the moving average of the lowest scores of co-visible feature patches in training.

As shown in Fig. 1, by applying our proposed backbone and feature filtering module on existing detector-free methods, we can reduce the time cost of the backbone and coarse matching stages, therefore improving the running speed of the entire network. Compared to detector-based methods, our approach maintains the performance of detector-free methods, offering an advantage in pose estimation accuracy. In addition, the feature filtering module not only reduces the time cost, but also brings the benefit of more accurate matches. First, the feature interaction is more accurate since the participants are mostly from co-visible area. Second,

matching interference is less since the candidates from non-co-visible areas are filtered out in advance. Experiments on various datasets show that our method can improve the speed of LoFTR by a large margin, while preserving similar or better pose estimation performance.

2. Related Work

2.1. Image Matching

Image matching methods can be categorized into detector-based ones and detector-free ones. Detector-based methods use detectors to detect keypoints and generate corresponding descriptors for image pairs. Keypoints are matched through the descriptors. Among them, SIFT [23] is a traditional one, which can generate scale-invariant keypoints and descriptors. While SIFT is time-consuming, ORB [28] can run at a faster speed and thus is more popular in applications like SLAM and SFM. Compared to detector-based methods like Superpoint [9] and LightGlue [21], detector-free methods like LoFTR [30] have leading performance, but they also face the slow inference speed, which hinders their applications. Our work focuses on improving the speed of detector-free methods, especially for LoFTR-like methods.

2.2. Feature Filtering in Image Matching

Keypoint detection in detector-based methods, as well as the feature pruning process in LightGlue, can be considered as a kind of feature filtering, which brings the effect of fast speed. In contrast, detector-free methods extract, update, and match features densely without feature selections. However, there are methods that do feature filtering implicitly. OETR [7] detects co-visible rectangle areas in image pairs and then runs image matching on these areas. MESA [35] uses SAM [16] to detect areas with semantics and runs image matching in them. These methods reduce the space needed to be searched for image matching and thus reduce the time cost. Our method is motivated by co-visible area detection and feature pruning in LightGlue, progressively filtering the features that are not from co-visible area in the feature updating process of detector-free image matching methods.

3. Method

3.1. Overview

The goal of our method is to improve the inference speed of detector-free methods for image matching, without damaging the performance. Given two images A and B , our image matching method aims to find correspondences $\{(i, j)\}$ in these two images. As illustrated in Fig. 2, there are three stages in our approach:

The first stage is feature extraction. A lighter backbone based on FPN [19] is used to extract coarse feature maps

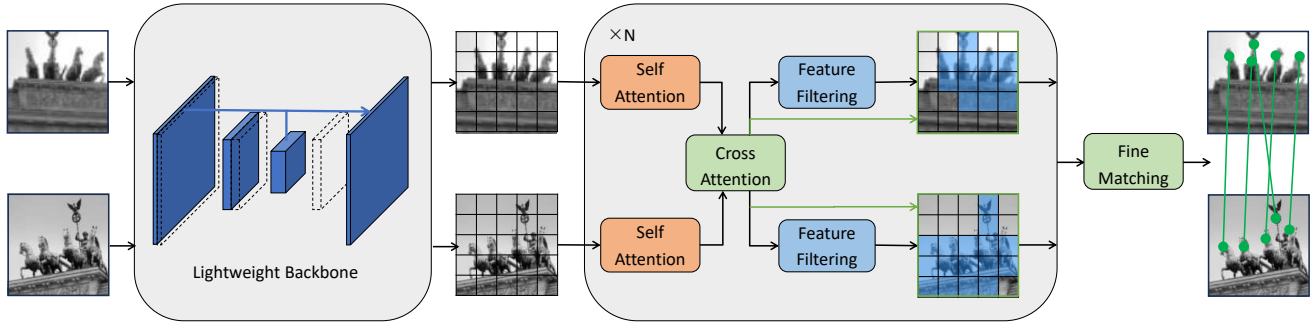


Figure 2. An overview of our approach. In the lightweight backbone, the dashed line indicates the dimension or layer we reduce. Based on FPN, we reduce the mid-layer dimensions and skip the $1/4 \times 1/4$ feature map. The feature filtering module is added after each cross-attention layer. We detect co-visible features and filter out those that are not co-visible.

and fine feature maps from the images. The lightweight backbone has a smaller dimension of intermediary variables and fewer layers for fine feature extraction, thus consuming less time.

The second stage is coarse matching. Coarse feature maps are updated through self-attentions and cross-attentions alternatively, and then are matched through dual softmax and mutual nearest neighbors (MNN) to produce coarse matches. A feature filtering module is proposed to filter out features corresponding to the area that are not co-visible across images. This is implemented by classification after each cross-attention layer. The goal is to reduce the time cost of feature updating and matching.

The third stage is fine matching. Similar to LoFTR [30], local windows are cropped from fine feature maps at the positions of coarse matches. Fine matches are refined from coarse matches in the local windows.

3.2. Lightweight Backbone

The backbones in existing detector-free methods are unnecessarily complex and consume a lot of time. So here we explore how simple the backbone could be when it keeps the ability of feature extraction. Our optimization on the backbone includes two folds:

The first is using less channels for mid-layers. A straightforward way to reduce the cost of a network is to use fewer channels for mid-layers in the network. To keep the output dimensions unchanged, we use 1×1 convolution.

The second is using less layers for fine feature extraction. In LoFTR, fine features are compared only in small local windows and thus require less information. This motivates us to reduce the number of layers for fine feature extraction. Specifically, we skip the generation of $1/4$ feature map and directly produce $1/2$ feature map from previous $1/2$ feature map and $1/8$ feature map, and use 1×1 convolution instead of 3×3 .

For training, since our method plugs the feature filtering module into trained LoFTR without changing its weights, it's a good choice for the lightweight backbone to imitate

the output of the original backbone in LoFTR. For this reason, we distill the LoFTR FPN into our lightweight backbone by aligning their outputs with L2 loss. The distillation could be not perfect yet, so we then fine-tune the backbone with other trained and frozen LoFTR modules end-to-end, with all LoFTR supervisions.

3.3. Feature Filtering Module

The feature updating process in existing detector-free methods is redundant since they update useless features from non-co-visible areas. This motivates us to design the feature filtering module.

Non-co-visible area features filtering is beneficial. Only features belonging to co-visible areas are useful in feature matching. Therefore, filtering out features corresponding to the non-co-visible area is less likely to harm the matching results while being able to reduce the time for feature updating and matching. Moreover, co-visible area feature filtering can make feature interaction more accurate because useless features are removed and never attended to.

It needs cross-information from both images to classify the co-visibility of features. The cross-attention layers for feature updating in LoFTR provide features with cross-information. So we run co-visibility classification after each cross-attention layer. Specifically, for the classification after the l -th cross-attention layer, a score is calculated from each feature for each area with a linear layer and a sigmoid layer:

$$S_i^l = \text{Sigmoid}(\text{Linear}_l(F_i^l)). \quad (1)$$

The score of a feature reflects to what extent the feature belongs to co-visible areas.

In inference, feature filtering is run after co-visibility classification. After the classification after the l -th cross-attention layer, the features are filtered as:

$$F^l = \{F_i^{l-1} | S_i \geq \sigma^l\}, \quad (2)$$

where F^0 is the input feature map to the coarse module. When classification goes forward after different layers, the

feature it makes use of contains more cross-information. This enables the classification and filtering to be progressively more accurate.

To avoid setting hyper-parameters for thresholds at different layers, we use the moving average of the lowest scores of features from co-visible areas in training as the thresholds in inference. That is, at l -th layer, after training step t :

$$s_t^l = \min\{S_{i,t}^l | C_i = 1\}$$

$$\sigma_t^l = \begin{cases} \beta_1 \sigma_{t-1}^l + (1 - \beta_1) s_t^l, & s_t^l < \sigma_{t-1}^l, \\ \beta_2 \sigma_{t-1}^l + (1 - \beta_2) s_t^l, & \text{otherwise}, \end{cases} \quad (3)$$

where C is the ground truth indicating co-visible areas and $\beta_1 = 0.9$ is lower than $\beta_2 = 0.999$, in order to make the threshold more sensitive to lower scores in co-visible areas.

3.4. Supervision of Feature Filtering Module

The first loss is Co-visibility classification loss. We treat the co-visibility classification as a per-pixel binary classification task. To improve the scores of features from co-visible areas, and decrease the scores of features that are not, we use Focal Loss [20](abbr. as FL hereafter):

$$\mathcal{L}_{co-visible} = \sum_l FL(S^l, C). \quad (4)$$

The second loss is Repeatability loss. We also consider the co-visibility classification as a keypoint detection task. We borrow the idea of keypoint repeatability from R2D2 [27]. Specifically, for images A and B , we maximize the cosine similarity of corresponding feature scores:

$$\mathcal{L}_{cosim} = \sum_l 1 - \cosim(S_A^l[p_A], S_B^l[p_B]), \quad (5)$$

where $\{(p_A)_i, (p_B)_i\}$ are indexes of ground truth matching pairs. $\mathcal{L}_{cosim}$ helps preserve more corresponding features in feature filtering since it forces the corresponding features to have similar scores.

To prevent the scores of corresponding features from being constant, we also use the peaky loss from R2D2:

$$\mathcal{L}_{peaky} = \sum_l (1 - (\max S_A^l[p_A] - \text{mean} S_A^l[p_A]) + 1 - (\max S_B^l[p_B] - \text{mean} S_B^l[p_B])). \quad (6)$$

We use the losses above to supervise the classification networks in the feature filtering module:

$$\mathcal{L} = \mathcal{L}_{co-visible} + \mathcal{L}_{cosim} + \mathcal{L}_{peaky}. \quad (7)$$

After training the lightweight backbone with other LoFTR modules, we freeze the LoFTR weights and only train the feature filtering module (i.e. the linear layers). We use the same initial learning rate and other architecture hyper-parameters with LoFTR.

3.5. Inference

After training the lightweight backbone and the feature filtering module progressively, we replace the backbone in LoFTR with our lightweight one and plug the feature filtering module after each cross-attention layer in LoFTR. Then we infer with the whole network, where the filtering module filters features in the feature set that comes into the next layer.

4. Experiment

4.1. Relative Pose Estimation

We use ScanNet [8], MegaDepth [18] and YFCC100M [31] to demonstrate the effectiveness of our method for pose estimation in indoor and outdoor scenes.

ScanNet is a dataset of indoor scenes. ScanNet consists of 1613 indoor sequences as well as ground-truth poses and depth maps. we follow the training and testing split used by LoFTR, in which 1.5K images are used in validation. All images are resized to 640×480 . The key challenge on ScanNet is matching the texture-less regions.

MegaDepth is a dataset of outdoor scenes. MegaDepth contains 1M internet images of 196 different outdoor scenes and their depth maps. We follow LoFTR to build the validation set, only using 1500 pairs from the scenes of ‘‘Sacre Coeur’’ and ‘‘St. Peter’s Square’’. Images are resized to ensure that their longer dimensions are equal to 840 for both training and validation. MegaDepth contains image pairs with extreme viewpoint changes and repetitive patterns, which brings challenges.

YFCC100M is also a dataset of outdoor scenes. YFCC100M is composed of 100 million media objects. We follow PATS [26] to evaluate our method on a subset of YFCC100M, which includes 4 selected image collections of popular landmarks.

Following LoFTR, we train and evaluate the indoor model on ScanNet, while training the outdoor model on MegaDepth and evaluating it on Megadept and YFCC100M. We report the AUC of the pose error at thresholds (5° , 10° , 20°). We use RANSAC as the solver to solve the essential matrix from predicted matches.

Table 1 shows the experiment results in the indoor pose estimation task on ScanNet, where we compare our method with both detector-based and detector-free methods. The result consistently shows that our method brings improvement to LoFTR-like methods and surpasses other methods. This illustrates that our components will not do harm to the performance of existing networks when plugged into them.

Table 2 and Table 3 show the experiment results in the outdoor pose estimation task on MegaDepth and YFCC100M, in which we compare our method with both detector-based and detector-free methods. The result shows that our method brings improvement to LoFTR-like meth-

Table 1. **Evaluation on scannet [8] for indoor pose estimation.** †indicates our running results. ‡indicates model trained on megadepth.

Category	Method	Pose estimation AUC		
		@5°	@10°	@20°
Detector-based	D2-Net [10]+NN	5.3	14.5	28.0
	ContextDesc [24]+Ratio Test [23]	6.6	15.0	25.8
	SP [9]+NN	9.4	21.5	36.4
	SP [9]+PointCN [34]	11.4	25.5	41.4
	SP [9]+OANet [22]	11.8	26.9	43.9
	SP [9]+SGMNet [6]	15.4	32.1	48.3
	SP [9]+SuperGlue [29]	16.2	33.8	51.8
Detector-free	DRC-Net [17]	7.7	17.9	30.5
	LoFTR [30] †	22.35	41.06	57.81
	LoFTR [30] + Ours †	23.14	41.82	58.25
	AdaMatcher-LoFTR [15] †‡	18.32	34.76	50.54
	AdaMatcher-LoFTR [15] + Ours †‡	18.68	35.59	51.46

Table 2. **Evaluation on megadepth [18] for outdoor pose estimation.** †indicates our running results.

Category	Method	Pose estimation AUC		
		@5°	@10°	@20°
Detector-based	SP [9]+NN	31.7	46.8	60.1
	SP [9]+SGMNet [6]	40.5	59.0	73.6
	SP [9]+SuperGlue [29]	42.2	61.2	76.0
	SP [9]+LightGlue [21]	49.9	67.0	80.1
Detector-free	DRC-Net [17]	27.01	42.96	58.31
	LoFTR [30] †	51.98	68.82	80.84
	LoFTR [30] + Ours †	52.81	69.22	81.08
	AdaMatcher-LoFTR [15] †	52.05	68.97	81.52
	AdaMatcher-LoFTR [15] + Ours †	52.97	69.89	82.26

Table 3. **Evaluation on yfcc100m [31] for outdoor pose estimation.** †indicates our running results.

Category	Method	Pose estimation AUC		
		@5°	@10°	@20°
Detector-based	RootSIFT [1]+SGMNet [6]	35.5	55.2	71.9
	SP [9]+SuperGlue [29]	38.7	59.1	75.8
Detector-free	DRC-Net [17]	29.5	50.1	66.8
	PDC-Net+ [32]	39.1	60.1	76.5
	LoFTR [30] †	45.4	64.1	78.3
	LoFTR [30] + Ours †	46.0	64.5	78.4
	AdaMatcher-LoFTR [15] †	46.14	64.17	78.26
	AdaMatcher-LoFTR [15] + Ours †	45.60	64.20	78.36

ods, including LoFTR [30] and Adamatcher-LoFTR [15], and outperforms other methods. This indicates that our method can keep the advantages of existing networks when

Table 4. **Inference time on MegaDepth [18]. Our method reduces LoFTR’s time cost by half.**

Runtime (ms/pair)	backbone	coarse matching	fine matching	total
LoFTR [30]	72.42	63.32	14.87	150.61
LoFTR [30]+ours	26.73	36.17	15.49	78.39

Table 5. **Number of parameters and FLOPs.**

methods	Parameters	FLOPs
LoFTR	11.6M	829.33G
LoFTR+Ours	8.5M	278.44G

Table 6. **Ablation of our method applied to LoFTR on Megadepth.** The lightweight backbone, feature filtering module, repeatability loss, and dynamic threshold are evaluated respectively on pose estimation AUC and time cost.

	Light Weight Backbone	Feature Filtering	Repeatability Loss	Dynamic Thresholds	Pose Estimation AUC @5°	@10°	@20°	Time Cost
(i)					52.0	68.8	80.8	150ms
(ii)	✓				52.3	68.4	80.6	104ms
(iii)	✓	✓	✓		50.2	67.0	79.6	86ms
(iv)	✓	✓		✓	50.6	67.2	79.7	79ms
(v)	✓	✓	✓	✓	52.8	69.2	81.1	78ms

plugged into them.

Our method brings improvements to the baseline. The reason why our method brings benefits may include two aspects: (1) The features are updated in a more accurate way since non-co-visible features are filtered out and don’t take part in the feature interaction process. (2) There is less interference in coarse matching since most candidates from non-co-visible areas are filtered in advance.

4.2. Runtime Evaluation

To test the timing of inference, we collect the time statistics in the evaluation on MegaDepth. Table 4 shows that our method brings nearly half time savings to LoFTR on Megadepth. Specifically, the time cost of the backbone is reduced by almost two-thirds. The inference time of the coarse matching stage in LoFTR is nearly halved. The results show the effectiveness of our components in achieving our goal of fast speed detector-free methods.

4.3. Parameter and Computational Cost Evaluation

We evaluate the number of parameters and the FLOPs of both the LoFTR model and our proposed method using the `thop` library. The results are presented in Table 5. As shown, our method exhibits a reduction in the number

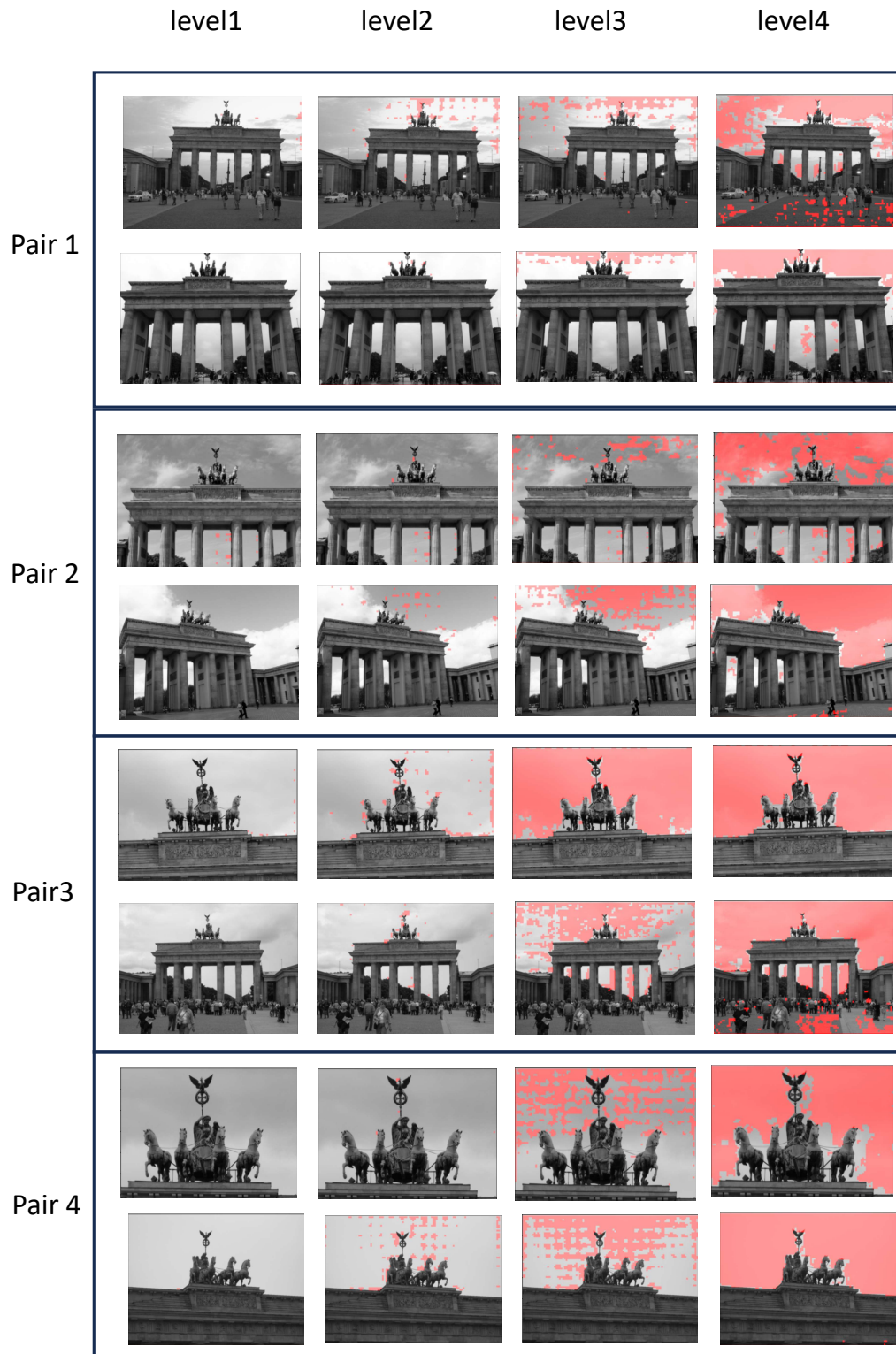


Figure 3. Visualization of area filtered by progressive feature filtering module at each level on MegaDepth dataset. Light red areas are filtered out.

of parameters compared to LoFTR, which is primarily attributed to the lightweight design of our backbone network and the simplicity of the feature filtering module. Moreover, our method demonstrates a substantial decrease in computational cost. This improvement stems not only from the streamlined backbone but also from the effective reduction in the number of features that need to be updated and matched, enabled by the proposed filtering module. Overall, our approach offers significant advantages in terms of computational efficiency.

4.4. Ablation Study

We ablate our work in several configurations on MegaDepth and LoFTR [30], which are listed in Table 6. In (ii), we evaluate the effect of our proposed lighter backbone by adding it to the baseline. The conclusion is that the backbone can achieve time cost reduction and comparable accuracy. In (v), we add our backbone and full feature filtering module to the baseline. By comparing (i), (ii), and (v), we can see that the feature filtering module can further improve the performance and accuracy. In (iii), we replace the dynamic threshold with fixed thresholds (0.01), and find a performance drop by compared to (v). In (iv), we remove the repeatability loss in training and also get worse results compared to (v). These results show the effectiveness of our dynamic threshold and repeatability loss designs.

4.5. Visualizations

We visualize the area filtered by our module at each level in Fig. 3 on MegaDepth dataset. We can see that at the first two levels, little area is masked as not co-visible and filtered. One possible reason behind this is that the features at the first two levels are not adequate to infer whether the area this feature belongs to is co-visible or not. At the third or fourth levels, the feature contains more cross-information after interaction between two images. As a result, the majority of non-co-visible areas are detected. The whole process is progressively filtering out non-co-visible area features, which is fully consistent with our idea.

5. Conclusion

In this paper, we propose a lighter backbone and a progressive feature filtering module for LoFTR-like detector-free image matching methods. The backbone is carefully trained by distillation and fine-tuning, thus able to preserve the performance while reducing large time cost. The progressive feature filtering module is ready to be plugged into cross-attention layers in the network and is able to improve the efficiency of the feature interaction stage by pruning the feature patches out of co-visible areas. The progressive feature filtering module also improves the matching accuracy by more accurate feature interactions and matching processes with less interference from non-co-visible areas. We further

use repeatability loss and propose dynamic threshold to improve the network. One is for more correspondence and the other is for less hyper-parameters. Experiments on various datasets show significant speed improvement, while the accuracy is even slightly better. In the future, we may apply the idea of feature filtering to more networks.

References

- [1] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2012. 5
- [2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, page 346–359, 2008. 1
- [3] Georg Bökman and Fredrik Kahl. A case for using rotation invariant features in state of the art feature matchers. pages 5110–5119, 2022. 1
- [4] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. *BRIEF: Binary Robust Independent Elementary Features*, page 778–792. 2010. 1
- [5] Jiahao Chang and Jiahuan Yu. Structured epipolar matcher for local feature matching. pages 6177–6186, 2023. 1
- [6] Hongkai Chen, Zixin Luo, Jiahui Zhang, Lei Zhou, Xuyang Bai, Zeyu Hu, Chiew-Lan Tai, and Long Quan. Learning to match features with seeded graph matching network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6301–6310, 2021. 5
- [7] Ying Chen, Dihe Huang, Shang Xu, Jianlin Liu, and Yong Liu. Guide local feature matching by overlap estimation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 365–373, 2022. 2
- [8] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 4, 5
- [9] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018. 1, 2, 5
- [10] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8092–8101, 2019. 1, 5
- [11] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. Dkm: Dense kernelized feature matching for geometry estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17765–17775, 2023. 1
- [12] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten

- Wadenbäck, and Michael Felsberg. RoMa: Robust Dense Feature Matching. *arXiv preprint arXiv:2305.15404*, 2023.
- [13] KhangTruong Giang, Soohwan Song, and Sungho Jo. Topicfm: Robust and interpretable feature matching with topic-assisted. 37(2):2447–2455, 2022.
- [14] KhangTruong Giang, Soohwan Song, and Sungho Jo. Topicfm+: Boosting accuracy and efficiency of topic-assisted feature matching. *IEEE Transactions on Image Processing*, 2023.
- [15] Dihe Huang, Ying Chen, Yong Liu, Jianlin Liu, Shang Xu, Wenlong Wu, Yikang Ding, Fan Tang, and Chengjie Wang. Adaptive assignment for geometry aware local feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5425–5434, 2023. 1, 5
- [16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 2
- [17] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual-resolution correspondence networks. *Advances in Neural Information Processing Systems*, 33:17346–17357, 2020. 1, 5
- [18] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2041–2050, 2018. 4, 5
- [19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2
- [20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 4
- [21] 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. 1, 2, 5
- [22] Xin Liu and Jufeng Yang. Progressive neighbor consistency mining for correspondence pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9527–9537, 2023. 5
- [23] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. 1, 2, 5
- [24] Zixin Luo, Tianwei Shen, Lei Zhou, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Contextdesc: Local descriptor augmentation with cross-modality context. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2527–2536, 2019. 5
- [25] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, page 438–469, 2009. 1
- [26] Junjie Ni, Yijin Li, Zhaoyang Huang, Hongsheng Li, Hujun Bao, Zhaopeng Cui, and Guofeng Zhang. Pats: Patch area transportation with subdivision for local feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17776–17786, 2023. 1, 4
- [27] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2, 4
- [28] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *International Conference on Computer Vision*, pages 2564–2571. Ieee, 2011. 1, 2
- [29] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4938–4947, 2020. 1, 5
- [30] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8922–8931, 2021. 1, 2, 3, 5, 7
- [31] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: the new data in multimedia research. *Commun. ACM*, 59(2):64–73, 2016. 4, 5
- [32] Prune Truong, Martin Danelljan, Radu Timofte, and Luc Van Gool. Pdc-net+: Enhanced probabilistic dense correspondence network. *arXiv preprint arXiv:2109.13912*, 2021. 5
- [33] Yifan Wang, Xingyi He, Sida Peng, Dongli Tan, and Xiaowei Zhou. Efficient loftr: Semi-dense local feature matching with sparse-like speed. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21666–21675, 2024. 1
- [34] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2666–2674, 2018. 5
- [35] Yesheng Zhang and Xu Zhao. Mesa: Matching everything by segmenting anything. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20217–20226, 2024. 2
- [36] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter CY Chen, Qingsong Xu, and Zhengguo Li. Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. *IEEE Transactions on Instrumentation and Measurement*, 72:1–16, 2023. 1
- [37] Qunjie Zhou, Torsten Sattler, and Laura Leal-Taixe. Patch2pix: Epipolar-guided pixel-level correspondences. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4669–4678, 2021. 1
- [38] Shengjie Zhu and Xiaoming Liu. Pmatch: Paired masked image modeling for dense geometric matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21909–21918, 2023. 1