Supplementary Material: ArgMatch: Adaptive Refinement Gathering for Efficient Dense Matching

Yuxin Deng[1] Kaining Zhang[2] Linfeng Tang[1] Jiaqi Yang[3] Jiayi Ma[1] *
[1]Wuhan University, China [2]Hunan University, China
[3]Northwestern Polytechnical University, China
{acuo.dyx, zkn707196, linfeng0419, jyma2010}@gmail.com jqyang@nwpu.edu.cn

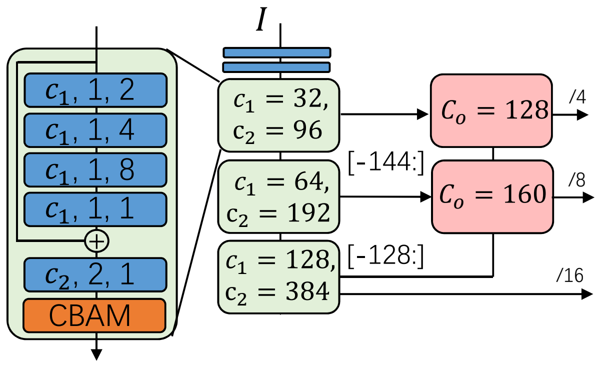

Figure 1. **Feature extractor**. The blue blocks labeled with A, B, C represent a 3x3 convolutional layer with an output channel size of A, a stride of B, and a dilation of C. CBAM refers to the convolutional attention module [8]. The notation [...] indicates indexing a subset of channels and passing them to the next layer.

1. Details of Model Design

In this section, we strive to provide a detailed description of the key components and operations in our model design, with a particular focus on the feature extractor and the global matcher. These elements are critical to the overall architecture and performance of our model.

1.1. Feature Extractor

As depicted in Fig. 1, our feature extractor utilizes a U-Net architecture [5]. In the downsampling branch, half-resolution features are first extracted using several convolutional layers (shown in blue). These features are then processed by a sequence of blocks, each of which first enhances the features via dilated convolutions [10] without increasing the number of channels. This is followed by downsampling, accompanied by an increase in channel width, and further refinement through a Convolutional Block Attention Module (CBAM) [8] augmented with depthwise separable convolutions [2] to to efficiently capture long-range context. Note that, each block selects the first c_1 channels of its input feature for processing. After three such blocks, a hierarchical feature pyramid is constructed.

In the upsampling branch, multi-scale features are aggregated using fusion blocks (shown in red). Each fusion block integrates low-resolution and high-resolution inputs by concatenating the upsampled low-resolution feature with the corresponding high-resolution feature. The concatenated representation is then refined through element-wise multiplication of two distinct feature mappings, effectively highlighting salient information. Finally, a convolutional layer compresses the refined feature to the target output dimension C_o.

The final output of the feature extractor is a feature pyramid comprising 384-dimensional features at 1/16 resolution, 160-dimensional features at 1/8 resolution, and 128-dimensional features at 1/4 resolution. The 384-dimensional feature map is further split into overlapping 224-dimensional and 192-dimensional components. The 224-dimensional features at 1/16 resolution are used for global matching, while the 192-, 160-, and 128-dimensional features at 1/16, 1/8, and 1/4 resolutions, respectively, are employed for refinement.

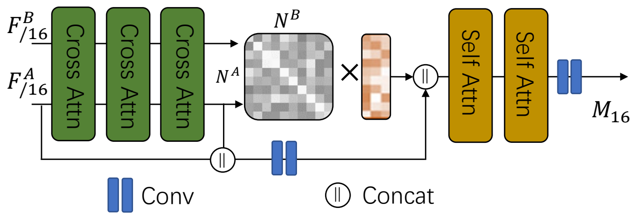

Figure 2. **Global matcher**. × denotes matrix multiplication.

1.2. Global Matcher

As depicted in Fig. 2, given two input feature sets F^A and F^B corresponding to images I^A and I^B, we first ap-

*Corresponding author.

Table 1. **Comparison on computational cost** at a resolution of 672×672. Note that, the test pipeline is different from the one used camera in pose estimation.

Consumption	ArgMatch	RoMa	DKM
Feature Extractor			
Time (ms)	135	540	136
Mem. (G)	0.61	3.15	0.62
#Para (M)	7.48	315	25.6
Global Matcher			
Time (ms)	24	51	41
#Para (M)	12.6	67.2	12.4

ply cross-attention [6, 7, 9] to enable inter-image interaction and enhance the similarity between corresponding features. We then compute the global correlation volume $P \in \mathbb{R}^{N^A \times N^B}$ based on the enhanced features, where N^A and N^B denote the number of pixels at 1/16 resolution in I^A and I^B, respectively.

Meanwhile, we map the grid coordinates of image I^B into a positional embedding $PE \in \mathbb{R}^{N^B \times C}$. The final match embedding is then obtained as:

$$M^E = softmax_{(\cdot,:)}(P)PE \tag{1}$$

Then the original feature F^A, the enhanced F'^A, and the embedding of matches are concatenated and compressed. A self-attention module is subsequently applied to refine the global matches. Finally, the global match embeddings are decoded using lightweight convolutions.

1.3. Computational Cost of Feature Extractor and Global Matcher

As shown in Table 1, our feature extractor and global matcher are most efficient. Specifically, RoMa [3] incorporates a fundamental model, DINOv2 [1], as a part of feature extractor significantly increasing computational burdens. Additionally, while DKM's feature extractor, based on ResNet50 [4], appears efficient, it generates up to 512-dimensional features for refinement, further adding to the computational load. For the global matcher, both DKM and RoMa utilize Gaussian-Process-based regressor as the core of their decoders, resulting in a time complexity of $\mathcal{O}(n^3)$. In contrast, our approach achieves a time complexity of $\mathcal{O}(n^2)$, making it significantly more efficient, especially at higher resolutions.

The strong performance of ArgMatch shown in the paper demonstrates that computational burdens caused feature extractor and global matcher can be effectively reduced, when the refinement pipeline properly guide their training and fully exploit the information they provide.

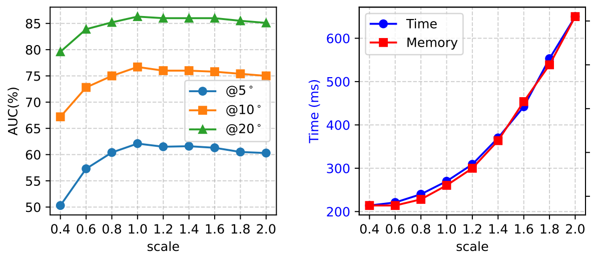

Figure 3. **Impact of resolution on MegaDepth**.

2. Details of Training

Our training scheme consists of two stages: warm-up and end-to-end training. During the warm-up stage, the refinement pipeline is excluded, and the learning rate is set to 2e-4. This stage concludes when 90% of points achieve an error below 1 pixel. The end-to-end training stage runs for 1,500,000 steps. The learning rate is again initialized at 2e-4 and is exponentially decayed to 1e-5 after the first third of total training steps. A weight decay of 0.01 is applied, and the gradient norm is clipped to a maximum of 0.01 in both stages. We also accumulate gradients over 4 steps to stabilize optimization. Additionally, we propose mixed-resolution training, where one step at 640 × 480 and one step at 672 × 672 are inserted every 20 training steps to enhance robustness across varying input scales.

3. More Experimental Results

3.1. Impact of Resolution

Image resolution significantly affects both matching accuracy and computational efficiency, requiring a trade-off between the two. As shown in Fig. 3, when scaling the default resolution of 608×800, performance saturates beyond a scale factor of 0.8. Therefore, we set the default test resolution to 608×800, matching the training resolution. While this choice ensures consistency, it may also indicate mild overfitting.

3.2. More Visualization

Additional visualization results are presented in Fig. 4. While our method effectively handles challenging cases such as repetitive patterns and textureless regions, it exhibits limitations in wide-baseline scenarios, as illustrated in the last row. This shortcoming may arise from the relatively weak global matching component, which we leave as a potential direction for future work.

References

[1] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerg-

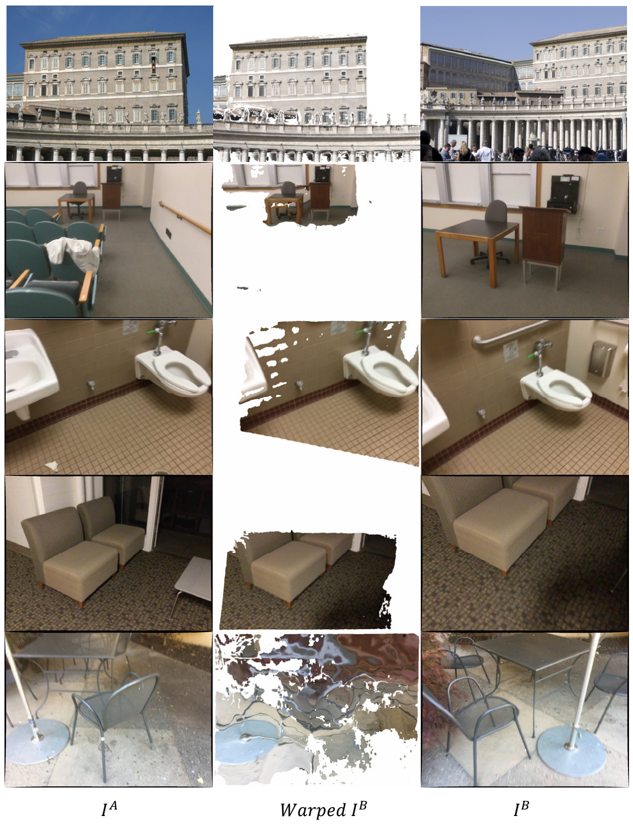

Figure 4. **More visualization for image registration**.

ing properties in self-supervised vision transformers. In *Proc. Int. Conf. Comput. Vis.*, 2021. 2

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 1251–1258, 2017. 1

[3] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Robust dense feature matching. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 19790–19800, 2024. 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 770–778, 2016. 2

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. Med. Image Comput. Comput. Assist. Interv.*, pages 234–241, 2015. 1

[6] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 4938–4947, 2020. 2

[7] A Vaswani. Attention is all you need. *Adv. Neural Inf. Process. Syst.*, 2017. 2

[8] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proc. Europ. Conf. Comput. Vis.*, pages 3–19, 2018. 1

[9] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pages 8121–8130, 2022. 2

[10] F Yu. Multi-scale context aggregation by dilated convolutions. *Proc. Int. Conf. Learn. Represent*, 2016. 1