CroCo v2: Improved Cross-view Completion Pre-training for Stereo Matching and Optical Flow – Supplementary Material –

Philippe Weinzaepfel Thomas Lucas Vincent Leroy
Yohann Cabon Vaibhav Arora Romain Brégier Gabriela Csurka
Leonid Antsfeld Boris Chidlovskii Jérôme Revaud
NAVER LABS Europe
https://github.com/naver/croco

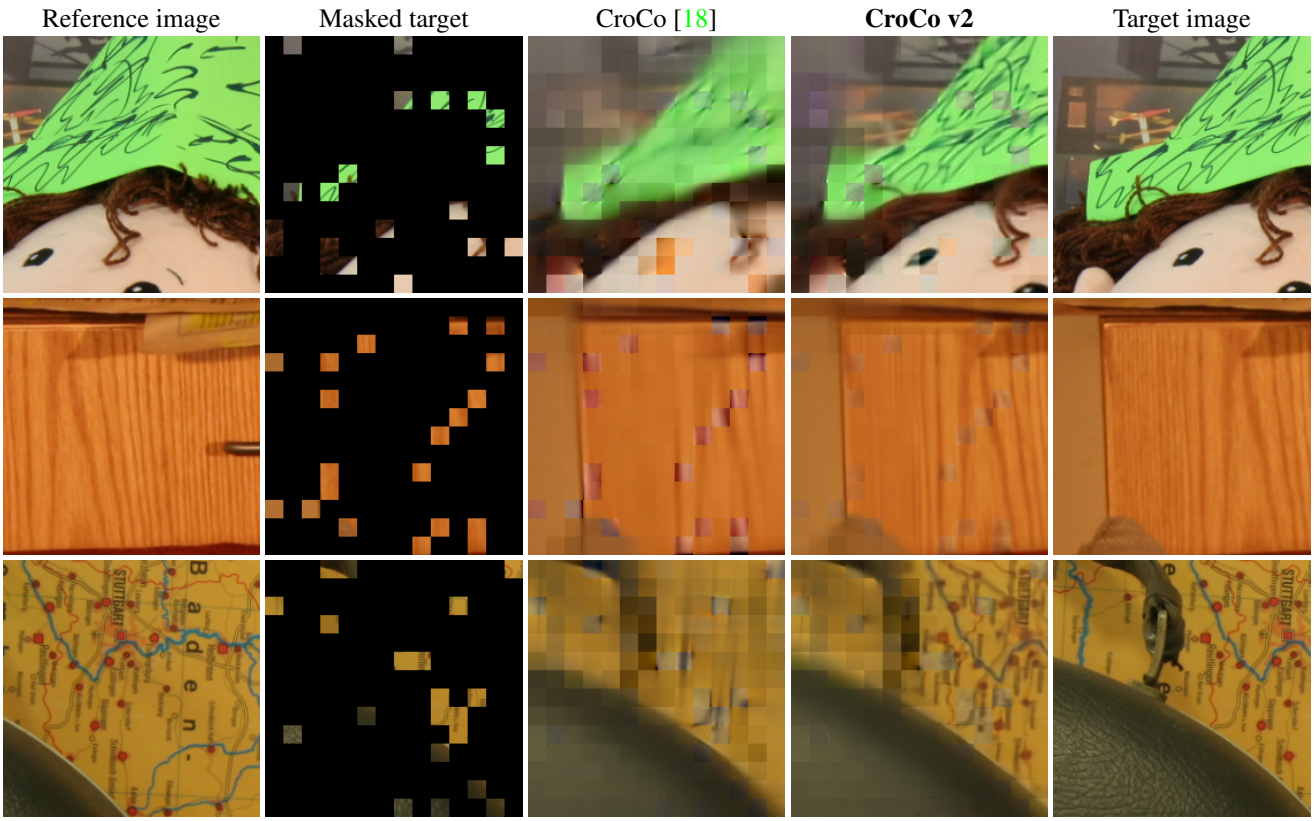

Figure 1: **Cross-view reconstruction examples** (pre-training pretext task) on scenes unseen during pre-training for the original CroCo [18] and with our improvements. The images come from the Middlebury stereo benchmark [14].

In this supplementary material, we first provide visualizations of the capabilities of CroCo v2 on the pretext task of cross-view completion (Section 1). We then present additional experimental results in Section 2, including in particular (a) the impact of pre-training, (b) the runtime of our model and (c) an analysis of the probabilistic distributions regressed by our CroCo-Stereo model for the stereo matching task. We finally detail our training setup and the dataset splits. (Section 3).

1. Cross-view completion examples

To qualitatively evaluate the impact of CroCo v2, *i.e.*, of the improvements that we propose on top of the CroCo [18] pre-training, we show several examples of cross-view completions on real-world scenes, coming either from Middlebury v3 [14] in Figure 1 or KITTI [11] in Figure 2. Note that these methods, as MAE [4], regress pixel values that are normalized according to the mean and standard deviation

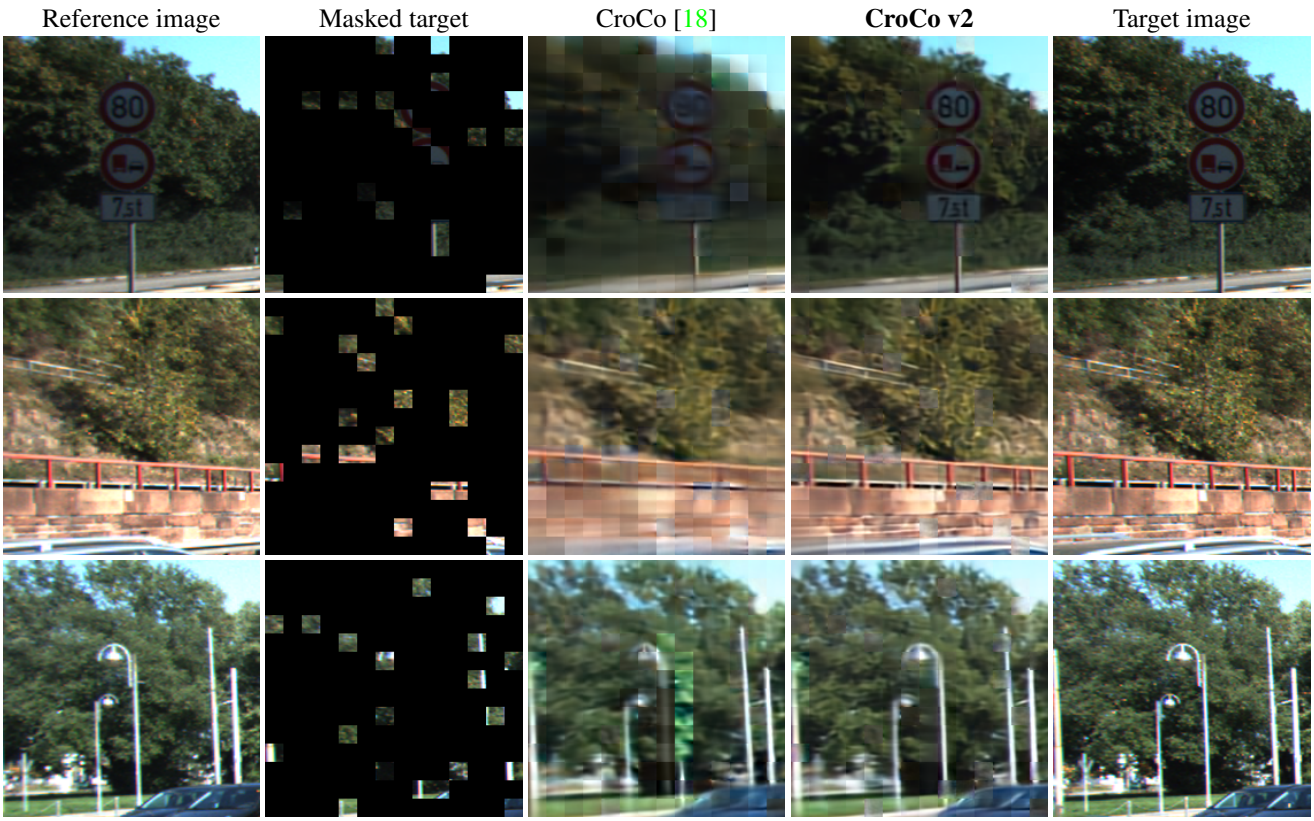

Figure 2: **Cross-view reconstruction examples** (pre-training pretext task) on scenes unseen during pre-training for the original CroCo [18] and with our improvements. The images come from the stereo benchmark of KITTI [3].

inside each patch, we thus apply the inverse transform for display: this means that the overall color of each patch will be correct, as it comes from the ground-truth values. While the most important measure of performance of these models is their transfer to downstream tasks, as explored in the main paper, a qualitative observation of the fact that our improved method is better at solving the pretext task is noteworthy. We clearly observe that the reconstructions from the original CroCo [18] tend to be quite blurry in many areas, which might come from the fact that it relies on a smaller model and was pre-trained only on synthetic data from indoor environments, while details are impressively preserved thanks to our improvements. In Figure 1, note how the lines and the eyes are well reconstructed in the first row, or the roads on the maps of the third row, despite the high masking ratio that is applied to the masked image (90%). Similarly, the text is clearly readable on the first row of Figure 2. Some predictions by our model have some blur (*e.g.* left of the first and thirds rows of Figure 1), which makes sense because these parts are not visible in the reference image.

2. Further experimental results

2.1. Impact of pre-training

In Table 1, we measure the impact of the pre-training on the downstream performance when the model is finetuned for stereo matching or optical flow. The first two rows compare our model, using our improved cross-view completion pre-training *vs*. a random initialization. We observe a clear gain of performance, *e.g.* on the FlyingThings flow test set in the final rendering with an EPE of 2.45 pixels with pre-training *vs*. 10.57 without it, or on the Middlebury v3 stereo validation set with a bad@1.0px of 15.5% with pre-training *vs*. 43.4% without it.

We are not aware of any other pre-training strategy, other than cross-view completion, that readily includes a binocular decoder or architecture. While it is still possible to initialize part of the layers using other pre-training strategies, this means that some important parts of the network are still being initialized at random. Nevertheless, to compare to other pre-training strategies, we consider MAE [4] pre-trained on ImageNet [13], thus with a cosine positional embedding, a ViT-Base encoder, and with a Small decoder that is randomly initialized. We compare that to the original CroCo [18] pre-trained on synthetic data only. We observe

Network initialization	Stereo (bad@1.0px↓)				Flow (EPE↓)			
	Md	ETH	SF(c)	SF(f)	FT(c)	FT(f)	Si.(c)	Si.(f)
RoPE positional embedding, ViT-L encoder, Base decoder, 2M Habitat + 5.3M real pre-training pairs								
CroCo v2 pre-training	**15.5**	**0.38**	**5.0**	**5.3**	**2.85**	**2.45**	**1.43**	**1.99**
random init.	43.4	1.06	11.0	11.2	10.53	10.57	4.84	5.49
cosine positional embedding, ViT-B encoder, Small decoder, 2M Habitat (synthetic only) pre-training pairs								
CroCo [18] pre-training	**26.3**	1.82	**6.7**	**7.0**	**3.89**	**3.56**	**2.07**	**2.57**
MAE [4] (ImageNet) pre-training (encoder only)	35.8	**1.68**	8.6	8.8	5.13	4.83	2.92	3.82
random init.	87.5	5.42	24.6	24.6	14.28	14.31	8.99	9.76

Table 1: **Impact of pre-training.** We compare the performance of our final model (first row) with improved cross-view completion pre-training to a randomly initialized version (second row). To compare to MAE [4], that is pre-trained on ImageNet [13], and which is based on cosine positional embeddings, we make the comparison with the original CroCo in the bottom rows.

Masking ratio	Stereo (bad@1.0px↓)				Flow (EPE↓)			
	Md	ETH	SF(c)	SF(f)	FT(c)	FT(f)	Si.(c)	Si.(f)
80%	32.5	1.96	7.3	7.5	4.29	4.06	2.06	2.71
85%	59.2	1.15	8.7	9.0	3.48	3.08	1.99	2.41
90%	**20.7**	**0.82**	**5.8**	**6.1**	**3.35**	**2.94**	**1.76**	**2.30**

Table 2: **Impact of the pre-training masking ratio** for a model with RoPE positional embeddings, a ViT-B encoder, a Small decoder, pre-trained on 2M Habitat + 5.3M real pairs.

that CroCo pre-training obtains the lowest errors, significantly outperforming the MAE pre-training and the random initialization.

Interestingly, the performance of this smaller model is also significantly better than the large one without pre-training. This again highlights the importance of the pre-training with such generic architecture.

Masking ratio. CroCo [18] finds that using a masking ratio of 90% performs best for cross-view completion on their synthetic data. This is higher than the 75% masking ratio of MAE [4], as the unmasked reference view of the same scene adds redundancy. A question is whether this masking ratio of 90% that has been found optimal on synthetic data generalizes to real data. Table 2 reports the performance on stereo and flow downstream tasks for a masking ratio of 80%, 85% and 90%. We find that a masking ratio of 90% performs best also in the case of using real data.

2.2. Smaller training data

Most optical flow methods also report the performance on the MPI-Sintel training set when training on FlyingChairs and FlyingThings only. We report these values in Table 3. For RAFT [19] and GMFlow [20], we report the numbers before and after using iterative refinement procedures. Interestingly, CroCo-Flow performs better than these two methods before their refinement. Overall, our ranking is similar to the ones on the MPI-Sintel test set where we use more training data. This indicates that our finetuning on geometric downstream tasks do not necessarily need large-scale training data, despite the size of our architecture.

Method	MPI-Sintel(↓)	
	clean	final
LiteFlowNet2 [6]	2.24	3.78
FM-RAFT [7]	1.29	2.95
FlowFormer [5]	**1.01**	**2.40**
RAFT [16] before refinement	4.04	5.45
RAFT [16]	1.41	2.69
GMFlow [20] before refinement	1.31	2.96
GMFlow [20]	<u>1.08</u>	<u>2.48</u>
CroCo-Flow	1.28	2.58

Table 3: **Optical flow results when training on FlyingChairs and FlyingThings only.** We report the EPE on MPI-Sintel training set (clean or final rendering). Numbers for the first three rows come from [5], numbers for RAFT and GMFlow (before and after refinement) from [20].

2.3. Runtime and tiling

Runtime. In Table 4, we report the runtime for different sizes of our model. On one single tile of the same size as training for stereo, *i.e.*, 704×352, on a NVIDIA A100 GPU. Our method remains relatively fast on one tile, in the order of a few tens of milliseconds.

Number of parameters. We also report the number of

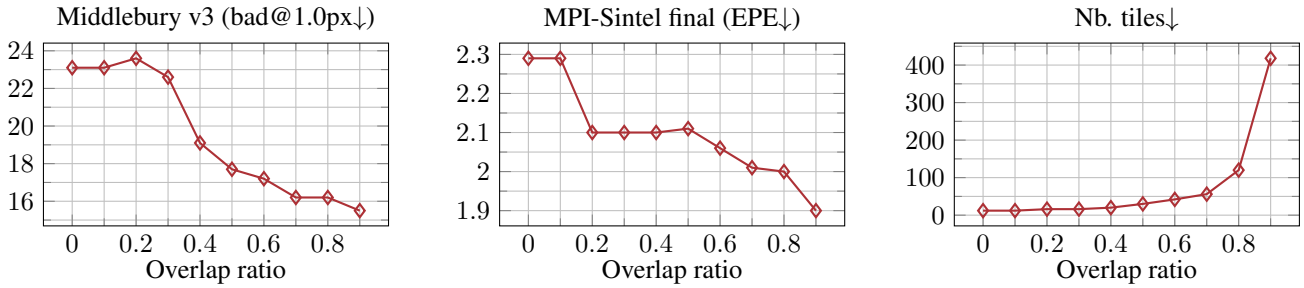

Figure 3: **Impact of the overlap ratio between tiles during inference.** We plot the stereo performance (bad@1.0px in %) on Middlebury v3 (left) and the flow performance on MPI-Sintel in its final rendering (middle) when varying the overlap ratio during inference with tiling. We also plot the number of tiles it represents for a 1920×1080 image (right), which is proportional to the total runtime, for a crop size of 704×352 as CroCo-Stereo.

Pos.	Encoder	Decoder	runtime	#Parameters
cosine	ViT-B	Small	25ms	139.4M (85.6M+34.0M+19.7M)
RoPE	ViT-B	Small	26ms	139.4M (85.6M+34.0M+19.7M)
RoPE	ViT-B	Base	29ms	219.7M (85.6M+114.0M+20.1M)
RoPE	ViT-L	Base	53ms	437.4M (303.1M+114.2M+20.1M)

Table 4: **Runtime and number of parameters.** Runtime is measured for a single tile of size 704×352, on a NVIDIA A100 GPU. For the number of parameters we report in parenthesis the numbers for the encoder, the decoder and the DPT head separately.

trainable parameters in Table 4. This number of parameters is one order of magnitude higher than most existing stereo and flow methods. We did not study how this number of parameters could be reduced, and we also do not claim that our models are better than existing work for a fixed computational budget. Indeed, task-specific approaches have the advantage of being more sample efficient, *i.e.*, requiring less data, by leveraging prior knowledge about the task. They also have the drawback of not being readily compatible with large-scale training on unlabeled data, because of task-dependent components, which limits the use of large generic models. Existing methods cannot be scale up to a larger number of parameters easily, as training large models requires lot of data. In the case of stereo and optical flow, for which labeled data is limited, this means using self-supervised learning, which cannot be straightforwardly applied for models that involve task-specific designs like cost volumes, image warping, *etc*. Thus, our contribution and our aim in this work is to show that pre-training large, generic architectures and finetuning them for stereo matching and optical flow is a valid path forward.

Impact of the overlap ratio during tiling. In Figure 3, we report the performance and the number of tiles for a Full HD image (1920×1080) when varying the overlap ratio during inference with the tiling strategy. While the performance improves with a higher overlap ratio, the number of tiles can rapidly explodes. With an overlap around 0.5 or 0.7, performance is quite close to the one obtained with 0.9 while the number of tiles remains reasonable. This may thus be the best trade-off in practical scenarios where inference time has to stay small.

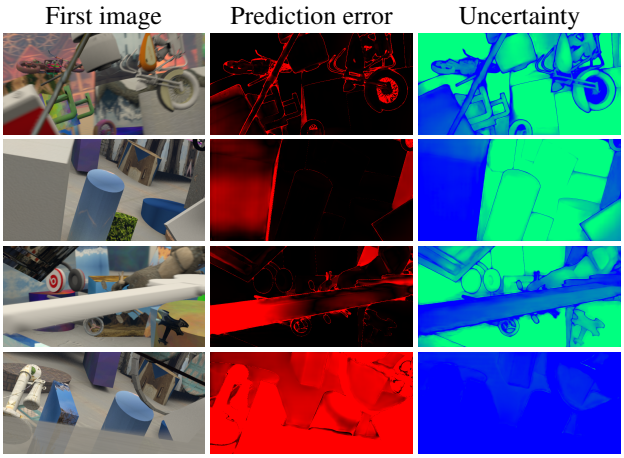

Figure 4: **Visualization of the uncertainty predicted by CroCo-Stereo** on a few examples from the SceneFlow test set. The first column shows the first image, the second column shows the error of the prediction clamped within the segment $[0, 10]$, the third column shows the logarithm of the predicted scale of the Laplacian distribution output by the model: green colors denote confident areas while blue colors denote uncertain areas.

2.4. Laplacian-based loss

For flow and stereo, we regress a Laplacian distribution: the location parameter corresponds to the disparity or flow prediction, while the scale parameter could be seen as a measure of uncertainty. We thus denote here by 'uncertainty' the logarithm of the predicted scale of the Laplacian distribution that our downstream model outputs, *i.e.*, $log(d_i)$ from Equation 3 of the main paper.

Visualization of the uncertainty. We visualize in Figure 4

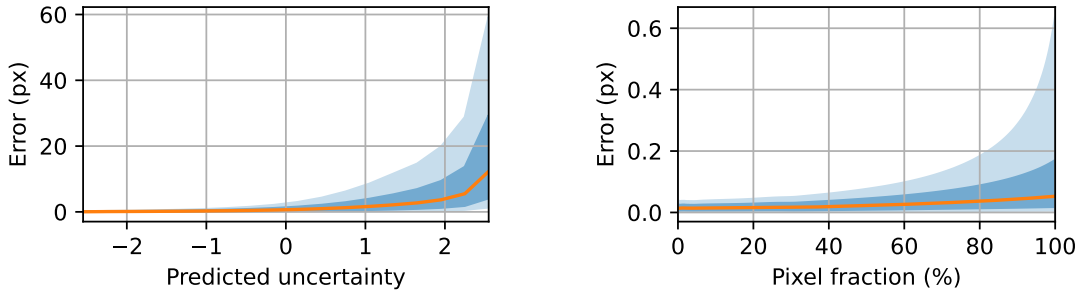

Figure 5: **Statistics on the uncertainty predicted by CroCo-Stereo.** We subsample 1000 points per test image from SceneFlow in its clean renderings and compute the error of the prediction and the logarithm of the predicted scale of the Laplacian, *i.e.*, the pixelwise uncertainty. On the left plot, we show the median of the error for a given predicted uncertainty (orange line), the 25- and 75-percentile in dark blue, and the 10- and 90-percentile in light blue. On the right plot, we sort pixels according to their predicted uncertainty from the less uncertain to the more uncertain and show the median of the error over the fractions of pixels considered (orange line), the 25- and 75-percentile in dark blue, and the 10- and 90-percentile in light blue.

loss	Stereo (bad@1.0px↓)				Flow (EPE↓)			
	Md	ETH	SF(c)	SF(f)	FT(c)	FT(f)	Si.(c)	Si.(f)
L1	23.0	0.95	6.1	6.3	3.02	2.69	1.51	2.13
Lap.	**15.5**	**0.38**	**5.0**	**5.3**	**2.85**	**2.45**	**1.43**	**1.99**

Table 5: **Impact of the loss.** We compare a standard L1 loss *vs*. the Laplacian (Lap.) loss.

this uncertainty for a few examples. We observe that it is highly linked with the error of the predicted disparity as red areas in the error correspond to blue areas in the uncertainty maps.

Statistics on the uncertainty. To better measure the correlation of our predicted uncertainty with the error of the disparity prediction, we plot a few statistics in Figure 5. On the left one, we show some percentiles of the error when varying the predicted scale of the Laplacian distribution. We observe that a lower uncertainty clearly corresponds to pixels with lowest errors, while a high uncertainty corresponds to pixels with a higher error. On the right plot, we order pixels from the less uncertain to the more uncertain and show the percentiles of errors when increasing the ratio of pixels considered. We observe the same behavior, showing the correlation of our uncertainty with the error of the prediction. Note that 95% of the pixels have an error below 1, thus the scale of the y-axis of the plot.

Comparison with an L1 loss. In Table 5, we quantitatively evaluate the effect of using a loss on a Laplacian distribution (Equation 3 of the main paper) compared to using only an L1 loss. In the latter case, we cannot leverage the predicted scale of a Laplacian distribution for merging overlapping tiles. We thus follow [5] and use a weights that decrease with the distance to the center of the image. We observe that the Laplacian loss outperforms the L1 loss on all stereo and flow benchmarks. A Laplacian loss can be interpreted as an L1 term, weighted for each pixel according to an uncertainty measure, thus allowing to downweight uncertain pixels in practice. In addition, having access to the scale of the Laplacian allows a more elegant merging strategy for the overlapping tiles.

2.5. Towards smarter tiling

One limitation of our approach mentioned in the main paper is the tiling-based inference. For instance, CroCo-Stereo is based on crops with a width of 704 pixels, this means that for large disparity values, the matching pixels would be out of the scope of the corresponding tile in the second image. As an alternative, we have tried a strategy where a second tile in the second image is also considered, which is shifted by 150 pixels, thus reducing the disparity value by the same amount. With the model with ViT-Base encoder and Base decoder, such a strategy allows to reduce the bad@1.0 from 17.1% to 12.0% on Middlebury v3 validation set, when replacing the predictions over 200px from the original tile, with the ones from the secondary tile. While this strategy seems promising, it is however not really satisfactory as it multiples the number of tiles to proceed by 2. We hope to find better strategies in the future.

3. Training details

CroCo-Stereo training. We train CroCo-Stereo for 32 epochs using batches of 6 pairs of 704×352 crops. We detail the training/validation pairs we use for our ablations in Table 6. We use the AdamW optimizer [9] with a weight decay of 0.05, a cosine learning rate schedule with a single warm-up epoch and a learning rate of 3.10^{-5}. During training, we apply standard data augmentations: color jittering (asymmetrically with probably 0.2), random vertical flipping with probably 0.1, random scaling with probability

stereo dataset		# pairs	comment
CREStereo [8]		200,000	all training pairs
SceneFlow[10]		70,908	Driving, Monkaa and FlyingThings in both clean and final renderings 4,370 validation pairs from FlyingThings test for each rendering (clean and final)
ETH3D Low Res [15]	30×	24	'delivery_area_3s', 'electro_3l' 'playground_3l' (3 pairs) are kept apart for validation
Middlebury v3 [14]	50×	14	'Vintge' (1 pair) is kept apart for validation, we use the 'full' resolution
Middlebury 21	50×	335	'traproom1' and 'traproom2' are kept apart for validation (20 pairs)
Middlebury 14	50×	132	'Umbrella-umperfect' and 'Vintage-perfect' are kept apart for validation (6 pairs)
Middlebury 06	50×	171	'Rocks1' and 'Wood2' are kept apart for validation (18 pairs)
Middlebury 05	50×	45	'Reindeer' is kept apart for validation (9 pairs)
Booster [12]		213	only the 'balanced' subset, 'Vodka' and 'Washer' sequences (15 pairs) kept apart for validation
total		306,691	

Table 6: **Overview of our stereo training data.** We indicate here the train/val split used for the ablations, as well as the number of training pairs. For ETH3D and Middlebury, we also consider multiple times each pair in each epoch.

flow dataset	# pairs	prob.	comment
FlyingChairs [2]	22,232	12%	-
FlyingThings [10]	80,604	40%	40,302 pairs for both 'clean' and 'final' renderings we use the same 1,024 validation pairs from the test set as [20]
MPI-Sintel [1]	943	10%	sequences 'temple_2' and 'temple_3' (98 pairs) are kept apart for validation
TartanAir [17]	306,268	38%	-
total	410,047	100%	

Table 7: **Overview of our flow training data.** We indicate here the train/val split used for the ablations, as well as the number of remaining training pairs. During training, we set a number of images per epoch and randomly sample them among the available datasets with the percentages shown in the column 'prob.'.

0.8 in the range $[2^{-0.2}, 2^{0.4}]$ and stretching (resize different along the x and y axis) with probability 0.8 in the range $[2^{-0.2}, 2^{0.2}]$, and slightly jitter the right image with probability 0.5. When submitting to the official leaderboards, we include the pairs that were kept apart from the training sets for validation into the training epochs.

CroCo-Flow training. We train CroCo-Flow for 240 epochs of $30,000$ pairs each, randomly sampled from all available data, using batches of 8 pairs of crops of size 384×320. We detail the training/validation pairs we use for our ablations in Table 7. To better balance the datasets, we set the probability of choosing a random pair from these datasets, see Table 6. We use the AdamW optimizer, a weight decay of 0.05, a cosine learning rate schedule with linear warm-up over 1 epoch, and a base learning rate of 2.10^{-5}. During training, we apply standard augmentations [20]: random color jittering (asymmetrically with probably 0.2), random scaling with probably 0.8 with a scale sampled in $[2^{-0.2}, 2^{0.5}]$ and stretching with probability 0.8 in the range $[2^{-0.2}, 2^{0.2}]$.

References

[1] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 6

[2] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 6

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 2

[4] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked Autoencoders are Scalable Vision Learners. In *CVPR*, 2022. 1, 2, 3

[5] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. FlowFormer: A transformer architecture for optical flow. In *ECCV*, 2022. 3, 5

[6] Shihao Jiang, Yao Lu, Hongdong Li, and Richard Hartley. Learning optical flow from a few matches. In *CVPR*, 2021. 3

[7] Shihao Jiang, Yao Lu, Hongdong Li, and Richard Hartley. Learning optical flow from a few matches. In *CVPR*, 2021. 3

[8] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *CVPR*, 2022. 6

[9] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 5

[10] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 6

[11] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 1

[12] Pierluigi Zama Ramirez, Fabio Tosi, Matteo Poggi, Samuele Salti, Stefano Mattoccia, and Luigi Di Stefano. Open challenges in deep stereo: the booster dataset. In *CVPR*, 2022. 6

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 2, 3

[14] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *GCPR*, 2014. 1, 6

[15] Thomas Schops, Johannes L Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017. 6

[16] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020. 3

[17] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *IROS*, 2020. 6

[18] Weinzaepfel, Philippe and Leroy, Vincent and Lucas, Thomas and Brégier, Romain and Cabon, Yohann and Arora, Vaibhav and Antsfeld, Leonid and Chidlovskii, Boris and Csurka, Gabriela and Revaud Jérôme. CroCo: Self-Supervised Pre-training for 3D Vision Tasks by Cross-View Completion. In *NeurIPS*, 2022. 1, 2, 3

[19] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *CVPR*, 2022. 3

[20] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Trans. PAMI*, 2023. 3, 6