CroCo: Self-Supervised Pre-training for 3D Vision Tasks by Cross-View Completion

Philippe Weinzaepfel **Vincent Leroy** **Thomas Lucas**

Romain Brégier **Yohann Cabon** **Vaibhav Arora** **Leonid Antsfeld**

Boris Chidlovskii **Gabriela Csurka** **Jérôme Revaud**

NAVER LABS Europe
https://europe.naverlabs.com/research/computer-vision/croco/

Abstract

Masked Image Modeling (MIM) has recently been established as a potent pre-training paradigm. A pretext task is constructed by masking patches in an input image, and this masked content is then predicted by a neural network using visible patches as sole input. This pre-training leads to state-of-the-art performance when finetuned for high-level semantic tasks, *e.g.* image classification and object detection. In this paper we instead seek to learn representations that transfer well to a wide variety of 3D vision and lower-level geometric downstream tasks, such as depth prediction or optical flow estimation. Inspired by MIM, we propose an unsupervised representation learning task trained from *pairs* of images showing the same scene from different viewpoints. More precisely, we propose the pretext task of *cross-view completion* where the first input image is partially masked, and this masked content has to be reconstructed from the visible content and the second image. In single-view MIM, the masked content often cannot be inferred precisely from the visible portion only, so the model learns to act as a prior influenced by high-level semantics. In contrast, this ambiguity can be resolved with cross-view completion from the second unmasked image, on the condition that the model is able to understand the spatial relationship between the two images. Our experiments show that our pretext task leads to significantly improved performance for monocular 3D vision downstream tasks such as depth estimation. In addition, our model can be directly applied to binocular downstream tasks like optical flow or relative camera pose estimation, for which we obtain competitive results without bells and whistles, *i.e.*, using a generic architecture without any task-specific design.

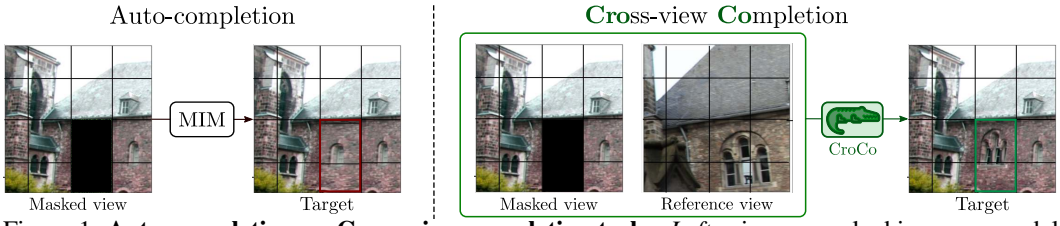


Figure 1: **Auto-completion *vs*. Cross-view completion tasks.** *Left:* given a masked image, a model trained for auto-completion can only leverage the visible context to fill-in the blanks and thus relies mainly on high-level semantic information. *Right:* given an additional view of the same scene, cross-view completion makes precise reconstruction possible, assuming that the model is able to understand both the scene geometry and the spatial relationship between the two images.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

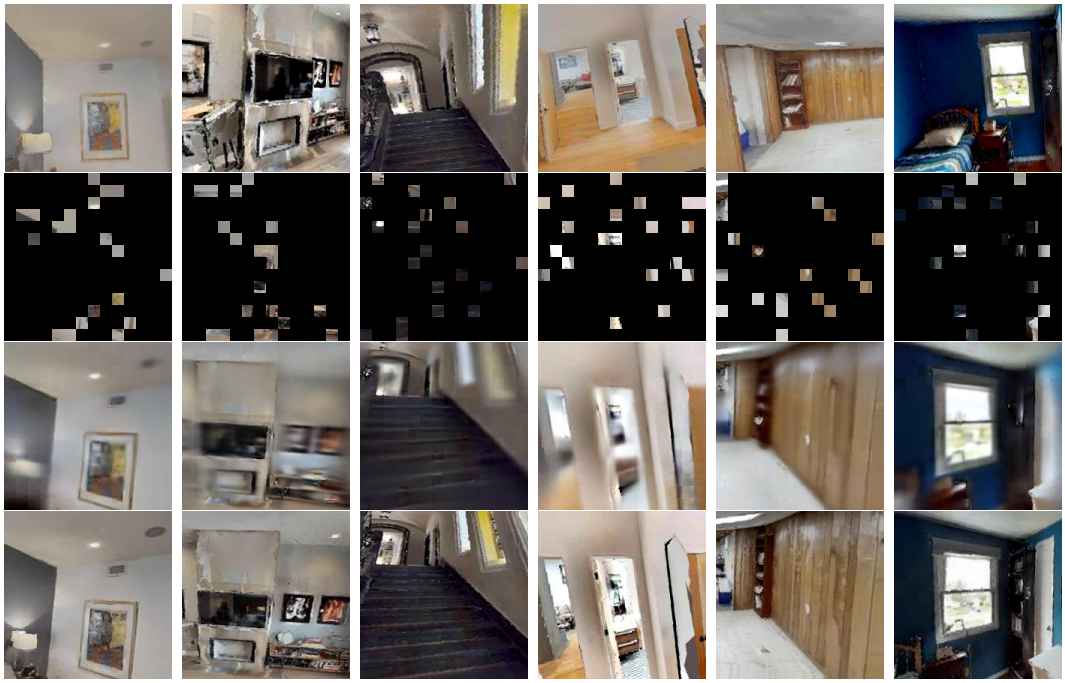


Figure 2: **Reconstruction examples from CroCo** on scenes unseen during training. From top to bottom: reference image (input), masked image (input), CroCo output, original target image.

1 Introduction

Self-supervised learning for model pre-training has allowed to achieve state-of-the-art performance in a number of high-level computer vision tasks, such as image classification or object detection. Contrary to traditional supervised learning, these models enable the use of unlabelled data via carefully designed pretext tasks. A popular method for self-supervision is instance discrimination [12, 13, 17, 28, 37, 39, 83] which constructs a pretext task by learning representations that are invariant to various data augmentations. More recently, Masked Image Modeling (MIM) [7, 16, 19, 32, 38, 60, 93] has emerged as a powerful alternative for self-supervision. Inspired by BERT [24], these models are trained using an auto-completion pretext task. An encoder, usually a Vision Transformer (ViT) [26], takes a partial view of an image input, obtained by splitting the image into patches and masking some of them, and encodes it into a latent representation. The masked patches are then predicted by a decoder using the latent representation. To solve the task, the model must leverage the context given by the visible portion and act as a prior for the ambiguous content that cannot be deduced from them. In practice these models are typically trained on object-centric datasets such as ImageNet [67] and thus tend to learn high-level semantic information; that makes them well suited for tasks such as image classification or object detection [4, 38, 47].

In this paper, we propose a self-supervised training objective specially designed to learn 3D geometry from unlabeled data, named *Cross-view Completion*, or *CroCo* in short. Given two images depicting the same scene, random parts of the first input image are masked and then predicted by the model using both (1) the visible parts of this first image, as well as (2) a second image called *reference* image as it depicts the same scene from a different point of view, see Figure 1. While multi-view image completion has a long history in image editing [22, 92], we are the first to explore its potential as a self-supervised representation learning tool. In contrast to single-view completion, the proposed pretext task of cross-view completion allows to perform masked image modeling conditionally on a second view. In this case, most of the ambiguity can be resolved by reasoning about the scene geometry and spatial relationship between the two views. This is illustrated in the reconstruction examples of our model in Figure 2.

Figure 3 provides an overview of our self-supervised model during pre-training. We divide both images into sets of non-overlapping patches, denoted as tokens. Most tokens from the first image are randomly discarded, and the remaining ones are fed to an image encoder, which we implement

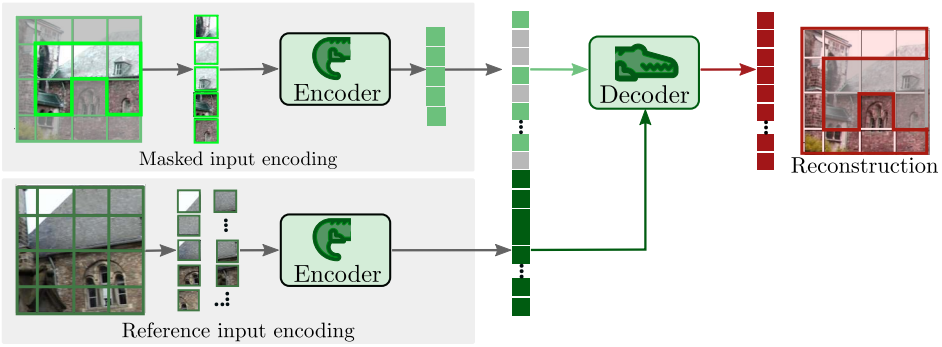


Figure 3: **Overview of our CroCo model during pre-training.** Patches from the first input image are partially masked; visible ones are encoded into their latent representations, then padded with masked tokens to account for hidden patches. The same encoder is used to encode the patches of the second image. The decoder receives the encoded tokens from both images and use them to reconstruct the masked parts of the first image.

using a Vision Transformer (ViT) backbone [26]. All patches from the second (reference) image are encoded in a *Siamese* manner, *i.e.*, using the same encoder with shared weights. The token latent representations output by the encoder from both images, including tokens to account for the masked patches of the first image, are then fed to a decoder whose goal is to predict the appearance of hidden patches. To this aim, we use a series of transformer decoder blocks comprising cross-attention layers. This allows non-masked tokens from the first image to attend tokens from the reference image, thus enabling cross-view comparison and reasoning. Our model is trained using a simple pixel reconstruction loss over all masked patches, similar to MAE [38]. To finetune the model on downstream tasks that process only a single image, *e.g.* monocular depth estimation, the decoder can be discarded and we use the pre-trained encoder alone. In the case of binocular tasks such as optical flow estimation, the original CroCo architecture is used as is.

For pre-training, CroCo relies on pairs of images depicting the same scene; in this work the pairs were obtained from synthetic renderings of indoor scenes produced with the Habitat [68] simulator. We empirically show that high masking ratios, *e.g.* 90%, lead to the best pre-training performance. Our model is evaluated on monocular downstream tasks, in particular depth estimation on the NYUv2 dataset [70] and a diverse set of dense 2D and 3D regression tasks taken from Taskonomy [91]. We show that CroCo leads to significantly better performance compared to existing MIM models, whether they were pre-trained on the same data or on ImageNet [67]. When evaluated on monocular high-level semantic tasks, such as ImageNet classification, CroCo obtains lower performance compared to established MIM models, however. This essentially comes from our use of indoor scenes for pre-training, instead of highly semantic datasets like ImageNet, as empirically shown in our experiments. Lastly, we demonstrate that CroCo can be applied to binocular downstream tasks in a straightforward manner without bells and whistles. For instance, optical flow estimation can be performed by directly regressing 2 values per pixel using the decoder, and likewise, relative pose regression is achieved by simply appending a pose regression head to CroCo. In both cases, we show that CroCo pre-training leads to competitive results in these 3D vision downstream tasks.

2 Related work

Self-supervised representation learning is a paradigm developed to learn visual features from large-scale sets of unlabeled data [44], before finetuning the model on downstream tasks, *e.g.* classification or detection. To achieve that, a *pretext* task is designed and exploits inherent data attributes to automatically generate surrogate labels. The earliest principle behind most existing pretext tasks for computer vision revolves around purposefully removing *some* information from an image, *e.g.* the color, the orientation or the ordering of a sequence of patches obtained from the image that the model then learns to recover [28, 34, 57, 58]. It has been shown that despite the lack of any semantic supervision, networks trained to recover this artificially removed information learn useful representations and facilitate the training of various supervised downstream tasks [1, 31].

Recently, the paradigm of instance discrimination has received a lot of attention, achieving highly competitive results in self-supervised visual representation learning [2, 13, 14, 17, 37, 39]. It seeks to learn outputs that are invariant to well designed classes of data augmentation. Positive image pairs, obtained from the same instance by data augmentation, are pulled closer by the model, while samples obtained from different instances are pushed apart in the embedding space.

More recently, motivated by the success of BERT [24] in NLP and by the introduction of Vision Transformers (ViT) [26], a variety of masked image prediction methods for self-supervised pre-training of vision models has been proposed. Reminiscent of denoising autoencoders [78] or context encoders [60], and aiming to reconstruct masked pixels [3, 16, 26, 30, 38, 85], discrete tokens [7, 95] or deep features [5, 82], these methods have demonstrated the ability to scale to large datasets and models and achieve state-of-the-art results on various downstream tasks. In particular, the masked autoencoder (MAE) [38] accelerates pre-training by using an asymmetric architecture that consists of a large encoder that operates only on unmasked patches followed by a lightweight decoder that reconstructs the masked patches from the latent representation and mask tokens. MultiMAE [4] leverages the efficiency of the MAE approach and extends it to multi-modal and multitask settings. Rather than masking input tokens randomly, the Masked Self-Supervised Transformer model (MST) [48] proposes to rely on the attention maps produced by a teacher network to dynamically mask low response regions of the input, and a student network is then trained to reconstruct it.

Self-supervised pre-training for dense downstream tasks. Seminal self-supervised methods [18, 37] were designed to output global image representations, thus encoding limited local information, and thereby hindering transferability of the learned models to downstream tasks involving dense per-pixel predictions, such as semantic segmentation. To alleviate this issue, several self-supervised methods have been proposed to perform contrastive learning on dense local representations rather than global ones [51, 81, 84]. In contrast, InsLoc [87] proposes to paste image instances at various locations and scales onto background images, then predicts instance categories and foreground bounding boxes in the composed images. In VADeR [61] and FlowE [86], dense image representations are learned for several dense semantic downstream tasks such as object detection and semantic segmentation, using multi-hierarchy features [80], while [90] combines multi-resolution parallel modules with local-window self-attention to improve the memory and computation efficiency of dense prediction. These self-supervision methods lead to improved performance on dense semantic downstream tasks such as object detection or semantic/instance segmentation. However, their design and evaluation protocols are not focused on 3D vision and lower-level geometric tasks such as depth, motion and flow estimation.

For such tasks, the self-supervision signal can be obtained via view synthesis, where the model is trained by enforcing photometric consistency [36, 56, 88, 97], or from RGB-D data, where the model is trained to match 2D visual features with 3D geometric representations [29, 41, 52].

3 Cross-view Completion Pre-training

In this section we present CroCo, our self-supervised pre-training method based on cross-view completion and tailored to 3D vision tasks.

Overview. Figure 3 gives an overview of the CroCo architecture. Let $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ be two images of the same scene taken from different viewpoints. Both images are divided into N non-overlapping patches $\boldsymbol{p_1} = \{\boldsymbol{p_1}^1, \ldots, \boldsymbol{p_1}^N\}$, $\boldsymbol{p_2} = \{\boldsymbol{p_2}^1, \ldots, \boldsymbol{p_2}^N\}$, and a number $n = \lfloor rN \rfloor$ of tokens from the first set $\boldsymbol{p_1}$ is randomly masked, with $r \in [0, 1]$ being a masking ratio hyper-parameter. We typically use $r = 0.9$, *i.e.*, 90% of patches from $\boldsymbol{p}_1$ are discarded. We denote the remaining set of visible patches from the first image by $\tilde{\boldsymbol{p}}_\mathbf{1} = \{\boldsymbol{p_1}^i | m_i = 0\}$, with $m_i = 0$ indicating that the patch $\boldsymbol{p_1}^i$ is not masked ($m_i = 1$ otherwise).

An encoder $\mathcal{E}_\theta$ processes $\tilde{\boldsymbol{p}}_\mathbf{1}$ and $\boldsymbol{p_2}$ independently, and a decoder $\mathcal{D}_\phi$ takes the encoding $\mathcal{E}_\theta(\tilde{\boldsymbol{p}}_\mathbf{1})$, conditioned on encoding $\mathcal{E}_\theta(\boldsymbol{p}_2)$, in order to reconstruct $\boldsymbol{p}_1$:

$$\hat{\boldsymbol{p}}_\mathbf{1} = \mathcal{D}_\phi \left(\mathcal{E}_\theta(\tilde{\boldsymbol{p}}_\mathbf{1}); \mathcal{E}_\theta(\boldsymbol{p}_2) \right). \tag{1}$$

Details on the encoder $\mathcal{E}_\theta$. A Siamese network denoted by $\mathcal{E}_\theta$ with shared weights θ, implemented as a ViT [26], serves to encode the two input images previously split into independent patch sets $\tilde{\boldsymbol{p}}_\mathbf{1}$ and $\boldsymbol{p_2}$. In this study, we consider images of resolution 224×224, with a patch size of 16×16 pixels. Following ViT, the encoder consists in a linear projection of the flattened input RGB patches,

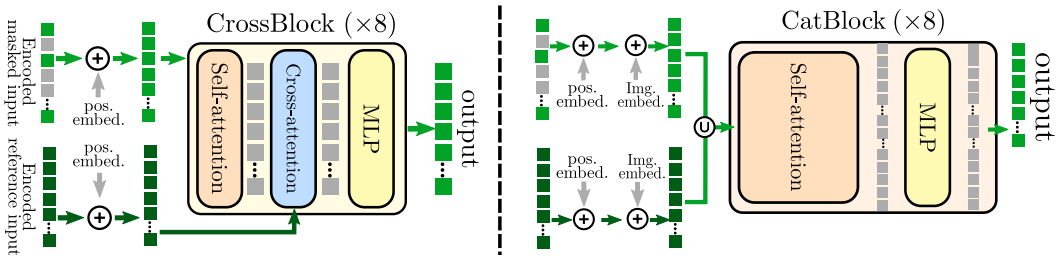


Figure 4: **Attention blocks in the decoder.** CrossBlock (left) combines information from the input sets by alternating self-attention and cross-attention, while CatBlock (right) concatenates the two sets before applying self-attention.

to which sinusoidal positional embeddings are added, followed by a series of transformer blocks [77], each composed of a multi-head self-attention block and an MLP (Multi-Layer Perceptron).

Details on the decoder $\mathcal{D}_\phi$. The decoder $\mathcal{D}_\phi$ with weights ϕ takes as input the set of encoded tokens from the first image $\mathcal{E}_\theta(\tilde{\boldsymbol{p}}_\mathbf{1})$ concatenated with a learned representation $\boldsymbol{e}_{\text{mask}}$ that is repeated n times to account for the masked patches. These tokens are iteratively processed by an attention-based block that also takes into account the encoded tokens $\mathcal{E}_\theta(\boldsymbol{p}_\mathbf{2})$ from the reference image $\mathbf{x_2}$. A sinusoidal positional encoding is added to all tokens before being decoded by $\mathcal{D}_\phi$. We experiment with two different architectures for the decoder block that are illustrated in Figure 4, see Appendix B for detailed equations. The first one, termed *CrossBlock*, consists of (a) multi-head self-attention on the tokens representing the first image; (b) multi-head cross-attention between these tokens and the tokens in $\mathcal{E}_\theta(\boldsymbol{p}_\mathbf{2})$ from the reference image, and (c) an MLP. The second architecture, termed *CatBlock*, proceeds by concatenating the tokens from the two images, with an added learnable embedding for each of the two images, and then feeding these tokens to a series of standard transformer blocks, *i.e.*, each consisting of (a) multi-head self-attention and (b) an MLP layer. Only the subset of tokens corresponding to the first image is taken into account for the final prediction. These two attention blocks represent a trade-off between model size and computational cost. Indeed, *CrossBlock* has more learnable parameters, due to the cross-attention module, but a lower computational cost as it avoids the quadratic complexity of computing self-attention over the joint token set of size $2N$.

Training the CroCo model. The decoder produces one feature vector per patch token, which is processed by a fully-connected layer that outputs 3 values (R, G and B) per pixel for each patch, *i.e.*, $16 \times 16 \times 3 = 768$ values for squared patches with side 16. These outputs serve as predicted reconstruction $\hat{\boldsymbol{p}}_\mathbf{1}$, evaluated using a reconstruction error such as the Mean Squared Error (MSE) loss between predictions and ground-truth values of the pixels, averaged over the set of all masked tokens, denoted $\boldsymbol{p}_\mathbf{1} \backslash \tilde{\boldsymbol{p}}_\mathbf{1}$. Thus to perform self-supervised pre-training, the network is trained to minimize:

$$\mathcal{L}(\boldsymbol{x}_\mathbf{1}, \boldsymbol{x}_\mathbf{2}) = \frac{1}{|\boldsymbol{p}_\mathbf{1} \backslash \tilde{\boldsymbol{p}}_\mathbf{1}|} \sum_{\boldsymbol{p}_\mathbf{1}^i \in \boldsymbol{p}_\mathbf{1} \backslash \tilde{\boldsymbol{p}}_\mathbf{1}} \| \hat{\boldsymbol{p}}_\mathbf{1}^i - \boldsymbol{p}_\mathbf{1}^i \|^2. \tag{2}$$

We additionally try a variant of the loss where each target patch is normalized using the mean and standard deviation of all pixel values within this patch.

Pre-training details. We implement our CroCo model in PyTorch [59] and train the network for 400 epochs using the AdamW optimizer [54]. We use a cosine learning rate schedule with a base learning rate of 1.5×10^{-4} for an effective batch size of 256; with a linear warmup in the first 40 epochs. As encoder, we use a ViT-Base/16 backbone, *i.e.*, a series of 12 transformer blocks with 768 dimensions and 12 heads for self-attention with patches of size 16×16. For both *CrossBlock* and *CatBlock* decoders, we use a series of 8 blocks with 512 dimensions and 16 attention heads. To account for the different dimensions of the encoder and the decoder, we apply a fully-connected layer between them.

Pre-training data. We train our model on a set of synthetic image pairs of 3D indoor scenes derived from the HM3D [64], ScanNet [23], Replica [72] and ReplicaCAD [75] datasets as follows. In each 3D scene, we randomly sample up to 1000 pairs of camera viewpoints with a co-visibility greater than 50%, and render these pairs of images using the Habitat simulator [68]. In total, we generated a dataset of 1,821,391 pairs, that we refer to as *Habitat* in this paper.

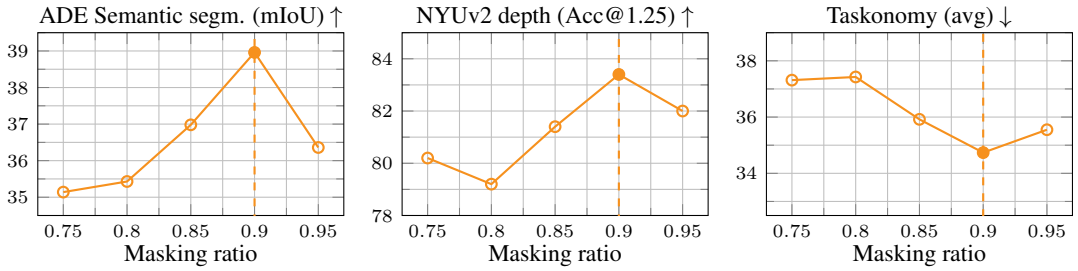


Figure 5: **Impact of the masking ratio** using the CrossBlock decoder and a loss on RGB values.

Finetuning CroCo. Our CroCo model can be used as pre-training to solve either monocular or binocular tasks, dense or not. For monocular task, the ViT-Base encoder is used alone, while binocular tasks benefit from the pre-training of both the encoder and the decoder.

4 Experimental results

We evaluate CroCo on both monocular downstream tasks in Section 4.1, for which we provide ablations (Section 4.1.1) and a comparison to the state of the art (Section 4.1.2), and on binocular tasks in Section 4.2. Training details for each task are available in the appendix.

4.1 Monocular transfer tasks

High-level semantic tasks. While targeting 3D vision tasks, we still evaluate some high-level semantic tasks, namely *image classification with linear probing* on ImageNet-1K [67] and report top-1 accuracy. We follow the exact same protocol as MAE [38] for that, with global average pooling for CroCo as we did not include a [CLS] token in our model. Additionally, we report results on *semantic segmentation* on ADE20k [94] with 150 classes and 20,210 training images. We follow the protocol of [4] and report mean Intersection-over-Union (mIoU) on the validation set, using the same ConvNext [53] prediction head on top of the encoder.

3D vision tasks. We evaluate *monocular depth prediction* on NYUv2 [70] (795 training and 655 test images) by reporting δ_1, *i.e.*, the percentage of pixels that have an error ratio (max over prediction divided by ground-truth and its inverse) below 1.25. We additionally report results on the Taskonomy dataset [91] (800 training images) and report L1-loss on the 'tiny' test set (54,514 images) for the same 8 tasks as [4]: *curvature, depth, edges, 2D keypoints, 3D keypoints, normal, occlusion and reshading*. For better clarity with decimal number, we report the L1-losses multiplied by 1000. We also report the average ranking (rank.) over the 8 tasks in tables. For these tasks, we use a DPT head [65] on top of the encoder. We additionally experiment on the monocular task of absolute camera pose regression in Appendix D.4 and the binocular task of stereo image matching in Appendix E.3.

4.1.1 Ablations

Masking ratio. We first report in Figure 5 the performance on semantic segmentation (ADE), depth estimation (NYUv2) and Taskonomy when varying the masking ratio r using a *CrossBlock* decoder and a loss without normalization. We observe that the best performance is obtained with a masking ratio of $r = 90\%$. This optimal ratio is higher than what was found for instance in MAE [38] for the auto-completion task. We attribute this to the help provided by the reference image in cross-view completion. We show some reconstruction examples in Figure 2. Despite the small number of visible patches, this amount, combined with the reference view, seems to provide sufficient information to reconstruct the first image. Note that the reconstructions tend to be blurry. This is due to the MSE loss, but as noted in MAE [38], beyond a certain point, sharper reconstructions do not necessarily lead to better pre-training.

Normalized targets. Regressing RGB values normalized by the mean and standard deviation within each patch has proven to be effective for MAE [38]. The first two rows of Table 1 compare the results with or without this normalization for CroCo. We observe that it indeed consistently improves the performance on all tasks, and we therefore use it in the remainder of the experiments.

Table 1: **Impact of normalizing targets and decoder block** with a masking ratio of 90%. We also indicate the number of FLOPs and parameters for the full network (encoder and decoder).

Normalized Target	Decoder Block	ADE ↑ segm.	NYUv2 ↑ depth	Taskonomy ↓ avg.	Taskonomy ↓ rank.	FLOPs	Params
	CrossBlock	39.0	83.4	34.74	2.50	50.2G	120M
✓	CrossBlock	40.6	85.6	**33.00**	**1.63**	50.2G	120M
✓	CatBlock	**41.3**	**86.2**	33.35	1.88	58.5G	111M

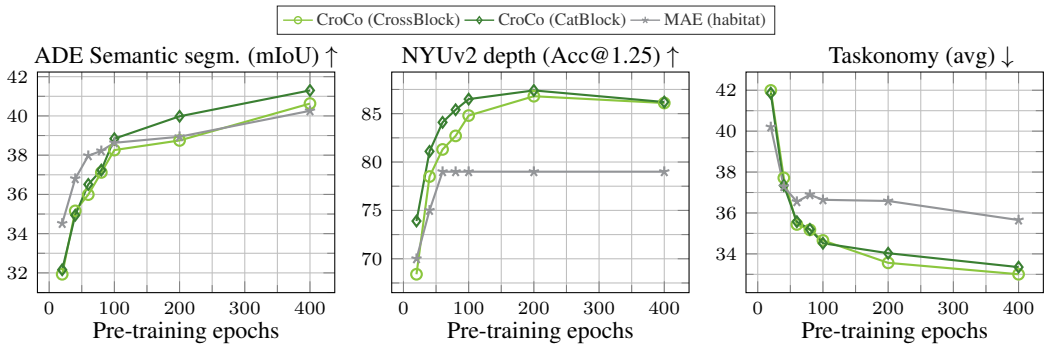


Figure 6: **Performance as a function of the number of pre-training epochs** for various models.

Decoder architecture. In Table 1, we compare decoders built with the two proposed attention blocks: CrossBlock which uses cross-attention to exchange information between the two images, and CatBlock which concatenates the tokens from both images. Note that CatBlock has a smaller number of learnable parameters but a higher number of FLOPs. The performances obtained with the two proposed architectures are quite similar; CatBlock yielding slightly better performance for semantic segmentation or depth estimation while CrossBlock performs better on Taskonomy. We favor the use of CrossBlock in what follows. In Appendix C.3, we provide an ablation study on the number of blocks used in the decoder.

Leveraging the decoder. Note that the pre-trained decoder can also be used when finetuning the model on monocular tasks: it suffices to feed the input image twice to the whole CroCo network, which in practice is done by duplicating the encoder output before passing it to the decoder. While this significantly increases the computational cost of the model, it yields a consistent gain of performance from 40.6 to 41.0 on ADE, 86.1 to 88.1 on NYUv2, and an improvement on 6 out of 8 tasks on Taskonomy, see further details in Appendix D.2.

Training profile. In Figure 6 we show the performance on the downstream tasks as the pre-training progresses up to 400 epochs, when using CrossBlock or CatBlock in the decoder. We observe that the performance on 3D vision tasks – depth estimation on NYUv2 and Taskonomy – reaches a plateau and thus we evaluate our model pre-trained for 400 epochs in what follows. We additionally train a MAE model[1] on the Habitat dataset, using all available images (*i.e.*, twice the number of pairs). When pre-training on Habitat, both models achieve similar performance on the task of semantic segmentation, while CroCo significantly outperforms MAE on depth prediction and Taskonomy. This shows the benefit of using cross-view completion pre-training, rather than pure auto-completion, for 3D vision downstream tasks.

Training pairs. CroCo pre-training requires pairs of images, which we obtain in our experiments by sampling two different points of view using the Habitat simulator. As an alternative, we evaluate using geometric transformations (homographies, rotations, scaling, crop, *etc.*) applied to an input image in order to generate the reference view; we report the results in Table 2 after 400 epochs of pre-training on Habitat. We observe a clear drop in performance on all downstream tasks, *i.e.*, semantic segmentation, monocular depth estimation and Taskonomy. One hypothesis is that in this case – when there exists a synthetic transform between the two images – the model can solve the

[1] `https://github.com/facebookresearch/mae`

Table 2: **Comparison with artificial pairs.** We compare pairs obtained by geometric transformations of one image to pairs sampled from two different viewpoints.

image pairs from	ADE ↑	NYUv2 ↑	Taskonomy ↓	
	segm.	depth	avg.	rank.
two viewpoints	**38.8**	**86.8**	**33.56**	**1.25**
geometric transformations of one image	27.0	66.1	47.81	1.75

Table 3: **Comparison to the state-of-the-art pre-training methods** on semantic tasks (image classification on ImageNet-1K with linear probing and semantic segmentation on ADE) and 3D vision tasks (NYUv2, Taskonomy), with a ViT-Base/16 backbone. We indicate the pre-training data in parenthesis. Best and second best results are **bold** and underlined.

pre-training method (data)	IN1K ↑	ADE ↑	NYUv2 ↑	Taskonomy ↓									
	lin.	segm.	depth	curv.	depth	edges	kpts2d	kpts3d	normal	occl.	reshad.	avg.	rank.
DINO [14] (IN1K)	**78.2**	44.7	66.8	43.04	38.42	3.80	0.16	45.85	65.71	0.57	115.02	39.07	5.00
MAE [38] (IN1K)	75.1	46.1	79.6	41.59	35.83	**1.19**	0.08	44.18	59.20	**0.55**	106.08	36.09	2.13
MutliMAE [4] (IN1K)	60.2	**46.4**	83.0	41.42	35.38	2.17	**0.07**	44.03	60.35	0.56	105.25	36.17	2.75
MAE (Habitat)	32.5	40.3	79.0	42.06	33.63	1.79	0.08	44.81	59.76	0.56	102.54	35.65	2.88
CroCo (Habitat)	37.0	40.6	**85.6**	**40.91**	**31.34**	1.74	0.08	**41.69**	**54.13**	**0.55**	**93.58**	**33.00**	**1.25**

pretext task of cross-view completion by directly fitting this transform, without reasoning about the geometry of the scene.

This in turn raises the question of how to sample optimal viewpoints when generating synthetic image pairs using the Habitat simulator. To answer it, we study the relation between downstream performance and co-visibility in pre-training image pairs (*i.e.*, how much visual content is shared between the two images) in Appendix C.2. In short, it shows that using image pairs with a co-visibility ratio of approximately 0.5 leads to the best pre-trained models, and that on the one hand, very little co-visibility between the two images is sub-optimal as the task boils down to auto-completion, and on the other hand , if the two images composing a pair overlap too much, performance drops as the pre-training task becomes trivial.

4.1.2 Comparison to the state of the art

In Table 3, we compare the proposed CroCo pre-training strategy to the state of the art on monocular tasks. In particular, we compare to DINO [14], a state-of-the-art self-supervised method based on instance discrimination, and to MIM methods with MAE [38] and MultiMAE [4]. The latter uses extra data modalities, namely depth and semantic segmentation generated by off-the-shelf supervised models for pre-training. All three methods rely on the ImageNet-1K (IN1K) dataset [67] for pre-training. This leads to high performance on high-level semantic tasks compared to our approach. Note that DINO, which outputs a global representation, performs best on ImageNet-1K, while MIM-based approaches, which already output dense patch-level predictions during pre-training, obtain better results on dense tasks such as semantic segmentation or depth.

To measure the impact of the pre-training dataset, we report the performance of a MAE model trained on Habitat, *i.e.*, with exactly the same data and ViT-Base architecture as used for CroCo. In this case, the performance on semantic tasks largely drops compared to pre-training on ImageNet (*e.g.* 75.1% to 32.5% in classification on IN1K, see Table 3). This confirms that the high performance on semantic tasks are mostly due to the use of IN1K for pre-training. When evaluating 3D vision downstream tasks, such as depth estimation on NYUv2 or Taskonomy, CroCo significantly outperforms all other methods. Interestingly, it even outperforms MultiMAE for depth estimation on NYUv2 with 85.6% Acc@1.25 *vs*. 83.0%, while other approaches are below 80%. On Taskonomy, CroCo performs best on 6 tasks out of 8 and is second best on the two other tasks, with an average rank of 1.25/5.

4.2 Applications to binocular tasks

We now empirically demonstrate that our CroCo model achieves competitive performance in two binocular tasks, namely optical flow and relative pose estimation, without any task-specific design.

Table 4: **Optical flow results** on the training set of the MPI-Sintel dataset for various pre-training methods when finetuned on AutoFlow.

encoder init.	decoder init.	MPI-Sintel clean	MPI-Sintel final
random	random	18.81	18.97
MAE (IN1K)	random	4.68	5.16
MAE (Habitat)	random	4.63	5.24
CroCo (Habitat)	**CroCo** (Habitat)	**3.00**	**3.60**

Optical Flow. We treat optical flow as a straightforward regression task and do not change the pre-training architecture except for modifying the regression head to output two flow channels instead of 3 RGB color channels. We finetune our network on the public 40,000 synthetic pairs from AutoFlow [73], using a simple MSE loss on 224×224 crops, for 100 epochs, without any data augmentation besides color jittering. We evaluate on the MPI-Sintel [10] dataset (1041 image pairs in the train split), on the clean and final renderings, using the average endpoint error (AEPE) metric. To test on high resolution 1024×536 images, we regularly sample overlapping tiles of size 224×224 at the same position in both input images. We regress the flow between each pair of corresponding tiles. To recombine flow values, at each pixel location in the final prediction we use the flow value predicted by the nearest tile (*i.e.*, based on the tile center).

In Table 4 we compare performance for various pre-training strategies and datasets on the MPI-Sintel training set. Without pre-training, the network is unable to learn anything useful, as shown by the large AEPE values, despite thorough but unsuccessful hyper-parameter tuning. This may be due to the high difficulty of the task coupled with the limited amount of training data. The performance significantly improves when we initialize the encoder with an MAE pre-trained model. With a CroCo initialization of both encoder *and* decoder weights, however, we again observe a large improvement with an average decrease of the error by almost 2 pixels *w.r.t.* the best MAE model on both clean and final renderings. Interestingly, we find that even with just 2 blocks in the decoder, a model pre-trained with CroCo still outperforms an MAE model having 8 decoder blocks (3.59/4.35 AEPE on clean/final, resp., compared to 4.63/5.24 AEPE for MAE; see Appendix E.1 for a complete ablation). These results clearly demonstrate the benefits from pre-training with the cross-view completion task.

Our results (3.00/3.60 on clean/final, resp.) are slightly behind recent state-of-the-art approaches like PWC-Net[74] (2.55/3.93), LiteFlowNet2 [42] (2.24/3.78) and RAFT [76] (1.43/2.71), all trained on FlyingChairs [27] and FlyingThings [55] combined. However, our CroCo model remains competitive for this task considering the simplicity of our approach. In particular, note that (a) our model is trained from limited data without any sophisticated data augmentation, (b) our architecture is generic and not specially designed for optical flow prediction, unlike [74, 42, 76] which all rely on 4D cost volumes, (c) regression at test time is simple and straightforward, (d) the small 224×224 resolution used at test time hinders our performance on large displacements, a problem which we leave to future work. We refer to Appendix E.1 for additional ablative experiments and comparisons.

Relative pose estimation. We experiment on the relative pose regression (RPR) downstream task. Given a pair of images, the goal is to predict the relative pose of the camera with respect to a reference view. To do so, we feed to the CroCo model two images corresponding to normalized camera views, and we regress the rigid transformation $(\boldsymbol{R}, \boldsymbol{t}) \in SO(3) \times \mathbb{R}^3$ between these two views using a differentiable Procrustes layer [9] on top of the prediction head to ensure that $\boldsymbol{R}$ is a rotation matrix. We finetune the RPR model with an MSE loss between the predicted relative pose $(\boldsymbol{R}, \boldsymbol{t})$ and a ground truth $(\hat{\boldsymbol{R}}, \hat{\boldsymbol{t}})$: $\|\boldsymbol{R} - \hat{\boldsymbol{R}}\|_F^2 + \lambda \|\boldsymbol{t} - \hat{\boldsymbol{t}}\|^2$, with λ set to 100 in practice. During training, we augment the training image pairs with random permutations, color jittering and virtual camera rotations. We evaluate on the 7-scenes dataset [35], with a single model finetuned on the union of the official training sets of all scenes, and tested on the corresponding test splits. Images are rescaled and cropped to a 224×224 resolution for simplicity. At test time, for each image of the test split we retrieve the nearest image from the training set according to their AP-GeM-18 descriptors [66], and we predict the pose relative to this retrieved image.

For each scene, we report the standard median position and median orientation error in Table 5. We compare results obtained with the model initialized with CroCo to that of a model with an encoder initialized with MAE (Habitat) and a decoder trained from scratch (last two rows). Unsurprisingly,

Table 5: **Relative pose estimation results** with the median camera position and orientation errors on 7-scenes. Finetuning a model pre-trained with CroCo achieves competitive results compared to existing methods directly regressing relative camera pose, without applying any fusion technique.

Method / pre-training	chess	fire	heads	office	pumpkin	redkitchen	stairs	Average
RelocNet* [6]	12cm, 4.14°	26cm, 10.44°	14cm, 10.5°	18cm, 5.32°	26cm, 4.17°	23cm, 5.08°	28cm, 7.53°	21cm, 6.74°
NC-EssNet* [96]	12cm, 5.63°	26cm, 9.64°	14cm, 10.66°	20cm, 6.68°	22cm, 5.72°	22cm, 6.31°	31cm, 7.88°	21cm, 7.50°
CamNet*† [25]	4cm, **1.73°**	**3cm, 1.74°**	5cm, **1.98°**	4cm, **1.62°**	**4cm**, **1.64°**	**4cm**, **1.63°**	**4cm**, **1.51°**	**4cm**, **1.69°**
top1 AP-GeM-18	27.9cm, 12.81°	40.4cm, 16.06°	21.6cm, 16.46°	37.5cm, 12.79°	44.4cm, 12.58°	46.7cm, 13.92°	32.2cm, 14.59°	36cm, 14.2°
MAE (Habitat)	13.2cm, 9.44°	32.0cm, 15.10°	16.0cm, 16.75°	24.8cm, 11.54°	25.4cm, 10.62°	29.4cm, 13.32°	32.8cm, 14.88°	24.8cm, 13.09°
CroCo (Habitat)	**2.4cm**, 2.81°	4.0cm, 3.86°	**3.1cm**, 4.00°	**3.4cm**, 2.53°	4.9cm, 2.79°	5.5cm, 3.72°	11.7cm, 4.53°	5.0cm, 3.46°

*: fuse multiple pose predictions †: exploit temporal information and multi-step retrieval

the former leads to significantly better results than the latter. We furthermore compare our model pre-trained with CroCo to other state-of-the-art RPR methods which directly regress relative camera poses [6, 25] or essential matrices [96] as well as the the retrieval baseline consisting in predicting the identity as relative pose (see Table 5 upper part). Our model outperforms all methods except CamNet [25]. Note that the latter fuses multiple pose predictions, exploit temporal information and multi-step retrieval; in contrast our model directly predicts the relative pose from a (query, map) image pair, without any further processing, and is thus significantly simpler.

5 Discussion

We have introduced the novel task of cross-view completion for pre-training computer vision networks tailored to 3D vision downstream tasks. Our experiments show that cross-view completion allows to learn representations better suited to 3D vision tasks than classical MIM with auto-completion, and straightforwardly transfer to monocular and binocular tasks. A limitation of our model pre-trained with cross-view completion is that it seems less tailored to high-level semantic tasks. This is arguably more related to the choice of the dataset than to the pre-training task itself, and future work could explore the use of cross-view completion in conjunction with more object-centric datasets like ImageNet. Our approach requires pairs of images depicting the same scene, and in this work, we leverage synthetic renderings only. Future work could also extend it to real-world image pairs, which can be obtained without any supervision, for instance with Structure-from-Motion [49, 63] or geo-referencing.

References

[1] Yuki M. Asano, Christian Rupprecht, and Andrea Vedaldi. A Critical Analysis of Self-supervision, or what We Can Learn from a Single Image. In *ICLR*, 2020.

[2] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Michael Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *ECCV*, 2022.

[3] Sara Atito, Muhammad Awais, and Josef Kittler. SiT: Self-supervised vIsion Transformer. *arXiv preprint arXiv:2104.03602*, 2021.

[4] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. MultiMAE: Multi-modal Multi-task Masked Autoencoders. In *ECCV*, 2022.

[5] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language. *arXiv preprint arXiv:2202.03555*, 2022.

[6] Vassileios Balntas, Shuda Li, and Victor Prisacariu. RelocNet: Continuous Metric Learning Relocalisation Using Neural Nets. In *ECCV*, 2018.

[7] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers. In *ICLR*, 2022.

[8] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-Aware Learning of Maps for Camera Localization. In *CVPR*, 2018.

[9] Romain Brégier. Deep regression on manifolds: a 3D rotation case study. In *3DV*, 2021.

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

[11] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2. *arXiv preprint arXiv:2001.10773*, 2020.

[12] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018.

[13] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. In *NeurIPS*, 2020.

[14] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*, 2021.

[15] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018.

[16] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative Pretraining From Pixels. In *ICML*, 2020.

[17] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, 2020.

[18] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020.

[19] Xiaokang Chen, Mingyu Ding, Xiaodi Wang, Ying Xin, Shentong Mo, Yunhao Wang, Shumin Han, Ping Luo, Gang Zeng, and Jingdong Wang. Context Autoencoder for Self-Supervised Representation Learning. *arXiv preprint arXiv:2202.03026*, 2022.

[20] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. In *NeurIPS*, 2020.

[21] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[22] Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. Region Filling and Object Removal by Exemplar-Based Image Inpainting. *IEEE Trans. Image Processing*, 2004.

[23] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *CVPR*, 2017.

[24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL HLT*, 2019.

[25] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. CamNet: Coarse-to-Fine Retrieval for Camera Re-Localization. In *ICCV*, 2019.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021.

[27] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In *ICCV*, 2015.

[28] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative Unsupervised Feature Learning with Convolutional Neural Networks. In *NeurIPS*, 2014.

[29] Mohamed El Banani and Justin Johnson. Bootstrap Your Own Correspondences. In *ICCV*, 2021.

[30] Alaaeldin El-Nouby, Gautier Izacard, Hugo Touvron, Ivan Laptev, Hervé Jegou, and Edouard Grave. Are Large-scale Datasets Necessary for Self-Supervised Pre-training? *arXiv preprint arXiv:2112.10740*, 2021.

[31] Linus Ericsson, Henry Gouk, and Timothy M. Hospedales. How Well Do Self-Supervised Models Transfer? In *CVPR*, 2021.

[32] Yuxin Fang, Li Dong, Hangbo Bao, Xinggang Wang, and Furu Wei. Corrupted image modeling for self-supervised visual pre-training. *arXiv preprint arXiv:2202.03382*, 2022.

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

[34] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised Representation Learning by Predicting Image Rotations. In *ICLR*, 2018.

[35] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time RGB-D camera relocalization. In *ISMAR*, 2013.

[36] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017.

[37] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*, 2020.

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

[39] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*, 2020.

[40] Jared Heinly, Johannes L. Schönberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the World in Six Days as Captured by the Yahoo 100 Million Image Dataset. In *CVPR*, 2015.

[41] Ji Hou, Saining Xie, Benjamin Graham, Angela Dai, and Matthias Niessner. Pri3D: Can 3D Priors Help 2D Representation Learning? In *ICCV*, 2021.

[42] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow CNN - revisiting data fidelity and regularization. *IEEE Trans. PAMI*, 2021.

[43] Martin Humenberger, Yohann Cabon, Nicolas Guerin, Julien Morat, Jérôme Revaud, Philippe Rerole, Noé Pion, Cesar de Souza, Vincent Leroy, and Gabriela Csurka. Robust Image Retrieval-based Visual Localization using Kapture. *arXiv preprint arXiv:2007.13867*, 2020.

[44] Longlong Jing and Yingli Tian. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. *IEEE Trans. PAMI*, 2021.

[45] Alex Kendall and Roberto Cipolla. Geometric Loss Functions for Camera Pose Regression with Deep Learning. In *CVPR*, 2017.

[46] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: a Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, 2015.

[47] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring Plain Vision Transformer Backbones for Object Detection. In *ECCV*, 2022.

[48] Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, Rui Deng, Liwei Wu, Rui Zhao, Ming Tang, and Jinqiao Wang. MST: Masked Self-Supervised Transformer for Visual Representation. In *NeurIPS*, 2021.

[49] Zhengqi Li and Noah Snavely. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. In *CVPR*, 2018.

[50] Biyang Liu, Huimin Yu, and Yangqi Long. Local similarity pattern and cost self-reassembling for deep stereo matching networks. In *AAAI*, 2022.

[51] Songtao Liu, Zeming Li, and Jian Sun. Self-EMD: Self-Supervised Object Detection without ImageNet. *arXiv preprint arXiv:2011.13677*, 2020.

[52] Yunze Liu, Li Yi, Shanghang Zhang, Qingnan Fan, Thomas Funkhouser, and Hao Dong. P4Contrast: Contrastive Learning with Pairs of Point-Pixel Pairs for RGB-D Scene Understanding. arXiv preprint arXiv:2012.13089, 2020.

[53] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *CVPR*, 2022.

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

[55] Nikolaus Mayer, Eddy Ilg, Philip Häusser, 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.

[56] Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: Unsupervised Learning of Optical Flow with a Bidirectional Census Loss. In *AAAI*, 2018.

[57] Mehdi Noroozi and Paolo Favaro. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles. In *ECCV*, 2016.

[58] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting Self-Supervised Learning via Knowledge Transfer. In *CVPR*, 2018.

[59] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019.

[60] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context Encoders: Feature Learning by Inpainting. In *CVPR*, 2016.

[61] Pedro O. Pinheiro, Amjad Almahairi, Ryan Y. Benmalek, Florian Golemo, and Aaron Courville. Unsupervised Learning of Dense Visual Representations. In *NeurIPS*, 2020.

[62] Matteo Poggi, Fabio Tosi, Konstantinos Batsos, Philippos Mordohai, and Stefano Mattoccia. On the synergies between machine learning and stereo: a survey. *IEEE Trans. PAMI*, 2021.

[63] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning CNN image retrieval with no human annotation. *IEEE trans. PAMI*, 2018.

[64] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI. In *NeurIPS datasets and benchmarks*, 2021.

[65] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021.

[66] Jerome Revaud, Jon Almazan, Rafael Sampaio de Rezende, and Cesar Roberto de Souza. Learning with Average Precision: Training Image Retrieval with a Listwise Loss. In *ICCV*, 2019.

[67] 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.

[68] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A Platform for Embodied AI Research. In *ICCV*, 2019.

[69] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion Revisited. In *CVPR*, 2016.

[70] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012.

[71] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the World from Internet Photo Collections. *IJCV*, 2008.

[72] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[73] Deqing Sun, Daniel Vlasic, Charles Herrmann, Varun Jampani, Michael Krainin, Huiwen Chang, Ramin Zabih, William T Freeman, and Ce Liu. AutoFlow: Learning a better training set for optical flow. In *CVPR*, 2021.

[74] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In *CVPR*, 2018.

[75] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training Home Assistants to Rearrange their Habitat. In *NeurIPS*, 2021.

[76] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[77] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[78] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *JMLR*, 2010.

[79] Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. AtLoc: Attention Guided Camera Localization. In *AAAI*, 2020.

[80] Luya Wang, Feng Liang, Yangguang Li, Honggang Zhang, Wanli Ouyang, and Jing Shao. RePre: Improving Self-Supervised Vision Transformer with Reconstructive Pre-training. *arXiv preprint arXiv:2201.06857*, 2022.

[81] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense Contrastive Learning for Self-Supervised Visual Pre-Training. In *CVPR*, 2021.

[82] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan Yuille, and Christoph Feichtenhofer. Masked Feature Prediction for Self-Supervised Visual Pre-Training. In *CVPR*, 2022.

[83] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination. In *CVPR*, 2018.

[84] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning. In *CVPR*, 2021.

[85] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. SimMIM: A Simple Framework for Masked Image Modeling. In *CVPR*, 2022.

[86] Yuwen Xiong, Mengye Ren, Wenyuan Zeng, and Raquel Urtasun. Self-Supervised Representation Learning from Flow Equivariance. In *ICCV*, 2021.

[87] Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance Localization for Self-supervised Detection Pretraining. In *CVPR*, 2021.

[88] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised Learning of Geometry with Edge-aware Depth-Normal Consistency. In *AAAI*, 2018.

[89] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[90] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen, and Jingdong Wang. HRFormer: High-Resolution Transformer for Dense Prediction. In *NeurIPS*, 2021.

[91] Amir Zamir, Alexander Sax, William Shen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018.

[92] Sameh Zarif, Ibrahima Faye, and Dayang Rohaya. Image Completion: Survey and Comparative Study. *IJPRAI*, 2015.

[93] Yucheng Zhao, Guangting Wang, Chong Luo, Wenjun Zeng, and Zheng-Jun Zha. Self-supervised visual representations learning by contrastive mask prediction. In *ICCV*, 2021.

[94] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K Dataset. In *CVPR*, 2017.

[95] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. iBOT: Image BERT Pre-training with Online Tokenizer. In *ICLR*, 2022.

[96] Qunjie Zhou, Torsten Sattler, Marc Pollefeys, and Laura Leal-Taixe. To Learn or Not to Learn: Visual Localization from Essential Matrices. In *ICRA*, 2020.

[97] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In *CVPR*, 2017.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?
   Yes.
   (b) Did you describe the limitations of your work?
   Yes, in the Discussion section (Section 5).
   (c) Did you discuss any potential negative societal impacts of your work?
   No, we do not envision any potential negative societal impact beyond those that are generic to machine learning and data bias.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them?
   Yes.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results?
   N/A.
   (b) Did you include complete proofs of all theoretical results?
   N/A.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)?
   Not with the submission but we are planning to release code if accepted.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)?
   All details are available in the supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)?
   No, because pre-training is costly (about a week on 8 gpus), but our ablations with for instance the different decoder blocks concur towards the same conclusion that CroCo outperforms other pre-training strategies.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)?
   Yes, in the supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators?
   Yes.
   (b) Did you mention the license of the assets?
   Yes, in the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL?
   No.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating?
   N/A.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content?
   N/A.
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable?
N/A.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable?
N/A.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation?
N/A.

Appendix

This appendix is structured as follows. In Section A, we first provide a list of acronyms used throughout the paper, followed by further details about our decoder architectures (Section B). In Section C we detail the parameter setting used in our pre-training and we provide an analysis about how the overlap between the training image pairs affects the pre-training and downstream performance (Section C.2) as well as an ablation on the impact of the decoder depth (Section C.3). Section D then gives further details about monocular tasks, including a set of experiments on a complementary task, namely absolute pose regression (Section D.4). In Section E, we give more details and visual examples concerning the binocular tasks as well as a complementary task, namely stereo matching (Section E.3). Next, we list assets and computing resources in Section F. Finally, we provide additional reconstruction examples and introduce a video of sample scene reconstructions (Section G).

A List of acronyms

We provide the list of acronyms used in the paper, excluding acronyms related to papers or datasets with direct references.

acronym	meaning
Acc@X	Accuracy at a certain error threshold X
AEPE	Average EndPoint Error
CroCo	Cross-view Completion
DINO	DIstillation with NO labels [14]
DPT	Dense Prediction Transformer [65]
FLOPs	Floating-Point Operations
IN1K	ImageNet-1K [67]
mIoU	mean Intersection-over-Union
MAE	Masked Auto-Encoder [38]
MIM	Masked Image Modeling
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLP	Natural Language Processing
RPR	Relative Pose Regression
ViT	Vision Transformers [26]

B Details on the decoder architectures

In this section, we provide detailed equations about the blocks used in the CroCo model architecture.

For the encoder, we use a standard transformer block. Let $X \in \mathbb{R}^{N \times D}$ be the N D-dimensional tokens as input to the transformer block. The transformer block computes:

$$\begin{aligned} \bar{X} &= \text{LayerNorm}(X) \\ X' &= X + \text{Attention}\left(W_Q^S \bar{X}, W_K^S \bar{X}, W_V^S \bar{X}\right) \\ \text{Output} &= X' + \text{MLP}(\text{LayerNorm}(X')), \end{aligned} \tag{3}$$

where W_Q^S, W_K^S, W_V^S are learnable parameters. In practice, a bias is also learned and applied to these 3 projections but are omitted in the equations for the sake of clarity. The attention itself is computed classically as:

$$\text{Attention}(Q, K, V) = \text{Proj}\Big(\text{softmax}\left(\frac{QK^\top}{\sqrt{D}}\right) V\Big), \tag{4}$$

where Proj denotes a projection layer, *i.e.*, a fully-connected layer.

We now provide equations for the two blocks we have tried in the decoder, namely the *CrossBlock* and the *CatBlock*.

In the *CrossBlock* decoder architecture, self-attention and cross-attention are used one after the other. Let $X, Y \in \mathbb{R}^{N \times D}$ be the N input tokens to the block from the two views respectively. The CrossBlock output is computed by:

$$\begin{aligned}
\bar{X} &= \text{LayerNorm}(X) \\
\bar{Y} &= \text{LayerNorm}(Y) \\
X' &= X + \text{Attention}\left(W_Q^S \bar{X}, W_K^S \bar{X}, W_V^S \bar{X}\right) \\
X'' &= X' + \text{Attention}\left(W_Q^C \text{LayerNorm}(X'), W_K^C \bar{Y}, W_V^C \bar{Y}\right) \\
\text{Output} &= X'' + \text{MLP}(\text{LayerNorm}(X'')),
\end{aligned} \tag{5}$$

where W_Q^S, W_K^S, W_V^S denote learnable parameters for the self-attention and likewise W_Q^C, W_K^C, W_V^C for the cross-attention.

For the *CatBlock* decoder architecture, and following the same notations, we first concatenate X and Y to form $Z = [X + v_1, Y + v_2] \in \mathbb{R}^{2N \times D}$ before the first block, where v_1, $v_2 \in \mathbb{R}^D$ are learnable embeddings specifying the input view. The CatBlock output is computed by:

$$\begin{aligned}
\bar{Z} &= \text{LayerNorm}(Z) \\
Z' &= Z + \text{Attention}\left(W_Q^S \bar{Z}, W_K^S \bar{Z}, W_V^S \bar{Z}\right) \\
\text{Output} &= Z' + \text{MLP}(\text{LayerNorm}(Z')).
\end{aligned} \tag{6}$$

C Further details on cross-view completion pre-training

C.1 Detailed pre-training settings

We report below the detailed parameter setting we used in our pre-training.

Hyperparameters	Value
Optimizer	AdamW [54]
Base learning rate	1.5e-4
Weight decay	0.05
Adam β	(0.9, 0.95)
Batch size	256
Learning rate scheduler	Cosine decay
Training epochs	400
Warmup learning rate	1e-6
Warmup epochs	40
Masked tokens	90%
Input resolution	224×224
Augmentation	Homography, Color jitter

C.2 Ablation on overlaps between pre-training pairs

For pre-training, we use synthetic pairs generated using the Habitat simulator, keeping pairs with a co-visibility ratio over 0.5. Figure 7 presents some statistics about the distribution of viewpoints considered. Intuitively, the choice of this co-visibility threshold was guided by two important observations: (a) if two images composing a pair overlap too little, the task boils down to auto-completion, therefore we set a threshold on the minimal co-visibility ratio of 0.5 in our main experiments, (b) if two images composing a pair overlap too much, the task becomes trivial, therefore

we encourage large viewpoint changes between images in our training set. In this section, we present an ablation to better understand what makes good pre-training pairs.

Formally, we define the visibility ratio $v_{i,j}$ of a view i with respect to an other view j as the ratio of pixels of image i that are visible in image j. We ignore pixels where no geometry is rendered by the Habitat simulator for this computation. We similarly define the co-visibility ratio between two views i and j as $\min(v_{i,j}, v_{j,i})$. To study the influence of co-visibility on pre-training, we generate multiple training sets each composed of $700,000$ image pairs with different co-visibility distributions, and pre-train CroCo on these sets for a number of steps equivalent to 200 epochs of the original dataset, while keeping all other parameters fixed. Figure 8 provides examples of such training pairs. We report in Figure 9 the performance achieved when pre-training with such pairs and then finetuning on Taskonomy tasks. Overall, we observe better performance for most 3D-related tasks (in particular for *curvature*, *depth*, *keypoints3d*, *normal*, *reshading*) when pre-training with pairs with a co-visibility ratio close to 0.5, compared to pre-training with pairs of greater or lower co-visibility (blue curve in Figure 9). We also experimented with pairs chosen to have co-visibility ratios uniformly distributed over different value ranges and report results in Figure 9. We find that pre-training with co-visibility ratios uniformly distributed over $[0, 1.0]$ leads to worse results than when exclusively sampling pairs with a co-visibility ratio within $[0.4, 0.5]$. This confirms that pairs having an intermediate co-visibility ratio are the most suitable for pre-training, although more extensive experiments would be required to strongly support these findings.

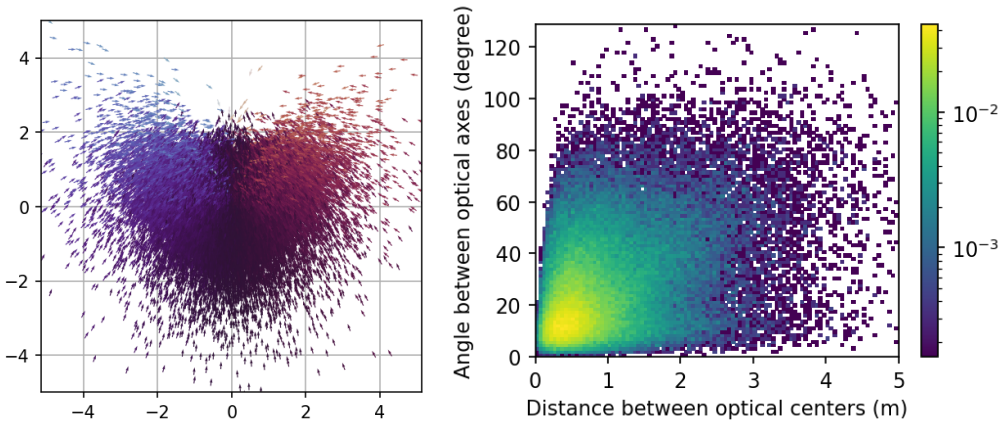
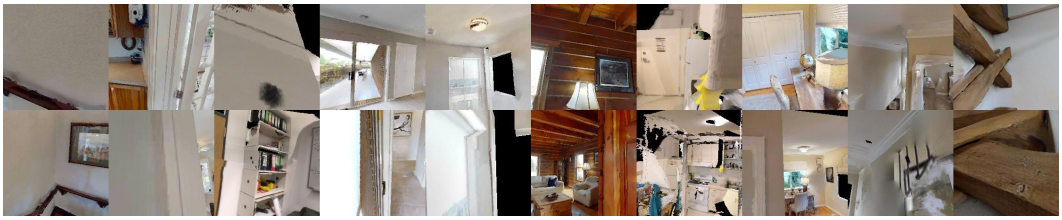
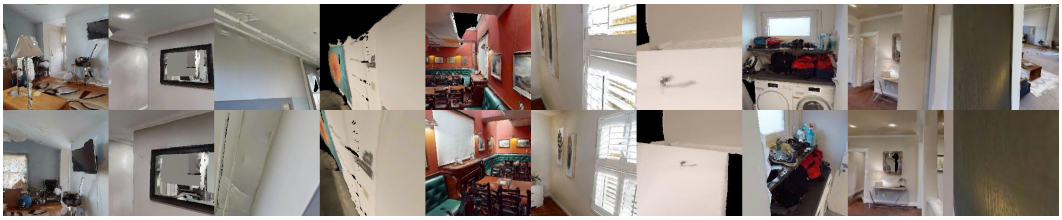


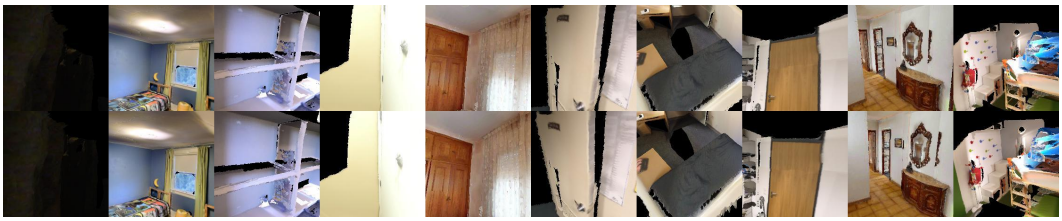
Figure 7: **Distribution of viewpoints pairs used for pre-training.** Left: 2D position (in meters) and orientation (arrow) of one view with respect to the other, once projected on the world horizontal plane. Right: Joint histogram of distances and angles within pairs of views used for training.



Co-visibility ratio between 0 and 0.1



Co-visibility ratio between 0.4 and 0.5



Co-visibility ratio between 0.9 and 1.0

Figure 8: **Some pre-training pair examples for various co-visibility ratios.**

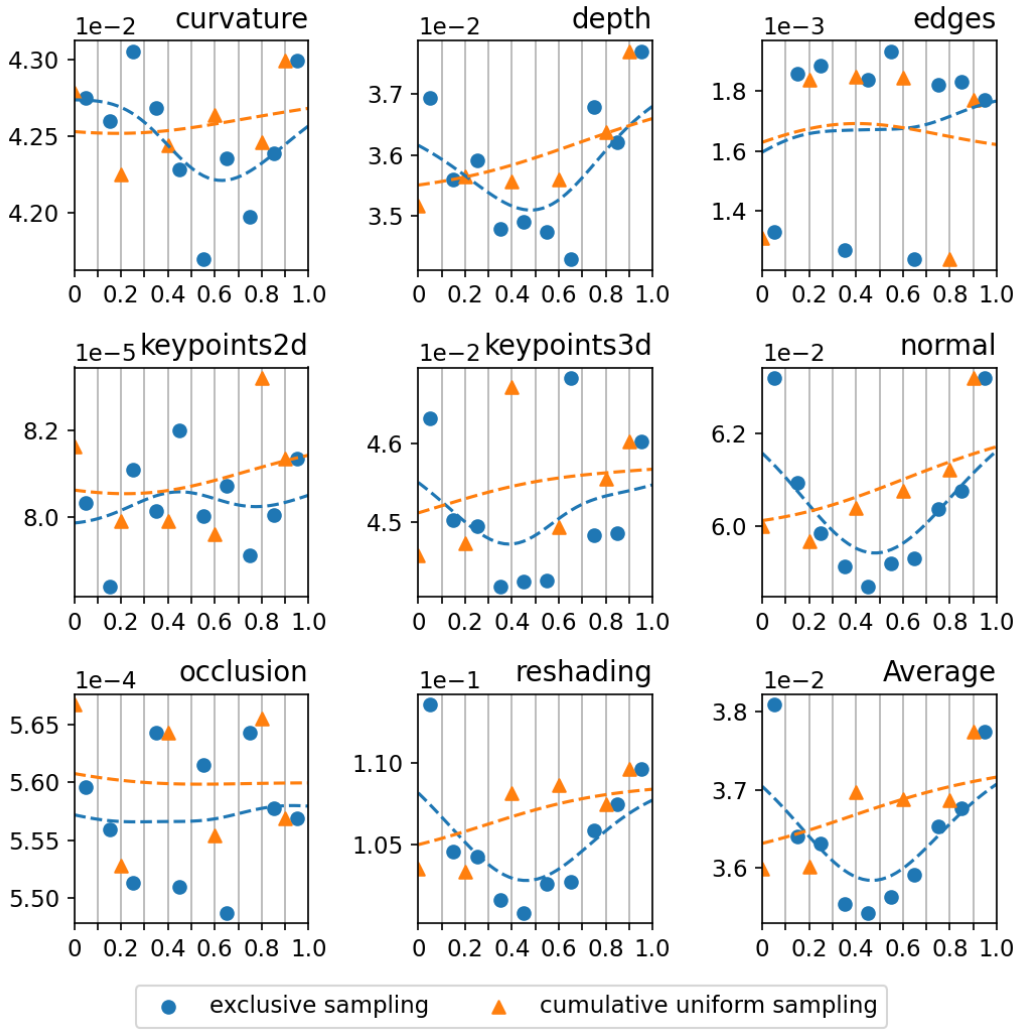


Figure 9: **Ablation on co-visibility between pre-training pairs**. Performance achieved on Taskonomy tasks (vertical axis, lower is better) by a CroCo model pre-trained with image pairs of co-visibility ratio exclusively distributed in the range $[x - 0.05, x + 0.05]$ (blue) or uniformly distributed in the range $[x, 1.0]$ (orange), for various values of x (horizontal axis). Dashed lines represent trend curves obtained by Gaussian smoothing.

C.3 Ablation on decoder depth

In this section, we report an ablation on the decoder depth, *i.e.*, varying the number of decoder blocks for a decoder using *CrossBlock*. The results are shown in Table 6. Note that in the main paper we used a decoder with 8 blocks. We observe that the decoder depth has overall little impact on the performance for monocular tasks (semantic segmentation on ADE, depth prediction on NYUv2 or Taskonomy dense tasks). For the binocular task of optical flow estimation on MPI-Sintel, the error clearly decreases as the depth is increased, showing the importance of a deeper decoder when the decoder is also leveraged. Note however that even with a decoder of depth 2, which means that the network comparing the two images is extremely shallow, a competitive performance can still be reached for optical flow. In particular, the performance with 2 decoder blocks is still largely superior to that of a MAE-pre-trained network having a deep decoder (8 blocks) finetuned in the same conditions.

Table 6: **Impact of decoder depth** with CrossBlock. We used 8 blocks in the decoder in the main paper. Best result per column in bold and second best underlined.

Decoder	ADE ↑	NYUv2 ↑	Taskonomy ↓		MPI-Sintel ↓		FLOPs	Params
Depth	segm.	depth	avg.	rank.	clean	final		
2	38.9	84.6	33.83	2.50	3.59	4.35	39.3G	94M
4	<u>40.2</u>	**86.7**	**32.80**	**1.50**	3.15	3.86	42.9G	103M
6	38.7	83.6	34.14	3.25	<u>3.09</u>	<u>3.79</u>	46.6G	111M
8	**40.6**	<u>85.6</u>	<u>33.00</u>	<u>1.88</u>	**3l.00**	**3.60**	50.2G	120M

D Further details and results for the monocular downstream tasks

D.1 Detailed finetuning settings for monocular tasks

We provide in this section details of the finetuning settings used for our monocular tasks experiments. Linear probing settings are summarized below, Note that they correspond exactly to the settings presented by the authors of MAE [38] with the LARS [89] optimizer and a large batch size.

Hyperparameters	Value
Optimizer	LARS [89]
Base learning rate	0.1
Weight decay	0
Optimizer momentum	0.9
Batch size	16,384
Learning rate sched.	Cosine decay
Warmup epochs	10
Training epochs	90
Augmentation	RandomResizeCrop

We further detail the training settings for semantic segmentation on ADE20k, monocular depth estimation on NYUv2 and Taskonomy regression tasks. These settings closely match those presented by the authors of MultiMAE [4].

Hyperparameters	ADE	NYUv2	Taskonomy
Optimizer	AdamW [54]	AdamW [54]	AdamW [54]
Learning rate	1e-4	3e-5	2.5e-4
Layer-wise lr decay 0.75	-	0.75	
Weight decay	0.05	1e-6	2.5e-2
Adam β	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Batch size	16	16	32
Learning rate sched.	Cosine decay	Cosine decay	Cosine decay
Training epochs	64	1500	300
Warmup learning rate	1e-6	-	1e-6
Warmup epochs	1	100	5
Input resolution	512×512	256×256	384×384
Augmentation	Large Scale jittering Color jittering	RandomCrop Color jittering	-
Drop path	0.1	0.0	0.1

D.2 Leveraging the decoder for monocular tasks

By default, for dense monocular tasks we append a DPT module [65] to the encoder of our CroCo model; this means that the decoder is discarded when finetuning. We now discuss other possible ways of finetuning our CroCo model on downstream tasks.

- `Using the encoder alone.` The simplest and most lightweight option is to use the encoder alone, without the decoder nor the DPT module, by appending an output linear prediction head, trained from scratch for the downstream task. In Table 7 we see, by comparing rows 1 & 2 and rows 5 & 6, that removing the DPT module results in a significant degradation of performance across all tasks.
- `Decoder instead of the DPT module.` The decoder of our CroCo model can be used instead of the DPT module. While it does not fuse information from several layers at different depths, as the DPT module does, the decoder is trained to output dense predictions, and as such it should be easy to finetune for dense tasks other than RGB predictions. To achieve this we initialize the full CroCo model with pre-trained weights, duplicate the encoded features from the input image to serve as input to the decoder, and simply replace the prediction head of the network by a new one trained from scratch. In Table 7 we observe by comparing rows 2 & 3 as well as 6 & 7 that on most tasks, this leads to a very minor degradation of performance. Furthermore, we found that finetuning the model this way was one order of magnitude faster than training a DPT module: convergence on NYUv2 requires approximately 1500 epochs for models without a decoder, against approximately 200 for models using the decoder.
- `Decoder and DPT module.` The decoder can be used together with the DPT module, by extracting the features to be used as input to the DPT module from the CroCo decoder rather than from the encoder. This increases the cost of running the model. However, in Table 7 we see that this leads to consistent performance gains, by comparing rows 2 & 4, or rows 6 & 8.
- `Frozen backbones.` In all cases, we can choose to freeze the backbone and only train the new prediction layers (DPT module or output prediction head when no DPT module is used). Such an experiment is useful to probe what information the pre-trained model captures. It can also help with over-fitting in cases where very little data is used to train for the downstream task. Empirically, we observe by comparing the lower half of Table 7 to the upper part that freezing the backbone degrades the performance in a majority of cases. On average, however, the performance remains surprisingly high, which demonstrates that frozen output features produced by the CroCo model are already meaningful representations for a wide diversity of 3D vision tasks.

Table 7: **Performance of our CroCo model when finetuned for monocular downstream tasks using different modules**, and possibly freezing the backbone network. Note that in the main paper, performance was reported without the decoder, with DPT and with finetuning of the backbone.

architecture			NYUv2 ↑	Taskonomy ↓								
dec.	DPT	frozen	depth	curv.	depth	edges	kpts2d	kpts3d	normal	occl.	reshad.	avg.
With finetuned backbone												
✗	✗	✗	67.8	45.67	63.56	10.50	0.51	60.20	139.69	0.65	186.6	63.42
✗	✓	✗	86.1	40.91	31.34	1.74	**0.08**	41.69	54.13	**0.55**	93.58	33.00
✓	✗	✗	85.9	45.05	33.93	4.18	0.15	44.90	63.02	**0.55**	103.1	36.86
✓	✓	✗	**88.1**	**39.93**	**30.88**	**1.66**	0.58	**40.87**	**52.26**	0.56	**90.77**	**32.18**
With frozen backbone												
✗	✗	✓	51.3	49.39	69.25	34.76	0.68	60.15	170.10	0.80	198.2	72.92
✗	✓	✓	85.2	42.01	**38.18**	**2.43**	**0.09**	45.50	**64.58**	0.55	117.4	38.85
✓	✗	✓	86.4	44.73	75.97	35.16	0.56	60.54	177.12	0.65	220.0	76.84
✓	✓	✓	**87.1**	**41.57**	41.27	3.64	0.24	**43.49**	**62.77**	**0.54**	**116.9**	**38.81**

D.3 Visualization of monocular depth prediction and Taskonomy results

In Figure 10 we display input images from the NYUv2 dataset (validation set), corresponding depth predictions, alone and overlayed on the input images.

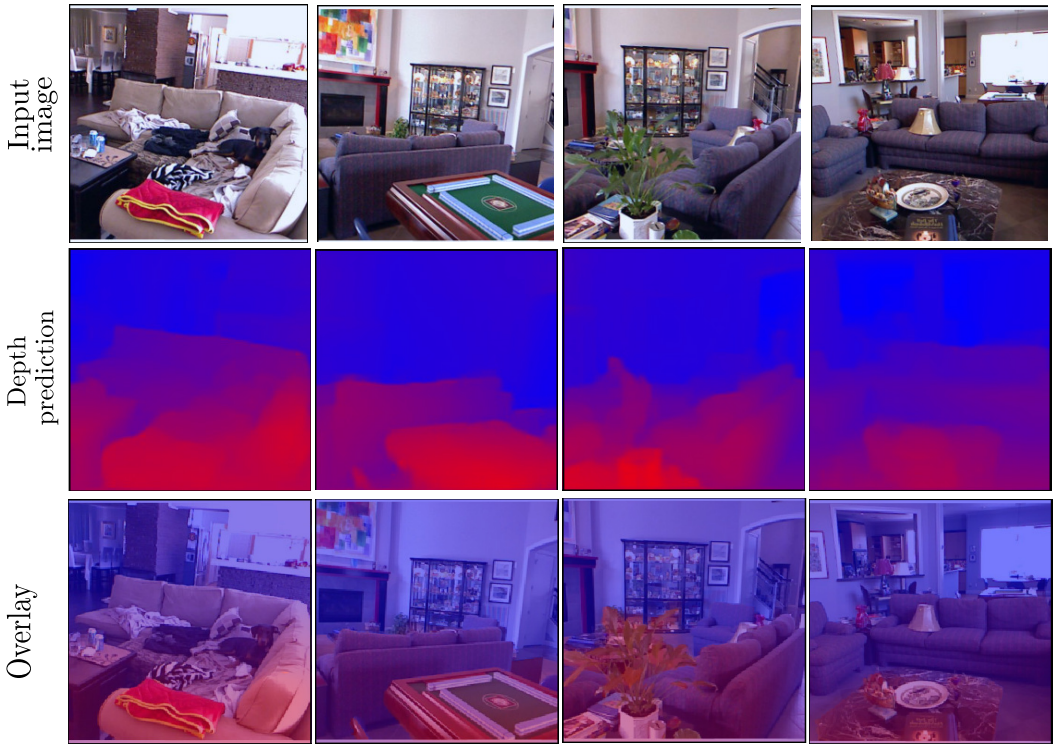


Figure 10: **Qualitative visualization of depth prediction.** First row: the input image; second row: the depth prediction; third row: the depth prediction overlayed over the input image.

Furthermore, in Figure 11 we present results on the Taskonomy dataset for the 8 dense regression tasks.

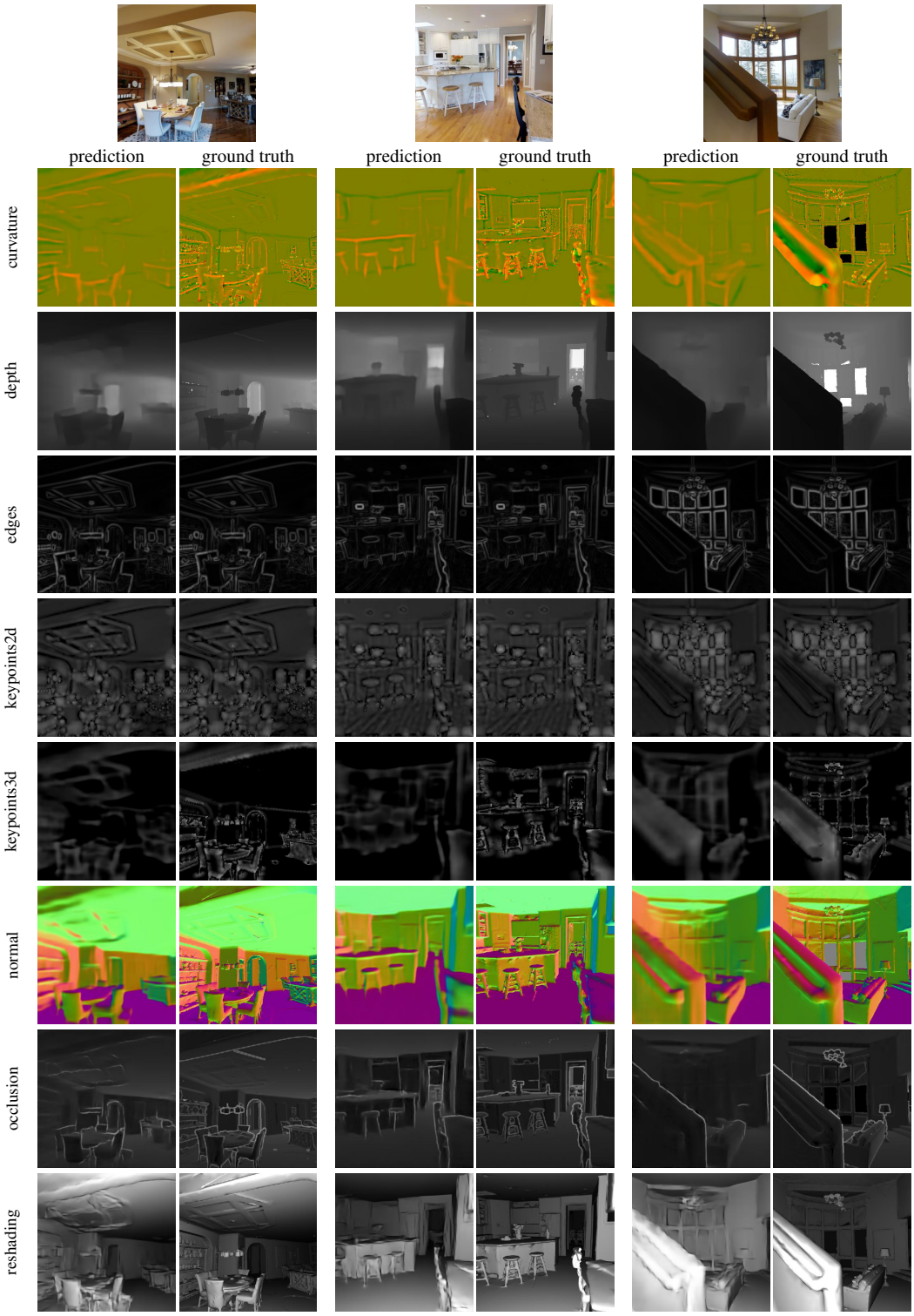


Figure 11: **Example results on Taskonomy.** For the images in the top row, we show the prediction made by our finetuned model and the ground truth, for each of the 8 tasks.

D.4 Application to absolute pose regression from a single image

Task and setting. We apply our pre-trained model to the task of monocular absolute pose regression, where a model directly predicts the absolute camera pose of a given image in a fixed and given environment for which it was specifically trained. In contrast to classical structure-based methods, camera localization methods based on deep neural networks directly regress the absolute pose of a query image [46, 45, 8, 79] and do not require database images or 3D maps at test time. While these methods are in general significantly less accurate than structure-based methods [69, 71, 40, 43], they only require images and their corresponding camera poses as training data.

We use the evaluation code and protocol from AtLoc [79], a recent state-of-the-art improvement of the seminal PoseNet [46] work, which adds an attention module between the encoder output and the pose regressor heads to reweigh the encoded features. The pose regressor is composed of two independent MLP heads: one for predicting the camera pose and one for the camera rotation, represented as a quaternion. In our experiments we append the same regression head (attention module and pose regressor heads) on top of the CroCo encoder, with a global average pooling layer inserted in-between to get a global image representation from the encoder tokens. To finetune the model, we rely on the loss proposed in [8] and also used in [79]:

$$\mathcal{L} = e^{-\beta}\|\boldsymbol{p} - \hat{\boldsymbol{p}}\|_1 + e^{-\gamma}\|\log \boldsymbol{q} - \log \hat{\boldsymbol{q}}\|_1 + \beta + \gamma \tag{7}$$

where β and γ are learned weights that balance the position loss and rotation loss and $\log \boldsymbol{q}$ is the logarithmic form of a unit quaternion (*i.e.*, the corresponding rotation vector). To remove ambiguity between $\boldsymbol{q}$ and $-\boldsymbol{q}$, all quaternions are restricted to the same hemisphere. The finetuning parameters used for the pose regression are the followings:

Hyperparameters	Value
Optimizer	Adam (Base learning rate = 5e-5; weight decay = 5e-4)
Training epochs	500 (with batch size 64)
dropout rate probability	0.5
weights β and γ	initialized with 0 respectively -3
Finetuning dataset	7-Scenes [35]
Input resolution	224×224 (Input field of view 49°)
Augmentation	color jittering and homographies

We benchmark performance on the 7-Scenes dataset [35] on the test split of each scene considered independently. We considered the RGB camera pose annotations from Kapture [43] and the images were rescaled and cropped to a 224×224 resolution with a constant 49° field of view (the largest value fitting in the original images).

Results. We evaluate the impact of the choice of backbone and pre-training method in Table 8 as a function of the size of the training set. The first row corresponds to a ResNet34 backbone, as used in AtLoc, pre-trained with supervision on ImageNet. Other rows correspond to a ViT-Base/16 backbone with different pre-training methods and datasets (MAE, MultiMAE, CroCo). We observe that when using the full training set, all models achieve similar performance, except for the model initialized by MultiMAE which consistently underperforms. Note that CroCo achieves peak validation performance in approx. $50 - 100$ epochs, while ResNet34 pre-trained models need approximately 500 epochs to converge. To better assess the performance of the pre-training model, we significantly reduce the size of the training set and report results in Table 8 using only 5, 10 and 20% of the full training set. We observe that the original AtLoc model, a pre-trained ResNet34 encoder, is unable to learn from such a small amount of data even with data augmentations (random color jittering and homographies simulating camera rotations). Models pre-trained using MAE/MultiMAE perform better, but Croco pre-training leads to the best performance. Finally, while a model trained from scratch (two last rows in Table 8) can achieve decent localization performances using the full training set compared to the other baselines, it generalizes poorly when trained using a more limited amount of data.

Table 8: **Absolute pose regression results** as averaged median errors over the 7 scenes for different backbones, pre-training methods and datasets. Each column corresponds to different ratios of the training set. Best result per column on bold and second best underlined.

Architecture	pre-training	100%	20%	10%	5%
ResNet34	Supervised (ImageNet)	**27.1cm**, 9.0°	34.0cm, 10.9°	43.7cm, 130.°	62.3cm, 17.3°
ViT-Base/16	MAE (ImageNet)	27.9cm, 9.0°	28.0cm, 8.5°	30.6cm, 9.2°	34.4cm, **9.4°**
ViT-Base/16	MAE (Habitat)	28.3cm, 9.1°	28.0cm, 8.3°	30.7cm, 9.1°	35.3cm, 10.1°
ViT-Base/16	MultiMAE (ImageNet)	33.1cm, 9.8°	32.6cm, 9.7°	36.8cm, 10.8°	44.1cm, 12.2°
ViT-Base/16	**CroCo** CrossBlock (Habitat)	27.7cm, **8.6°**	**26.3cm, 7.3°**	**29.0cm, 8.3°**	**32.7cm**, 9.5°
ResNet34	Random	28.1cm, 8.9°	36.6cm, 11.4°	51.3cm, 14.0°	69.3cm, 17.6°
ViT-Base/16	Random	29.1cm, 9.3°	38.3cm, 11.6°	43.6cm, 12.5°	52.7cm, 14.1°

E Further details and results on the binocular downstream tasks

E.1 Optical Flow estimation

Experimental details. We treat optical flow as a straightforward regression task and do not change the pre-training architecture except for modifying the regression head to output two flow channels instead of 3 RGB color channels: the two images are input as such in the Siamese encoders, and the decoder regresses two flow values (u, v) for each pixel using a simple linear head running independently for each output token. We use a *CrossBlock* decoder with 8 layers (similar results are obtained with a *CatBlock* decoder). Training details are provided in Table 9.

Qualitative Results. We show in Figure 12 qualitative visualization of the flow predicted by our method. Overall, we observe that the flow is correctly estimated, even in the case of varied and fast motion. As a limitation, the estimation gets slightly inaccurate when occlusions happen, and blurry on extremely fine-grained motion such as hair tips, as for most methods. Again, we wish to emphasize that the model was finetuned only on 40,000 images without any sort of elaborated data augmentation.

E.2 Relative pose regression

Model Architecture. We replace the original head of the decoder pre-trained with CroCo by a simple head regressing a relative pose $(\boldsymbol{R}, \boldsymbol{t}) \in SO(3) \times \mathbb{R}^3$. This head consists of the following layers. First, a linear projection reduces the dimension of tokens produced by the decoder to 64, in order to limit the computation costs. Second, these tokens are flattened into a unique feature vector, which is

Table 9: **Parameter settings for optical flow, relative pose regression and stereo matching.**

Hyperparameters	Optical Flow	Relative Pose Regression	Stereo Matching
Optimizer	AdamW [54]	AdamW [54]	AdamW [54]
Base learning rate	3e-5	1e-7	1e-4
Weight Decay	0.05	-	-
Batch size	20	64	8
Adam β	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)
Learning rate sched.	Cosine decay	Cosine decay	1 cycle
Finetuning epochs	100	30	400
Linear warmup epochs	1	1	1
Translation weight λ	-	100m	-
Finetuning dataset	AutoFlow [73]	7-scenes [35]	VKITTI [11]
Input resolution	224×224	224×224	1242×375
Augmentation	random crop, color jitter	homography, color jitter	random horizontal flip, color jitter
Input field of view	-	49°	-

Figure 12: **Qualitative visualization of the estimated flow** on some examples from the MPI-Sintel training set, unseen during training. First and third rows: input image pair, second and fourth row: predicted (*left*) and ground-truth (*right*) flows.

processed by a MLP (with a hidden layer of size 1024 and ReLU activations) to regress a 12D output. It is reshaped into an affine transformation $(\boldsymbol{M}, \boldsymbol{t}) \in \mathbb{R}^{3\times3} \times \mathbb{R}^3$. Lastly we use a differentiable special Procrustes layer [9] to orthonormalize $\boldsymbol{M}$ into a rotation matrix $\boldsymbol{R}$ and to predict the relative pose $(\boldsymbol{R}, \boldsymbol{t})$.

Experimental details. The RGB camera pose annotations for the 7-scenes dataset [35] are obtained from Kapture [43]. To select training pairs, we extract global AP-GeM-18 descriptors [66] for every image, and match each training image with its 20 closest neighbors according to their descriptor similarity. Our training set is composed of 26k images and 520k training pairs in total. We rescale and crop the images to a square 224×224 resolution with a constant 49° field of view (the largest value fitting in the original images), and we augment the training image pairs with random permutations, color jittering and homographies that simulate camera rotation.

In our experiments, we trained models using the AdamW [54] optimizer using settings described in Table 9. We found the training to be quite sensitive to its initialization, and we had to try multiple random seeds to achieve decent performances on the training set when training the decoder from scratch with an encoder pre-trained with MAE on Habitat. Using an encoder pre-trained with MAE on ImageNet furthermore always led to poor performance on both the training and test sets in our experiments. We had none of these issues however when using an encoder and decoder pre-trained using CroCo.

E.3 Stereo image matching

Task and settings. In this section, we experiment on the binocular downstream task of stereo matching [62]. We finetune a pre-trained CroCo model to predict pixel-wise disparity values, by simply replacing the linear prediction head. The model takes two rectified images as input and predicts the disparity of every pixel by matching corresponding pixels in the images. We use a

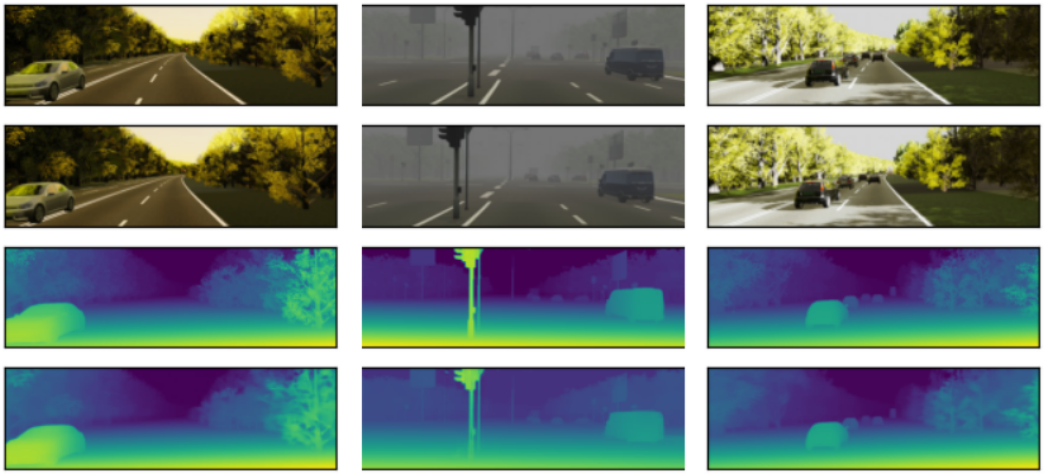


Figure 13: **Qualitative visualization of the estimated disparity** on examples from the VKITTI validation set. First row: input left image; second row: input right image, third row: ground-truth (left) disparity, forth row: predicted (left) disparity. The three examples come from "sunset" (left column), "fog" (center) and "clone" (right column) weather conditions.

MSE-log loss for finetuning and report the error using the 3-pixel 5% discrepancy [15, 20]. We evaluate the stereo matching downstream task on the Virtual KITTI dataset [11] which consists of 5 synthetic sequences cloned from the KITTI tracking benchmark [33]. The dataset additionally provides 9 variants of these 5 sequences under different weather conditions (*e.g.* 'fog', 'rain', *etc.*) and modified camera configurations (*e.g.* rotation to left or right by 15 or 30 degrees). We downscale VKITTI images to a resolution of 224×742, preserving the aspect ratio and crop to 224×736. Additional details on the finetuning hyper-parameters and settings are described in Table 9.

Results. In Table 10 we report results obtained on all 10 weather condition and camera orientation variants in the VKITTI dataset. We compare results obtained when finetuning the CroCo model and the MAE model with a randomly initialized decoder, pre-trained on Habitat. We observe that CroCo pre-training leads to significantly better results for all scenarios. Next, we compare to the state-of-the-art methods for stereo matching, in particular, PSMNet [15], LaC-GweNet [50] and LEAStereo [20]. Finetuning a model pre-trained with CroCo achieves results competitive with the state of the art, without any task-specific model design, such as spatial pyramid pooling [15], hierarchical neural search in 4D feature volume [20] or local similarity patterns for explicit neighbor relationships [50]. We point out that, in contrast to the other methods, our model is directly finetuned on VKITTI without pre-training on the large SceneFlow dataset [55]. In Figure 13 we visualize the disparity prediction by our method, for three different conditions in the VKITTI dataset.

Table 10: **Stereo matching results** with the average 3-px error for 10 VKITTI variants. We compare CroCo pre-training with MAE pre-training as well as other state-of-the-art methods. Best result per column on bold and second best underlined.

Method/ Pre-training	fog	sun-set	clone	over-cast	rain	mor-ning	15°-left	15°-right	30°-left	30°-right	Ave-rage
MAE (Habitat)	1.80	1.75	1.96	1.89	2.13	2.17	2.93	2.07	4.06	2.25	2.30
CroCo (Habitat)	**1.15**	1.69	1.72	1.49	1.77	1.68	2.49	1.58	3.43	1.78	1.88
PSMNet [15]	2.36	2.30	2.38	2.45	2.31	2.39	2.36	2.41	2.33	2.25	2.36
LaC-GwcN [50]	1.67	**1.41**	1.52	1.39	1.30	1.72	1.65	**1.51**	1.50	1.79	1.54
LEAStereo [20]	1.24	1.42	**1.27**	**1.17**	**1.05**	**1.09**	**1.18**	1.65	**1.06**	**1.22**	**1.23**

F Compute resources, code and dataset assets

F.1 Floating point operations (FLOPs)

We calculate the number of floating point operations (or FLOPs) in the most commonly accepted manner, *i.e.*, counting additions and multiplications as separate floating-point operations. To that aim, we borrow and slightly adapt the `flops_computation.py` code from [21]. Note that most of the FLOPs comes from matrix-matrix multiplication operations in the attention and feed-forward layers. For a standard ViT (encoder only), the number of FLOPs can thus be approximated as

$$\#FLOPs \simeq 2L\left(F_{attn-kqv} + F_{attn-scores} + F_{attn-avg} + F_{attn-out} + F_{ff-in} + F_{ff-out}\right)$$

where

$$\begin{aligned} F_{attn-kqv} &= 3ND^2 \\ F_{attn-scores} &= DN^2 \\ F_{attn-avg} &= DN^2 \\ F_{attn-out} &= ND^2 \\ F_{ff-in} &= 4ND^2 \\ F_{ff-out} &= 4ND^2. \end{aligned}$$

For instance, for ViT-Base/16, setting the number of blocks $L = 12$, the number of tokens $N = 14^2$ and the embedding dimension $D = 768$, we find 34.7 GFLOPs with the formula above, which is close to the 35.3 GFLOPs calculated including all other operations (layer norms, patch embeddings, *etc.*). Note that we report exact theoretical FLOPs (counting all operations) in the paper.

F.2 Compute resources

Pre-training a CroCo model for 400 epochs takes about 64 GPU-days on NVIDIA V100. For instance on a 4-GPU server, this is about two weeks.

F.3 Assets used in this submission

We provide below an overview of assets used in our experiments and their licenses.

Asset	License
Habitat pre-training	
HM3D [64]	academic, non-commercial research [hyperlink]
ScanNet [23]	non-commercial research and education [hyperlink]
Replica [72]	non-commercial research and education [hyperlink]
ReplicaCAD [75]	Creative Commons Attribution 4.0 International (CC BY 4.0) [hyperlink]
Habitat simulator [68]	MIT [hyperlink]
High-level semantic tasks	
ImageNet-1K [67]	non-commercial research and education [hyperlink]
ADE [94]	images: non-commercial research and education – annotations: Creative Commons BSD-3 [hyperlink]
Monocular 3D vision tasks	
NYUv2 [70]	public [hyperlink]
Taskonomy [91]	non-commercial research and education [hyperlink]
Optical flow	
AutoFlow [73]	Create Commons Attribution 4.0 International (CC BY 4.0) [hyperlink]
MPI-Sintel [10]	images: Creative Commons Attribution 3.0 (CC BY 3.0) – copyright Blender Foundation | www.sintel.org [hyperlink]
Absolute / relative pose regression	
7-scenes [35]	non-commercial [hyperlink]
Kapture package [43]	BSD 3-Clause Revised [hyperlink]
Stereo matching	
Virtual KITTI [11]	Creative Commons Attribution-NonCommercial-ShareAlike 3.0 [hyperlink]
Code and pre-trained models	
MAE [38]	Creative Commons Attribution-NonCommercial 4.0 International (CC BY NC 4.0) [hyperlink]
MultiMAE [4]	Creative Commons Attribution-NonCommercial 4.0 International (CC BY NC 4.0) [hyperlink]
DINO [14]	Apache License 2.0 [hyperlink]

G Further visual examples

G.1 Video of sample CroCo reconstructions

We attach a video of reconstruction results obtained using the proposed CroCo model on some validation scenes from the HM3D dataset [64], unseen at training time (see Figure 14). For each scene, we consider a constant reference input image and try to reconstruct a sequence of target images from masked inputs, each frame being processed independently using CroCo. The capability of CroCo to solve this challenging task suggests that it is somehow able to capture the 3D layout of the scene, as well as the relative pose of the masked view with respect to the reference view.

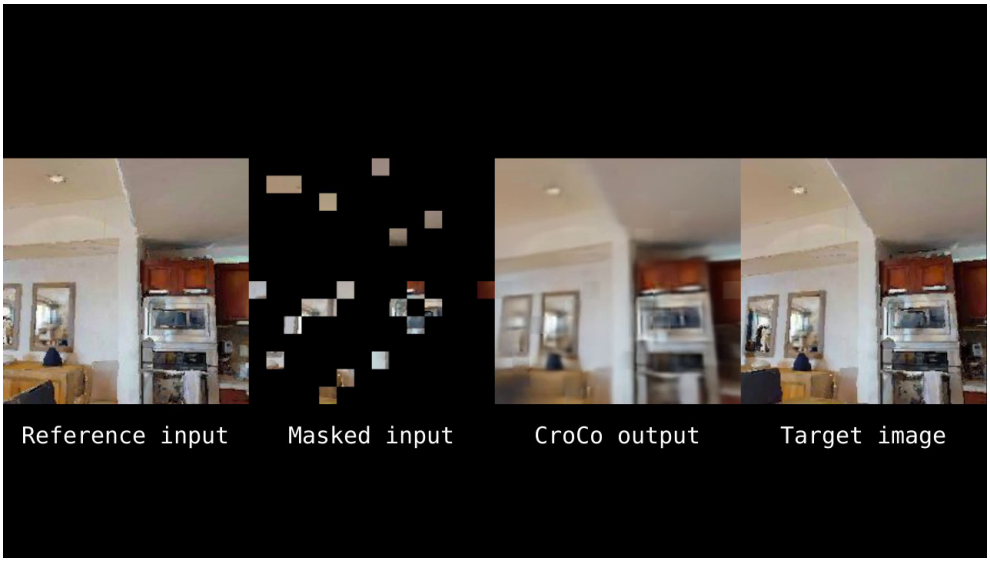


Figure 14: **Excerpt from the supplementary video.** Reconstructions are displayed while varying the viewpoint, with a fixed reference input.

G.2 Additional reconstruction examples

We provide additional reconstruction examples in Figure 15, where we compare reconstructions obtained using our CroCo model pre-trained with an RGB loss and a reference input image (*CroCo RGB*) with reconstructions obtained after replacing this reference image by an image of random uniform noise, independent at each pixel (*CroCo RGB random noise ref.*). Cross-view completion enables a decent reconstruction of the masked image in general, except for areas non-visible in the reference image (*e.g*. *CroCo RGB*, left part of the painting in the third column). When replacing the reference image by some random noise, the reconstruction problem becomes similar to inpainting and some parts of the scene may be wrongly reconstructed due to the lack of available information (*e.g*. wrong size for the window in the first column, missing ice maker in the fridge in the second column).

We also compare reconstructions obtained with CroCo and MAE models pre-trained on Habitat to reconstruct normalized patches (*CroCo norm* and *MAE norm*, last two rows). We use patch statistics from the target image to un-normalize these reconstructions for visualization purposes, which explains that the mean color of each reconstructed patch is relatively well reconstructed in all cases. Yet, we observe that CroCo produces more detailed reconstructions than MAE, the latter often appearing quite tessellated due to an inconsistent reconstruction of normalized patches.

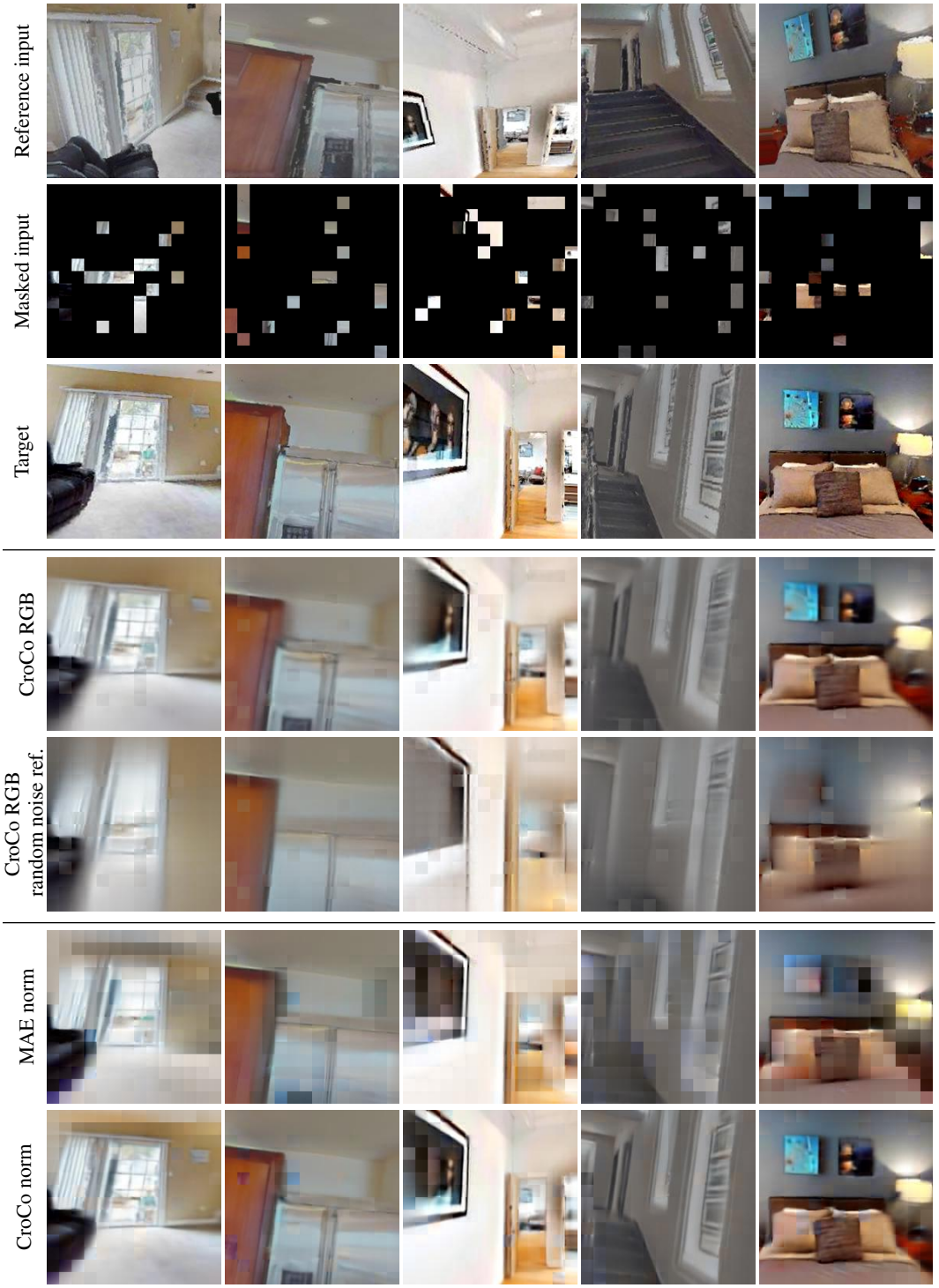


Figure 15: **Additional reconstruction examples** for scenes unseen during training. We compare image reconstructions obtained using our CroCo model pre-trained with an RGB loss using the reference input image (*CroCo RGB*) or replacing it by random noise (*CroCo RGB random noise ref.*). We also compare reconstructions of CroCo and MAE after a pre-training on Habitat to reconstruct normalized patches (*CroCo norm* and *MAE norm*). Patch un-normalization is performed using patch statistics from the target image, for visualization purposes.