

# SACReg: Scene-Agnostic Coordinate Regression for Visual Localization

Jerome Revaud<sup>†</sup>Yohann Cabon<sup>†</sup>Romain Brégier<sup>†</sup>JongMin Lee<sup>\*</sup>Philippe Weinzaepfel<sup>†</sup><sup>†</sup>Naver Labs Europe

firstname.lastname@naverlabs.com

<sup>\*</sup>Seoul University

sdrjseka96@naver.com

## Abstract

*Scene coordinates regression (SCR), i.e., predicting 3D coordinates for every pixel of a given image, has recently shown promising potential. However, existing methods remain mostly scene-specific or limited to small scenes and thus hardly scale to realistic datasets. In this paper, we propose a new paradigm where a single generic SCR model is trained once to be then deployed to new test scenes, regardless of their scale and without further finetuning. For a given query image, it collects inputs from off-the-shelf image retrieval techniques and Structure-from-Motion databases: a list of relevant database images with sparse pointwise 2D-3D annotations. The model is based on the transformer architecture and can take a variable number of images and sparse 2D-3D annotations as input. It is trained on a few diverse datasets and significantly outperforms other scene regression approaches on several benchmarks, including scene-specific models, for visual localization. In particular, we set a new state of the art on the Cambridge localization benchmark, even outperforming feature-matching-based approaches.*

## 1. Introduction

Image-based scene coordinate regression (*SCR*) consists in predicting the coordinates of the 3D point associated to each pixel of a given query image. *SCR* methods have numerous applications in computer vision, and previous work has shown promising potential over the last few years. Such methods have for instance been proposed for visual localization [55, 78, 61, 9] in combination with a Perspective-n-Point (*PnP*) solver [31]. Other applications include object pose estimation [5, 81], depth completion [24, 68, 12, 43, 41], augmented reality or robotics [55, 76].

Unfortunately, existing *SCR* approaches pose significant scalability issues and end up being rather impractical. Most of the time, 3D scene coordinates are directly embedded into the parameters of the learned model, being it a random

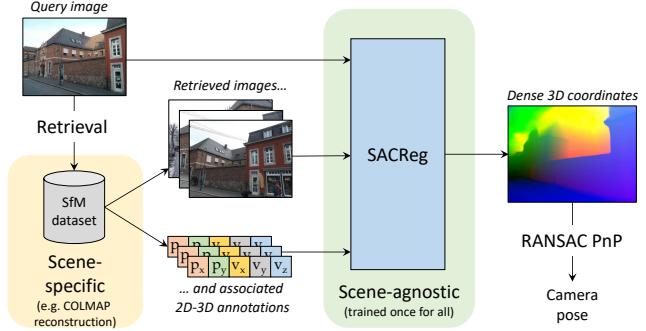


Figure 1: **Overview of the application of our scene-agnostic coordinate regression (SACReg) to visual localization.** Given a query image and a set of related views with sparse 2D/3D annotations retrieved from a database (bottom left), SACReg predicts absolute 3D coordinates for each pixel of the query image (right). This can be used for visual localization using a robust *PnP* algorithm (bottom right). Importantly, SACReg is scene-agnostic: it does not need any retraining for new datasets, only the images and 2D-3D annotations that serve as input are scene-specific.

forest [55] or a neural network [6, 9, 8, 76], hence *de facto* limiting one model to a specific, generally small, scene for which it was trained on. Some recent attempts to mitigate this issue, such as training different experts [8], sharing scene-agnostic knowledge between scenes [62], or heavily relying on dense 3D reconstructions at test time [61, 78], improve by some aspects but still require scene-specific finetuning, can be limited to small scenes, and do not offer scene-agnostic solutions yet.

In this paper, we propose a new paradigm for scene coordinates regression that allows to train a generic model once, and deploy it to any test dataset. As illustrated in Figure 1, our scene-agnostic coordinate regression (SACReg) model takes as input a query image as well as a set of relevant database images for which 3D scene coordinates are available at sparse 2D locations. It then predicts dense 3D coordinates, from which the camera pose can be obtained by solving a Perspective-n-Point (*PnP*) problem. Note that

all inputs to SACReg can be obtained via off-the-shelf methods. Relevant database images, for instance, are obtained using image retrieval techniques [75, 26, 45], while the sparse 2D-3D correspondences are a by-product of the map construction procedure of Structure-from-Motion (SfM) pipelines [53].

As a first contribution, we introduce a generic model for scene-agnostic coordinate regression, illustrated in Figure 2, implemented using standard transformer and convolutional blocks only. It uses a Vision Transformer (ViT) [18] to encode both the query image and the database images. For each database image, input sparse 2D-3D correspondences are injected into the database image features with a so-called *3D mixer* module. This module consists of a series of transformer decoder blocks, as they were called originally [69], *i.e.* ViT blocks with additional cross-attention modules. Then, another transformer decoder combines information from these database features with features extracted from the query image, which are further processed by a convolutional head to finally regress pixelwise 3D scene coordinates.

As a second contribution, we propose to regress an encoding of the 3D coordinates rather than the raw 3D coordinates. Doing so solves a major limitation of existing scene-agnostic approaches [61, 78] which assume small scenes with zero-centered coordinate systems and cannot generalize to unbounded scenes. To that aim, we introduce an invertible and noise-resistant cosine-based encoding of 3D coordinates. We show that it can generalize effortlessly to arbitrary coordinate ranges at test time.

To further ensure generalization, we train our model on diverse sources: outdoor buildings with the MegaDepth dataset [34], indoor environments from the ARKitScenes [3] dataset and synthetic data from the Habitat simulator [52]. We additionally pre-train the network using cross-view completion [73]. Our third contribution is to evaluate our model *as-is* on various visual localization benchmarks, without any finetuning. On the Aachen Day-Night [51] and Cambridge-Landmarks [30] benchmarks, SACReg outperforms current scene-specific and dataset-specific *SCR* methods, while being competitive with state-of-the-art structure-based methods [26]. Even though our model is scene-agnostic, it remains competitive on 7-Scenes [55] compared to scene-specific methods.

## 2. Related work

**Scene-specific coordinates regression.** Several methods have been proposed to estimate dense 3D coordinates for a query image in a scene known at training time. Early approaches [55, 22, 67] used regression forest models to predict the correspondence of a pixel in a RGB-D frame to its 3D world coordinate. However, more recent works [6, 7, 8, 9, 33, 78, 61, 83, 76, 17, 25] have replaced regres-

sion forests with CNN-based models that only require an RGB image. For example, Brachmann *et al.* [6, 7, 8, 9] train neural networks for this task and combine them with a differentiable RANSAC strategy for camera relocalization. Dong *et al.* [17] and Li *et al.* [33] later introduce region classification into their pipelines for effective scene memorization. Huang *et al.* [25] propose to add a segmentation branch to obtain segmentation on scene-specific landmarks, which can then be associated with 3D landmarks in the scene to estimate camera pose. These methods are designed to memorize specific scenes, making them hard to scale and impractical in many scenarios where the test scene is unknown at training time. In contrast, our method can adjust to any scene or dataset at test time by relying on external image retrieval techniques to find relevant database images.

**Scene-agnostic coordinates regression with dense database 3D points.** More related to our work are the scene-agnostic methods of [78, 61]. They regress dense scene coordinates given some reference views for which dense coordinates are already available. They thus require a dense multi-view reconstruction of the scene. Moreover, their methods are limited to small scenes with unit-normalized world coordinates. In contrast, our approach only requires a sparse multi-view reconstruction of the scene and imposes no restriction on coordinate range, making it better suited to large-scale environments.

**Image-based localization** consists in estimating 6-DoF camera pose from a query image. Different approaches can be used towards that goal, and *SCR* is one of them. Recently, learning-based methods in which the pose of a query image is directly regressed with a neural network have been proposed [30, 28, 10, 71, 70, 4]. By training the network with database images and their known ground-truth poses as training set, they learn and memorize the relationship between RGB images and associated camera pose. These direct approaches however need to be trained specifically for each scene. This issue was somehow solved by relative pose regression models [2, 16, 84, 1], which train a neural network to predict the relative pose between the query image and similar database image found by image retrieval. However, their performance tends to be inferior to structure-based methods [53, 57, 23, 26]. Structure-based visual localization frameworks use sparse feature matching to estimate the pose of a query image relative to a 3D map constructed from database images using SfM techniques, such as those employed in [53]. This involves extracting 2D features from images using interest point detectors and descriptors [37, 48, 15, 19, 46, 66, 32, 39, 64, 72, 42, 80, 38], and establishing 2D-3D correspondences. A PnP problem is solved using variants of RANSAC [20], which then return the estimated pose. However, the structure-based methods have to store not only 3D points but also keypoints de-

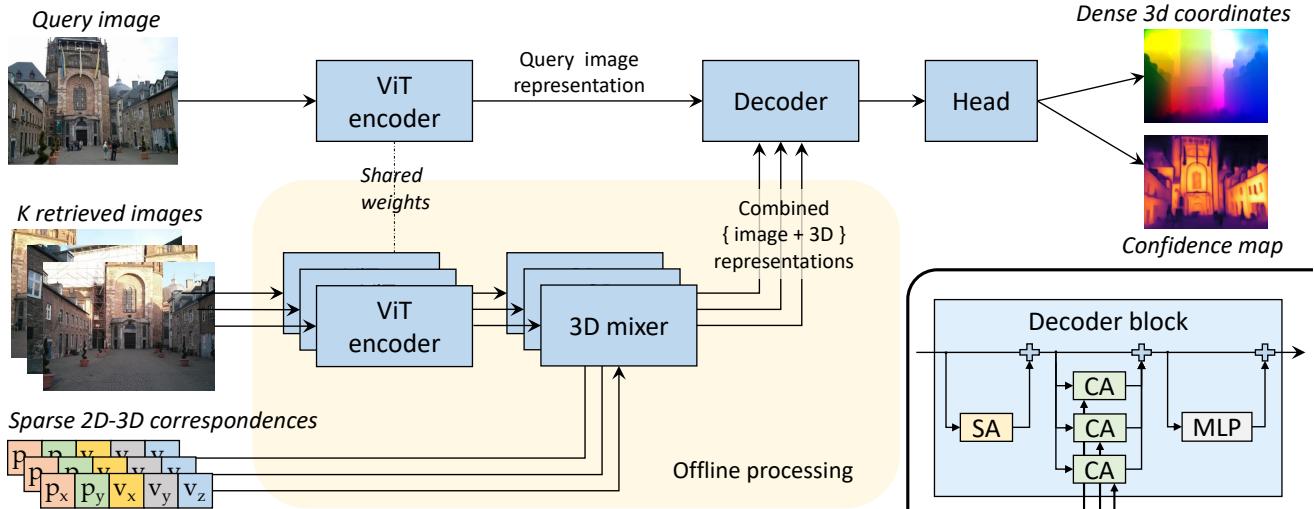


Figure 2: **Overview of the SACReg architecture.** The query and the  $K$  database images are first encoded with a vision transformer. For the database images, we additionally combine the encoded tokens with sparse 2D-3D correspondences obtained *e.g.* from SfM. A decoder then jointly processes tokens from the query image with the 3D-augmented tokens from database images and outputs dense 3D coordinates as well as a confidence map via the head. A decoder block, shown in the bottom-right corner, consists of self-attention (SA), multiple parallel cross-attention (CA) for each of the database image representations, and a MLP. Database images can be encoded and compressed offline for better test-time efficiency.

scriptors, and maintaining the overall localization pipeline is complex. Our approach in contrast does not require to store keypoints descriptors, and is arguably simpler.

### 3. The SACReg model

In this section, we first describe our proposed scene-agnostic coordinate regression model (Section 3.1), then describe in more detail our robust coordinate encoding and associated training loss (Section 3.2). We finally describe how the model can be applied to visual localization (Section 3.3) and present training details (Section 3.4).

#### 3.1. Model architecture

Our model takes as input a query image  $\mathcal{I}_q$ , as well as a set of  $K$  database images  $\{\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_K\}$ , for which some sparse 2D-3D annotations are available. We denote by  $\mathcal{V}_i = \{(\mathbf{p}_j^i, \mathbf{v}_j^i)\}$  the set of 2D-3D correspondences for the database image  $\mathcal{I}_i$ , *i.e.*,  $\mathbf{v}_j^i \in \mathbb{R}^3$  is a 3D point visible at pixel  $\mathbf{p}_j^i$  expressed in a world coordinate system. Our model predicts 3D coordinates for every pixel in the query image.

**Overview.** Figure 2 shows an overview of the model architecture. First, the query image is encoded into a set of token features with a Vision Transformer [18] (ViT) encoder. Similarly, the same encoder is used to encode each database image. For each database image, the information about the sparse 2D-3D coordinates are injected into the features using a transformer decoder, *i.e.*, a series of blocks, each composed of self-attention between the token features, cross-

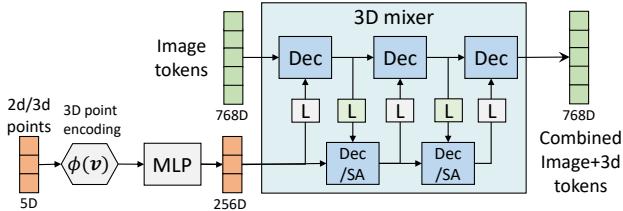
attention with the 3D coordinates, and a MLP. We refer to this module as *3D Mixer*. The next step consists in combining information from the query features with the database images and the associated 2D-3D coordinates. This is performed once again with a transformer decoder. Finally, a prediction head outputs dense 3D coordinates for each pixel of the query image. We now detail each module: the image encoder, the 3D mixer, the decoder and the prediction head.

**Image encoder.** We use a vision transformer [18] to encode all query and database images. In more details, each image is divided into non-overlapping patches, and a linear projection encodes them into patch features. A series of transformer blocks is then applied on these features: each block consists of multi-head self-attention and a MLP. In practice, we use a ViT-Base model, *i.e.*,  $16 \times 16$  patches with 768-dimensional features, 12 heads and 12 blocks. Following [77, 73], we use RoPE [59] relative position embeddings. As a result of the ViT encoding, we obtain sets of token features denoted  $\mathcal{R}_q$  for the query and  $\mathcal{R}_i$  for the database image  $\mathcal{I}_i$  respectively:

$$\mathcal{R}_q = \text{Encoder}(\mathcal{I}_q), \mathcal{R}_i = \text{Encoder}(\mathcal{I}_i), i = 1 \dots K. \quad (1)$$

**3D mixer.** This step consists in injecting information about the sparse 2D-3D correspondence set  $\mathcal{V}_i$  into the features  $\mathcal{R}_i$  that represent the corresponding database image  $\mathcal{I}_i$ . As output of the 3D mixer, we obtain augmented token features  $\mathcal{R}'_i$ :

$$\mathcal{R}'_i = \text{3Dmixer}(\mathcal{R}_i, \mathcal{V}_i). \quad (2)$$



**Figure 3: Overview of the 3D-mixer** which combines together the image with its associated 3D keypoints in a single representation. “Dec” stands for a classical transformer decoder block with SA, CA and MLP. “Dec /SA” is a decoder without self-attention to scale to large number of 3D points. The linear projection (“L”) enables to map between the dimensions of image and 3D point tokens (768 vs. 256).

As shown in Figure 3, we first embed each sparse 2D-3D correspondences in a 256-dimensional feature space with a MLP and call the result *point token*. To that aim, we encode the 3D coordinates using a cosine point encoding  $\phi$  described in Section 3.2 before feeding them to the MLP. We then use a series of transformer decoder blocks, where each block consists of a multi-head self-attention (SA) between the 768-dimensional image tokens, a multi-head cross-attention (CA) that injects information from all point tokens into these image tokens, and a MLP.

To improve robustness against potential outliers in the input sparse 3D points (due to *e.g.* triangulation noise), we interleave the image-level decoder blocks with a new set of point-level decoder blocks. We remove self-attention in these point-level decoder blocks to reduce their complexity. To account for the different features sizes between point tokens (256) and image tokens (768), we use linear projections that are shared across decoder blocks. In practice, we use 4 image-level decoder blocks interleaved with 3 point-level decoder blocks.

**Decoder.** The next step is to inject information from the database images and their sparse 3D coordinates, *i.e.*, from  $\mathcal{R}'_i$  into the query features  $\mathcal{R}_q$ . We propose once again to do that with a transformer decoder, *i.e.*, via cross-attention. Specifically, we apply multi-head cross-attentions for each  $\mathcal{R}'_i$  in parallel, followed by an averaging operation, see Fig. 2 (bottom right). Such approach generalizes smoothly from the single-image to the multi-image case.

**Prediction head.** We reshape the token features from the last transformer decoder block into a dense feature map and apply a convolutional head. Specifically, we first project the features to 1024 dimensions using 1x1 convolution, then apply a sequence of 6 ResNeXT blocks, with a PixelShuffle [54] operation every two blocks to increase the resolution while halving the channel dimension. For a  $224 \times 224$  input image, we get a  $14^2 \times 1024$  token map after the initial projection, which is gradually expanded to  $28^2 \times 512$ ,  $56^2 \times 256$  and finally  $224^2 \times d$  where  $d$  is the output dimension.

### 3.2. Generalization and training loss

**Output space.** A naive approach consists in setting  $d = 3$ , *i.e.*, trying to directly regress dense 3D points  $\{\hat{\mathbf{v}}\} \in \mathbb{R}^3$  from the regression head. This is possible and could be trained with a standard  $\ell_1$  or  $\ell_2$  regression loss, but is subject to a major limitation. At test time, the network is typically unable to regress coordinates outside the range seen during training. Thus, except for small scenes, it cannot generalize to new datasets (see Section 4.2).

Instead, we propose to regress a higher-dimensional 3D point encoding  $\phi(\mathbf{v}) \in (\mathbb{S}^1)^{d/2} \subset [-1, 1]^d$ , with  $d \gg 3$ . We design  $\phi$  with several desirable properties holding for any given  $\mathbf{v} \in \mathbb{R}^3$ : (i)  $\phi$  is an injective mapping, with an inverse projection  $\phi^{-1}$  such that  $\phi^{-1}(\phi(\mathbf{v})) = \mathbf{v}$ ; (ii) the input space of  $\phi^{-1}$  is the unit-circle product  $(\mathbb{S}^1)^{d/2} \subset [-1, 1]^d$ , whose high dimension enables error-correcting mechanisms in  $\phi^{-1}$ . Thanks to these properties, our method can handle any coordinate at test time.

**Point encoding** Assuming uncorrelated  $x$ ,  $y$  and  $z$  coordinates we can decompose  $\phi(\mathbf{v}) = [\psi(x), \psi(y), \psi(z)]$  and define  $\psi(x)$  as

$$\psi(x) = [\cos(f_1 x), \sin(f_1 x), \cos(f_2 x), \sin(f_2 x), \dots] \quad (3)$$

where the  $f_i$ 's are frequencies defined as  $f_i = f_0 \gamma^{i-1}$ ,  $i \in \{1, \dots, d/6\}$ , with  $f_0 > 0$  and  $\gamma > 1$ . In practice, we set  $f_0$  and  $\gamma$  such that the periods of the lowest and highest frequencies  $f_1$  and  $f_{d/6}$  approximately corresponds to the maximum scale of a query scene (*e.g.* 300 meters) and the desired spatial resolution (*e.g.* 0.5 meter). The encoding dimension  $d$  then becomes a parameter that controls the level of redundancy.  $d$  must be carefully chosen, as too small encodings are not noise-resistant enough, while too large encodings may demand too much capacity for the decoder. The inverse mapping  $\psi^{-1}$  efficiently solves a least-square problem of the form  $\psi^{-1}(y) = \text{argmin}_x \|y - \psi(x)\|^2$ , see Supplementary and [47].

**Regression loss.** As for the naive regression case, we apply a standard  $\ell_1$  regression loss to train the network. We have

$$\mathcal{L}_{\text{reg}}(\mathbf{v}, \hat{\mathbf{y}}) = |\phi(\mathbf{v}) - \hat{\mathbf{y}}|, \quad (4)$$

where  $\hat{\mathbf{y}} \in \mathbb{R}^d$  is the network output and  $\mathbf{v}$  is the corresponding ground-truth 3D point. We further exploit the correlation between pairs of adjacent values in  $\phi(\mathbf{v})$ , as we have  $\cos^2(f_i x) + \sin^2(f_i x) = 1$ . Before applying the  $\mathcal{L}_{\text{reg}}$  loss, we thus  $\ell_2$ -normalize each pairs of consecutive values in  $\hat{\mathbf{y}}$ . We find this helps the training significantly.

**Pixelwise confidence.** Some parts of the query image are inevitably harder, or even impossible to regress, such as the sky or objects not visible in database images. We therefore jointly predict a per-pixel confidence  $\tau > 0$  that modulates the regression loss (4), following [29]:

$$\mathcal{L}_{\text{SCR}}(\mathbf{v}, \hat{\mathbf{y}}, \tau) = \tau \mathcal{L}_{\text{reg}}(\mathbf{v}, \hat{\mathbf{y}}) - \log \tau. \quad (5)$$

$\tau$  can be interpreted as the confidence of the prediction: if  $\tau$  is low for a given pixel, the corresponding  $\mathcal{L}_{\text{reg}}$  loss at this location will be down-weighted. The second term of the loss incites the model to avoid being under-confident.

### 3.3. Application to visual localization

We now present how our model can be applied to predict the camera pose of a given query image from a small set of relevant database images with sparse 2D-3D point correspondences. An overview of our visual localization pipeline is shown in Figure 1.

**Image retrieval.** Given a query image, we first follow the same retrieval step than for standard feature-matching-based localization approaches [26, 49, 50]. Namely, we utilize off-the-shelf image retrieval methods such as HOW [65], AP-GeM [45] or FiRE [75] to obtain a shortlist of  $K$  relevant database images for a given query image.

**Sparse 2D-3D annotations.** Our approach takes as input sparse 2D-3D correspondences for each database image. To get them, we randomly subsample 2D points from the dense RGB-D data and reproject them in 3D using the known camera poses, when available. If not, we rely on standard Structure-from-Motion pipelines [53] during which 2D key-point matches between images are used to recover the corresponding 3D point locations and the camera poses. This process directly yields a set of 2D-3D correspondences for each database image. Unless specified otherwise, we use the output of COLMAP [53] with SIFT [37] keypoints.

**Predicting camera poses.** Given all images and their corresponding sparse 3D points, our model predicts a dense 3D coordinate map and a corresponding confidence map for the query image, as can be seen in Figure 2. We first mitigate the noise by sorting all predicted 3D points according to the confidence  $\tau$  and retaining the top 50%. We then use an off-the-shelf PnP solver to obtain the predicted camera pose. Specifically, we rely on SQ-PnP [63] with 4096 randomly sampled 2D-3D correspondences, 10,000 iterations and a reprojection error threshold of 5 pixels.

**Multi-image fusion strategies.** We experiment with two different strategies to handle multiple database images, each one offering a different quality-versus-runtime trade-off. The first strategy is to let the decoder take care of fusing all  $K$  database inputs itself, *i.e.* directly computing  $\text{Head}(\text{Decoder}(\mathcal{R}_q, \mathcal{R}'_1, \dots, \mathcal{R}'_K))$ , which is denoted as *early fusion* (EF) in the following. The second strategy is to feed database images one at a time to the decoder, gathering each time the dense prediction maps, *i.e.* computing  $\{\text{Head}(\text{Decoder}(\mathcal{R}_q, \mathcal{R}'_1)), \dots, \text{Head}(\text{Decoder}(\mathcal{R}_q, \mathcal{R}'_K))\}$ . Then, all resulting dense maps are fused by keeping, for each pixel, the most confident prediction. This is referred to as *late fusion* (LF) in the following.

On the one hand, EF is typically more difficult than LF, since the decoder has to deal with several images at the same

step	Res	#img	LR schedule	#epochs	Loss	Frozen
<b>I</b>	224	1+1	fixed lr 1e-4	150	$L_{\text{CroCo}}$	-
<b>II</b>	224	1+1	fixed lr 1e-4	100	$L_{\text{SCR}}$	E
<b>III</b>	512	1+1	cosine decay	20	$L_{\text{SCR}}$	-
<b>IV</b>	512	1+8	cosine decay	20	$L_{\text{SCR}}$	E,M

Table 1: **Training recipe** for our SCR model, with the image resolution ('Res'), number of query and retrieved images ('#img'), learning rate schedule, number of epochs and training loss. The rightmost column shows which modules are frozen during each step (E=ViT Encoder, M=3D Mixer).

time, including potential outliers, with the same network capacity. On the other hand, LF is much more costly, since both decoder and regression head must be run  $K$  times. Results in Section 4.3 show that, depending on the desired accuracy and computing budget, both might be interesting.

### 3.4. Training details

While training the full SCR model from scratch would be possible, we find it prohibitively slow and sub-optimal. We instead train our SCR model in several steps, which are detailed in Table 1, gradually increasing the image resolution and the number of retrieved images. We first initialize the weights of the ViT encoder and the decoder with cross-view completion pre-training (CroCo) [73, 74] (step I). From step II onward, we introduce the randomly-initialized 3D mixer and regression head. Using the SCR loss from Equation 5 and freezing the pre-trained encoder, we jointly train the 3D mixer, decoder and prediction head, again using pairs of  $224 \times 224$  images (a query and one single database image). During step III, we switch to a higher image resolution of  $512 \times 384$  and finetune all components, still using only one database image. We find it important to finetune the encoder when changing the input resolution. During step IV, we finally finetune the decoder and head (*i.e.* freezing both encoder and 3D mixer) using 8 database images.

**Data.** We train our model on datasets that cover various scenarios for robustness: MegaDepth [34] contains SfM reconstruction of 275 (mainly) outdoor scenes, ARKitScenes [3] consists of indoor house scenes, and Habitat of synthetic indoor scenes derived from the HM3D [44], ScanNet [14], Replica [58] and ReplicaCAD [60] rendered using Habitat-Sim [52]. These three datasets provide dense depth estimates as well as camera poses, thus allowing to train our SCR model in a fully-supervised manner. We use 100K query from each dataset (300K in total). For each query, we retrieve beforehand a shortlist of  $K$  similar images using FiRE [75].

**Data augmentation.** We apply standard random crop and color jitter during training. For robustness to possible triangulation noise, we augment 5% of the sparse 3D points with simulated depth noise. We also apply random geometric

Point encoding	Aug	Camb. ↓	7scenes ↓	Aachen-Night ↑
$(x, y, z) \in \mathbb{R}^3$		1.69	<b>0.11</b>	0.0 / 0.0 / 0.0
$(x, y, z) \in \mathbb{R}^3$	✓	14.43	2.89	0.0 / 2.1 / 44.5
$\phi(\cdot) \in [-1, 1]^{24}$	✓	0.47	<b>0.11</b>	22.0 / 46.6 / 89.5
$\phi(\cdot) \in [-1, 1]^{36}$	✓	<b>0.43</b>	<b>0.11</b>	22.0 / <b>47.1 / 90.6</b>
$\phi(\cdot) \in [-1, 1]^{48}$	✓	0.55	<b>0.11</b>	<b>23.6 / 40.8 / 87.4</b>

Table 2: **Ablation on 3D point encoding.**

3D transformation to scene coordinates for better generalization. Namely, we apply random 3D rotation followed by random scaling in the range  $[1/2, 2]$  and random translation in  $[-1000m, 1000m]^3$ .

## 4. Experiments

After presenting the test datasets (Section 4.1), we present several ablations in Section 4.2 and study fusion strategies in Section 4.3. We compare our approach to the state of the art in visual localization (Section 4.4), and finally evaluate the accuracy of the regressed coordinates (Section 4.5).

### 4.1. Datasets and metrics

**Cambridge-Landmarks** [30] consists of 6 outdoor scenes with RGB images from videos and small-scale landmarks.

**7 Scenes**[56] consists of 7 indoor scenes with RGB-D images from videos. Each scene has a limited size, and the images contain repeating structures, motion blur, and textureless surfaces. We do not use the depth data of the query image during inference.

**Aachen Day-Night v1.1**[51][82] contains 6,697 database images captured at day time, and 1015 query images including 824 taken during daytime (Aachen-Day) and 191 during nighttime (Aachen-Night).

**Metrics.** For Cambridge and 7-Scenes, we report the median translation error. For Aachen, we report the percentage of successfully localized images within three thresholds: (0.25m, 2°), (0.5m, 5°) and (5m, 10°).

### 4.2. Ablative study

**Validation sets and metrics.** For the ablations, we report the visual localization performance on a selected subset of 5 diverse and relatively challenging datasets: 7scenes-stairs, 7scenes-pumpkin, Cambridge-GreatCourt, Cambridge-OldHospital and Aachen-Night. For 7scenes and Cambridge-Landmarks, we report the averaged median translation error, while for Aachen-Night we report the localization accuracy for the 3 standard thresholds.

**Default settings.** We use the training recipe from Table 1 as default setting for the ablations, unless specified otherwise. For the sake of time, we train and test on small images, *i.e.* performing training step III with 224x224 images and

Frozen	Camb. ↓	7scenes ↓	Aachen-Night ↑
-	0.54	0.14	18.3 / 37.7 / 85.3
Encoder	<b>0.43</b>	<b>0.11</b>	<b>22.0 / 47.1 / 90.6</b>

Table 3: **Ablation on freezing the encoder during step II.**

3D-mixer	Camb. ↓	7scenes ↓	Aachen-Night ↑
Simple, depth=1	0.66	0.11	16.8 / 40.8 / 87.4
Simple, depth=2	0.61	<b>0.08</b>	20.4 / 42.4 / 88.0
Simple, depth=4	0.55	0.10	<b>22.0 / 39.3 / 89.5</b>
Alternating, depth=1	0.56	0.11	18.8 / 41.4 / 88.0
Alternating, depth=2	0.46	0.11	<b>22.0 / 46.1 / 88.5</b>
Alternating, depth=4	<b>0.43</b>	0.11	<b>22.0 / 47.1 / 90.6</b>

Table 4: **Ablation on 3D-mixer variants.**

completely skipping step IV (hence evaluating with the top-1 database image only). For each ablation table, we put a gray background color on the row with default settings.

**Robust coordinate encoding.** We first study in Table 2 the impact of different point encoding options. Direct coordinate regression is only successful when the train and test output distributions are aligned. This is the case for small scenes with zero-centered spaces like 7-scenes, or Cambridge to a lesser extent. For larger scenes with unconstrained coordinates (like Aachen), direct regression utterly fails. One way to mitigate this issue is to augment 3D coordinates at training time, using random translations for instance (see Section 3.4). Augmentations somehow improve the situation for Aachen-Night, but the performance overall strongly degrades for Cambridge and 7scenes. In contrast, the cosine-based encoding  $\phi$  proposed in Section 3.2 effectively deals with indoor and outdoor scenes in any coordinate ranges. For our model, we find optimal to use 6 frequencies, yielding  $d = 36$ -dimensional outputs.

**Pretraining and encoder freeze.** Table 3 shows that freezing the encoder during step II is highly beneficial, meaning that pretraining with CroCo (step I) effectively learns adequate image representations. This might be explained by the fact that CroCo essentially learns to compare and match images, and its training set is much larger than our finetuning data (7M versus 300K pairs).

**Architecture of the 3D mixer.** We experiment with different architectures for the 3D-mixer module and report results in Table 4. We compare the interleaved decoder architecture from Figure 3 with a simpler architecture composed of a sequence of several image-level decoder blocks ('Simple'). In all cases, increasing the number of decoder blocks leads to improved performance. However, we observe a substantial gain with the alternating decoder architecture, due to the enhanced denoising ability of this architecture (Section 3.1).

**Separate heads.** We experiment with different architectures for the regression head, this time aiming at exploiting priors of the output space. Recall that for each pixel,

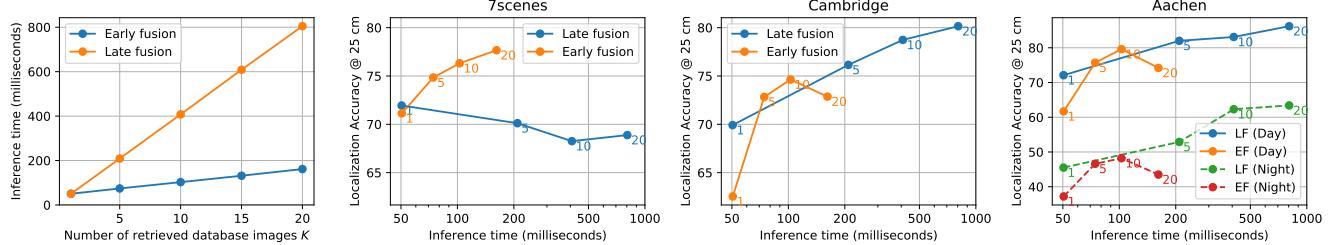


Figure 4: **Leftmost:** Inference time as a function of the shortlist size  $K$ . **Others:** Plots of the localization accuracy as a function of the inference time for different number of shortlist sizes  $K$  (indicated next to each point) for early and late fusion.

Regression head	Channels	Camb. ↓	7scenes ↓	Aachen-Night ↑
Linear	$x, y, z, \tau$	0.94	0.12	11.0 / 31.9 / 84.3
ConvNeXT	$xyz\tau$	0.64	<b>0.11</b>	19.9 / 39.8 / 88.0
ConvNeXT	$xyz, \tau$	0.61	<b>0.11</b>	15.7 / 41.4 / 87.4
ConvNeXT	$x, y, z, \tau$	<b>0.43</b>	<b>0.11</b>	<b>22.0 / 47.1 / 90.6</b>

Table 5: **Ablation on regression head.**

Train res.	Test res.	Camb. ↓	7scenes ↓	Aachen-Night ↑
224×224	224×224	0.43	0.11	22.0 / 47.1 / 90.6
512×384	512×384	0.21	0.10	39.3 / 63.4 / <b>94.8</b>
512×384	640×480	<b>0.20</b>	<b>0.07</b>	45.5 / 68.6 / <b>94.8</b>
512×384	768×512	0.24	<b>0.07</b>	45.5 / <b>70.2</b> / 93.7

Table 6: **Impact of training and test image resolution.**

we ultimately predict 4 values: 3 spatial components ( $x, y$  and  $z$ ) and a confidence  $\tau$ . *A priori*, these four components have no reason to be correlated. In fact, predicting them jointly could turn detrimental if there is a risk for the network to learn imaginary correlations. Therefore, we compare: (i) regressing the 4 components jointly using the same head, (ii) regressing the spatial and confidence components separately; (iii) regressing all 4 components separately, in which case we still use the same prediction head with shared weight for all spatial  $x, y$  and  $z$  components; (iv) as a baseline, a simple linear head, which is the same as 4 independent linear heads (one per component). From Table 5, it is clear that option (iii) yields the best performance, while the linear heads remains the worst option. This confirms that enforcing decorrelation between all components is strongly beneficial for generalization at test time.

**Image resolution** can have a strong impact on the test performance. Table 6 shows that test performance generally increases as a function of image resolution. Interestingly, the model is able to generalize to higher resolution at test time, as training in  $512 \times 384$  and testing on higher resolutions consistently yields better results. In the following, we always test on  $640 \times 480$  images.

### 4.3. Fusion strategies for multiple retrieved images

All experiments so far have been performed with a single database image using models trained until step III. We now compare the early and late fusion strategies (see Sec-

	Method	Aachen-Day ↑	Aachen-Night ↑
Kpts	Active Search [50]	57.3 / 83.7 / 96.6	28.6 / 37.8 / 51.0
	HLoc [49]	<b>89.6 / 95.4 / 98.8</b>	<b>86.7 / 93.9 / 100</b>
Learning-based	DSAC [6]	0.4 / 2.4 / 34.0	-
	ESAC (50 experts) [8]	42.6 / 59.6 / 75.5	-
	HSCNet [33]	71.1 / 81.9 / 91.7	40.8 / 56.1 / 76.5
	NeuMap [62]	76.2 / 88.5 / 95.5	37.8 / 62.2 / 87.8
SACReg, EF, $K = 8$	79.6 / 89.6 / <b>98.9</b>	51.3 / 75.4 / <b>96.9</b>	
SACReg, LF, $K = 20$	<b>86.2 / 93.8 / 99.6</b>	<b>63.4 / 86.9 / 100.0</b>	

Table 7: **Comparison to the state of the art on Aachen.**

tion 3.3) in terms of inference time and localization accuracy. Timings are measured on a A100 GPU and correspond to the query going through the encoder, decoder and regression head, *i.e.*, without the encoding of the database images and their coordinates, as this can be computed offline and saved. We average the localization accuracy for each dataset, except for Aachen where we display results for Aachen-Day and Aachen-Night separately.

On Cambridge and Aachen, we observe in Figure 4 that late fusion (LF) outperforms early fusion (EF), even though EF has benefited from more training epochs (*i.e.*, reaching step IV instead of stopping at step III). This is expected as LF is a simpler task for the decoder. On these datasets, EF performance starts deteriorating beyond 10 database images, which roughly corresponds to the point where most outliers start to appear in the retrieval shortlist. Since EF is trained with 8 database images, this may cause a discrepancy and hurt generalization, although results on 7-scenes show this is not always true. In all cases, LF is much slower than EF, and as a result, EF always manages to match or outperform late fusion for an equivalent run-time budget.

### 4.4. Visual localization benchmarking

We compare our approach to the state of the art for visual localization on indoor (7-scenes) and outdoor datasets (Cambridge-Landmarks, Aachen-DayNight). We compare to learning-based approaches as well as a few representative keypoint-based methods such as Active Search [50] and HLoc [49]. Results are presented in Table 7, Table 8 and Figure 6 respectively. Our late fusion model outperforms the early fusion one on Aachen-Day-Night and Cam-

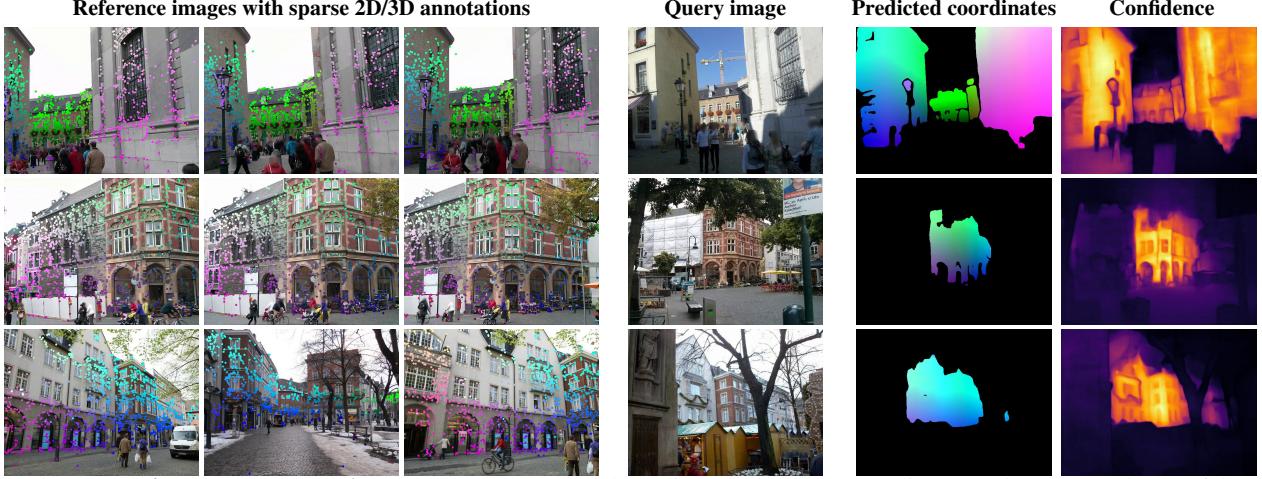


Figure 5: **Regression examples on Aachen-Day.** Our model predicts a dense 3D coordinates point map and a confidence map (5<sup>th</sup> and 6<sup>th</sup> columns) for a given query image (4<sup>th</sup> column) using reference images retrieved from a *SfM* database (1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup> columns). Only the first 3 reference images (out of  $K = 8$ ) are depicted. 3D coordinates and confidence are colorized for visualization purposes, and areas with a confidence below  $\tau = \exp(0)$  are not displayed. Best viewed in color.

		Chess ↓	Fire ↓	Heads ↓	Office ↓	Pumpkin ↓	Kitchen ↓	Stairs ↓
Kpts	Active search [50]	0.04, 1.96	0.03, 1.53	0.02, 1.45	0.09, 3.61	0.08, 3.10	0.07, 3.37	0.03, 2.22
	HLoc [49]	0.02, 0.79	0.02, 0.87	0.02, 0.92	0.03, 0.91	0.05, 1.12	0.04, 1.25	0.06, 1.62
Learning-based	RelocNet [2]	0.12, 4.14	0.26, 10.4	0.14, 10.5	0.18, 5.32	0.26, 4.17	0.23, 5.08	0.28, 7.53
	CamNet [16]	0.04, 1.73	0.03, 1.74	0.05, 1.98	0.04, 1.62	0.04, 1.64	0.04, 1.63	0.04, 1.51
	DSAC++ [7]	<u>0.02, 0.5</u>	0.02, 0.9	<b>0.01, 0.8</b>	<u>0.03, 0.7</u>	<u>0.04, 1.1</u>	0.04, 1.1	0.09, 2.6
	KFNet [83]	<u>0.02, 0.65</u>	0.02, 0.9	<b>0.01, 0.82</b>	<u>0.03, 0.69</u>	<u>0.04, 1.02</u>	<u>0.04, 1.16</u>	<u>0.03, 0.94</u>
	HSCNet [33]	<u>0.02, 0.7</u>	0.02, 0.9	<b>0.01, 0.9</b>	<u>0.03, 0.8</u>	<u>0.04, 1.0</u>	0.04, 1.2	<u>0.03, 0.8</u>
	SANet [78]	0.03, 0.88	0.03, 1.12	0.02, 1.48	<u>0.03, 1.00</u>	0.04, 1.21	0.04, 1.40	0.16, 4.59
	DSM [61]	0.02, 0.68	0.02, 0.80	<b>0.01, 0.8</b>	<u>0.03, 0.78</u>	<u>0.04, 1.11</u>	<b>0.03, 1.11</b>	0.04, 1.16
	SC-wLS[76]	0.03, 0.76	0.05, 1.09	0.03, 1.92	0.06, 0.86	0.08, 1.27	0.09, 1.43	0.12, 2.80
	NeuMaps [62]	0.02, 0.81	0.03, 1.11	0.02, 1.17	0.03, 0.98	0.04, 1.11	0.04, 1.33	0.04, 1.12
	SACReg, EF, K=8	0.03, 0.89	0.03, 1.15	<u>0.02, 1.07</u>	<u>0.03, 1.08</u>	0.05, 1.32	0.04, 1.36	0.10, 2.45
	SACReg, LF, K=20	0.03, 1.03	0.04, 1.52	<u>0.02, 1.18</u>	<u>0.03, 1.47</u>	0.07, 1.78	0.06, 1.84	0.07, 1.67

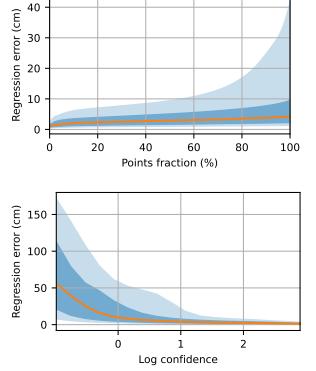


Figure 6: **Evaluations on 7-Scenes.** **Left: visual localization.** Comparison with the state of the art in terms of median translation (m) and angular (°) errors. **Right: coordinates regression.** Error distributions (first and last quartiles, deciles, and median) for relative and absolute confidence.

		ShopFacade ↓	OldHospital ↓	College ↓	Church ↓	Court ↓
Kpts	Active search [50]	0.12, 1.12	0.52, 1.12	0.57, 0.70	0.22, 0.62	1.20, 0.60
	HLoc [49]	<b>0.04, 0.20</b>	<u>0.15, 0.3</u>	<b>0.12, 0.20</b>	<u>0.07, 0.21</u>	<u>0.11, 0.16</u>
Learning-based	DSAC++ [7]	0.06, 0.3	0.20, <u>0.3</u>	0.18, 0.3	0.13, 0.4	0.20, 0.4
	DSAC* [9]	<u>0.05, 0.3</u>	0.21, 0.4	0.15, 0.3	0.13, 0.4	0.49, 0.3
	KFNet [83]	<u>0.05, 0.35</u>	<u>0.18, 0.28</u>	0.16, 0.27	0.12, 0.35	0.42, 0.21
	HSCNet [33]	0.06, 0.3	0.19, 0.3	0.18, 0.3	0.09, 0.3	0.28, 0.2
	SANet [78]	0.1, 0.47	0.32, 0.53	0.32, 0.54	0.16, 0.57	3.28, 1.95
	DSM [61]	0.06, 0.3	0.23, 0.38	0.19, 0.35	0.11, 0.34	0.19, 0.43
	SC-wLS [76]	0.11, 0.7	0.42, 1.7	<b>0.14, 0.6</b>	0.39, 1.3	1.64, 0.9
	NeuMaps [62]	0.06, <u>0.25</u>	0.19, 0.36	<u>0.14, 0.19</u>	0.17, 0.53	<b>0.06, 0.1</b>
	SACReg, EF, K=8	0.06, 0.32	0.18, 0.32	<u>0.14, 0.25</u>	<u>0.07, 0.23</u>	0.18, 0.12
	SACReg, LF, K=20	<u>0.05, 0.3</u>	<u>0.13, 0.29</u>	<u>0.12, 0.18</u>	<u>0.06, 0.21</u>	0.13, 0.09

Table 8: **Comparison to the state of the art on Cambridge** with the median trans. (m) and angular (°) errors.

bridge and perform on par on 7-Scenes. On the indoor 7-Scenes, our method obtains similar or slightly worse performance compared to other approaches, but overall still performs well with a median error of a few centimeters. On outdoor datasets, the proposed methods strongly outperforms other learning-based methods, in particular other

scene-specific or scene-agnostic coordinate regression approaches like [6, 9, 76, 78, 61, 62, 8]. Our method even sets a new state of the art on the Cambridge dataset. This is remarkable because, in contrast to any other learning-based approaches, SACReg is directly applied to each test dataset without any finetuning. In other words, the proposed approach works out of the box on test data that were never seen during training. Interestingly, it even reaches the performance of keypoints-based approaches such as Active Search [50] or HLoc [49].

#### 4.5. Scene coordinates regression

Lastly, to evaluate the regression performances of SACReg, we apply our model on 7-Scenes, which provides dense ground truth annotations. Using a shortlist size of  $K = 8$ , we predict the 3D coordinates and corresponding confidence for each pixel of the test images. We obtain a

median and mean error of 4.2cm and 13.2cm respectively. Results furthermore validate that confidence predictions are meaningful, as errors tends to get smaller when the confidence increases (Fig. 6, top right). Confidence can thus be used as a proxy to filter out regions where errors are likely to be large (Fig. 6, bottom right, and black regions in Fig. 5).

## References

- [1] Map-free Visual Relocalization: Metric Pose Relative to a Single Image. In *ECCV*, 2022.
- [2] Vassileios Balntas, Shuda Li, and Victor Prisacariu. RelocNet: Continuous Metric Learning Relocalisation Using Neural Nets. In *ECCV*, 2018.
- [3] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In *NeurIPS*, 2021.
- [4] Hunter Blanton, Connor Greenwell, Scott Workman, and Nathan Jacobs. Extending absolute pose regression to multiple scenes. In *CVPRW*, 2020.
- [5] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6D Object Pose Estimation Using 3D Object Coordinates. In *ECCV*, 2014.
- [6] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC — Differentiable RANSAC for Camera Localization. In *CVPR*, 2017.
- [7] Eric Brachmann and Carsten Rother. Learning Less is More - 6D Camera Localization via 3D Surface Regression. In *CVPR*, 2018.
- [8] Eric Brachmann and Carsten Rother. Expert Sample Consensus Applied to Camera Re-Localization. In *ICCV*, 2019.
- [9] Eric Brachmann and Carsten Rother. Visual camera re-localization from RGB and RGB-D images using DSAC. *IEEE Trans. PAMI*, 2021.
- [10] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-Aware Learning of Maps for Camera Localization. In *CVPR*, 2018.
- [11] Wentao Cheng, Weisi Lin, Kan Chen, and Xinfeng Zhang. Cascaded parallel filtering for memory-efficient image-based localization. In *ICCV*, 2019.
- [12] Xinjing Cheng, Peng Wang, Chenye Guan, and Ruigang Yang. Cspn++: Learning context and resource aware convolutional spatial propagation networks for depth completion. In *AAAI*, 2020.
- [13] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 2018.
- [14] 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.
- [15] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabivovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018.
- [16] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. CamNet: Coarse-to-Fine Retrieval for Camera Re-Localization. In *ICCV*, 2019.
- [17] Siyan Dong, Shuzhe Wang, Yixin Zhuang, Juho Kannala, Marc Pollefeys, and Baoquan Chen. Visual localization via few-shot scene region classification. *arXiv preprint arXiv:2208.06933*, 2022.
- [18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
- [19] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. *arXiv preprint arXiv:1905.03561*, 2019.
- [20] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 1981.
- [21] Robert M. Gray and David L. Neuhoff. Quantization. *IEEE Trans. Inf. Theory*, 1998.
- [22] Abner Guzman-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew Fitzgibbon, and Shahram Izadi. Multi-output learning for camera relocalization. In *CVPR*, 2014.
- [23] 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.
- [24] Mu Hu, Shuling Wang, Bin Li, Shiyu Ning, Li Fan, and Xiaojin Gong. Penet: Towards precise and efficient image guided depth completion. In *ICRA*, 2021.
- [25] Zhaoyang Huang, Han Zhou, Yijin Li, Bangbang Yang, Yan Xu, Xiaowei Zhou, Hujun Bao, Guofeng Zhang, and Hongsheng Li. VS-Net: Voting with segmentation for visual localization. In *CVPR*, 2021.
- [26] 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.
- [27] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. PAMI*, 2011.
- [28] Alex Kendall and Roberto Cipolla. Geometric Loss Functions for Camera Pose Regression with Deep Learning. In *CVPR*, 2017.
- [29] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018.
- [30] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: a Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, 2015.
- [31] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *IJCV*, 2009.

- [32] Kunhong Li, Longguang Wang, Li Liu, Qing Ran, Kai Xu, and Yulan Guo. Decoupling makes weakly supervised local feature better. In *CVPR*, 2022.
- [33] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *CVPR*, 2020.
- [34] Zhengqi Li and Noah Snavely. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. In *CVPR*, 2018.
- [35] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *CVPR*, 2022.
- [36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- [37] David G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.
- [38] Zixin Luo, Tianwei Shen, Lei Zhou, Siyu Zhu, Runze Zhang, Yao Yao, Tian Fang, and Long Quan. Geodesc: Learning local descriptors by integrating geometry constraints. In *ECCV*, 2018.
- [39] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. In *CVPR*, 2020.
- [40] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967.
- [41] Danish Nazir, Alain Pagani, Marcus Liwicki, Didier Stricker, and Muhammad Zeshan Afzal. Sematnet: Toward attention-based semantic aware guided depth completion. *IEEE Access*, 2022.
- [42] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: Learning local features from images. In *NeurIPS*, 2018.
- [43] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *ECCV*, 2020.
- [44] 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.
- [45] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, 2019.
- [46] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2D2: Reliable and repeatable detector and descriptor. In *NeurIPS*, 2019.
- [47] Jérôme Revaud, Guillaume Lavoué, and Atilla Baskurt. Improving zernike moments comparison for optimal similarity and rotation angle retrieval. *IEEE trans. PAMI*, 2008.
- [48] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, 2011.
- [49] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019.
- [50] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE trans. PAMI*, 2017.
- [51] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, et al. Benchmarking 6dof outdoor visual localization in changing conditions. In *CVPR*, 2018.
- [52] 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.
- [53] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion Revisited. In *CVPR*, 2016.
- [54] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.
- [55] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *CVPR*, 2013.
- [56] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *CVPR*, 2013.
- [57] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the World from Internet Photo Collections. *IJCV*, 2008.
- [58] 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.
- [59] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.
- [60] 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.
- [61] Shitao Tang, Chengzhou Tang, Rui Huang, Siyu Zhu, and Ping Tan. Learning Camera Localization via Dense Scene Matching. In *CVPR*, 2021.

- [62] Shitao Tang, Sicong Tang, Andrea Tagliasacchi, Ping Tan, and Yasutaka Furukawa. Neumap: Neural coordinate mapping by auto-transdecoder for camera localization. *arXiv preprint arXiv:2211.11177*, 2022.
- [63] George Terzakis and Manolis Lourakis. A consistently fast and globally optimal solution to the perspective-n-point problem. In *ECCV*, 2020.
- [64] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Balntas. Sosnet: Second order similarity regularization for local descriptor learning. In *CVPR*, 2019.
- [65] Giorgos Tolias, Tomas Jenicek, and Ondřej Chum. Learning and aggregating deep local descriptors for instance-level recognition. In *ECCV*, 2020.
- [66] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. In *NeurIPS*, 2020.
- [67] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015.
- [68] Wouter Van Gansbeke, Davy Neven, Bert De Brabandere, and Luc Van Gool. Sparse and noisy lidar completion with RGB guidance and uncertainty. In *ICMVA*, 2019.
- [69] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.
- [70] Florian Walch, Caner Hazirbas, Laura Leal-Taixé, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization with spatial lstms. In *ICCV*, 2017.
- [71] Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. AtLoc: Attention Guided Camera Localization. In *AAAI*, 2020.
- [72] Qianqian Wang, Xiaowei Zhou, Bharath Hariharan, and Noah Snavely. Learning feature descriptors using camera pose supervision. In *ECCV*, 2020.
- [73] Philippe Weinzaepfel, Vaibhav Arora, Yohann Cabon, Thomas Lucas, Romain Brégier, Vincent Leroy, Gabriela Csurka, Leonid Antsfeld, Boris Chidlovskii, and Jérôme Revaud. Improved cross-view completion pre-training for stereo matching. *arXiv preprint arXiv:2211.10408*, 2022.
- [74] Philippe Weinzaepfel, Vincent Leroy, Thomas Lucas, Romain Brégier, Yohann Cabon, Vaibhav Arora, Leonid Antsfeld, Boris Chidlovskii, Gabriela Csurka, and Jérôme Revaud. CroCo: Self-Supervised Pre-training for 3D Vision Tasks by Cross-View Completion. In *NeurIPS*, 2022.
- [75] Philippe Weinzaepfel, Thomas Lucas, Diane Larlus, and Yannis Kalantidis. Learning super-features for image retrieval. In *ICLR*, 2022.
- [76] Xin Wu, Hao Zhao, Shunkai Li, Yingdian Cao, and Hongbin Zha. Sc-wls: Towards interpretable feed-forward camera relocalization. In *ECCV*, 2022.
- [77] Tao Xie, Kun Dai, Ke Wang, Ruifeng Li, and Lijun Zhao. Deepmatcher: A deep transformer-based network for robust and accurate local feature matching. *arXiv preprint arXiv:2301.02993*, 2023.
- [78] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. SANet: Scene Agnostic Network for Camera Localization. In *ICCV*, 2019.
- [79] Luwei Yang, Rakesh Shrestha, Wenbo Li, Shuaicheng Liu, Guofeng Zhang, Zhaopeng Cui, and Ping Tan. Sceenesqueezer: Learning to compress scene for camera relocalization. In *CVPR*, 2022.
- [80] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *ECCV*, 2016.
- [81] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. DPOD: 6D Pose Object Detector and Refiner. In *ICCV*, 2019.
- [82] Zichao Zhang, Torsten Sattler, and Davide Scaramuzza. Reference Pose Generation for Long-term Visual Localization via Learned Features and View Synthesis. *IJCV*, 2020.
- [83] Lei Zhou, Zixin Luo, Tianwei Shen, Jiahui Zhang, Mingmin Zhen, Yao Yao, Tian Fang, and Long Quan. Kfnet: Learning temporal camera relocalization using kalman filtering. In *CVPR*, 2020.
- [84] Qunjie Zhou, Torsten Sattler, Marc Pollefeys, and Laura Leal-Taixe. To Learn or Not to Learn: Visual Localization from Essential Matrices. In *ICRA*, 2020.

# Appendix

In this appendix, we first present additional qualitative results in Section A. We then elaborate on the offline computation, compression and storage of database images in Section B. Afterwards, we give detailed explanations and proofs about the injectivity of the point encoding and its inverse mapping in Section C. Finally, we provide details about online localization timings in Section D, the architecture of the regression head in Section E and the hyper-parameters used to train the model in Section F.

## A. More qualitative results

We present in Figure 7 regression outputs (dense 3D coordinates and confidence map) with their corresponding input images. From these outputs, we can notice several things. First, the model can successfully densify sparse inputs incoming from several images while respecting small structures like windows (*e.g.* in row #4). Second, the model is robust to occlusions, as can be especially seen in rows #2 and #4 where occluded parts in the query w.r.t. database images are also rated as unreliable in the output confidence map. Third, the model is typically not confident to extrapolate 3D predictions in query regions for which few or no 2D-3D annotations are available in the corresponding database images, such as the ground region, people or trees.

**3D reconstruction.** The 3D coordinates predicted by SACReg can be used to lift a query image into a 3D colored point-cloud, as we illustrate in the *supplementary video*. Point clouds corresponding to different query images of the same scene can then be merged into a single large scale 3D reconstruction, as illustrated in Fig. 8. In this case, we collected the output point clouds obtained for each query image from the Aachen-Day [51] dataset, removed low-confidence 3D points and simply concatenated all 3D points together before rendering the result using Blender [13]. Note that, because our approach directly predicts 3D coordinates in a metric world coordinate system, no further post-processing of the output is necessary to achieve these reconstructions.

## B. Database compression

One potential limitation of the proposed method might be the large volume of the pre-computed database image representations, if stored uncompressed. Indeed, considering an input resolution  $HW \triangleq 640 \times 480$ , and a ViT patch size of  $p \triangleq 16$ , an image representation typically consists of  $HW/p^2 = 1200$  token features of  $D \triangleq 768$  dimensions, each encoded using a 4-bytes floating point value. It thus leads to a memory use of  $4DHW/p^2 \approx 3.68\text{MB}$  per image. We show that these representations can be significantly compressed with very little loss of performance using product quantization.

**Product quantization (PQ)** [27] is a lossy data compression technique that works as follows. Given a vector  $\mathbf{x} \in \mathbb{R}^D$  to compress,  $\mathbf{x}$  is first split into  $N$  sub-vectors denoted as *blocks*  $\mathbf{x}_j \in \mathbb{R}^{D/N}$ ,  $j = 1 \dots N$  with  $N$  multiple of  $D$  and the block size being  $B = D/N$ . Then, each block  $\mathbf{x}_j$  is separately vector-quantized [21] using a codebook  $\mathcal{C}_j$ . Using codebooks with 256 entries, we can represent each block using a single byte index, effectively decreasing the memory requirement by a factor of  $4B$  (*i.e.* assuming 4 bytes per floating point value). In our specific case, given a 3D-augmented database image representation  $\mathcal{R}' = \{\mathbf{x}_i\}$  composed of token features  $\mathbf{x}_i \in \mathbb{R}^D$ , we split each token features into  $N$  blocks  $\mathbf{x}_{i,j} \in \mathbb{R}^B$  and replace each by the closest entry from codebook  $\mathcal{C}_j$ .

**Codebook learning.** To learn the set of codebooks  $\{\mathcal{C}_j\}$ , we first select a set of 30,000 database images  $\{\mathcal{I}_k\}$  from our universal training set and equally sampled from the *Habitat*, *ARKitScenes* and *MegaDepth* datasets. We encode them, along with their associated 2D-3D annotations, and get their 3D-augmented representation  $\{\mathcal{R}'_k\}$ , each of which being a set of 1200 token features for a resolution of  $640 \times 480$ . Gathering tokens from all images results in a set of  $M$  features vectors  $\mathbf{x}_i \in \mathbb{R}^D$ ,  $i = 1, \dots, M$ . We split each vectors into  $N$  blocks  $(\mathbf{x}_{i,1}, \dots, \mathbf{x}_{i,N})$  of dimensions  $B$ , and we cluster features  $(\mathbf{x}_{i,j})_{i=1, \dots, M}$  for each block  $j \in \{1, \dots, N\}$  into  $k = 256$  centroids using  $k$ -means [40] to form the codebook  $\mathcal{C}_j$ .

**Results.** During an offline phase, we compute, compress and store the representations of all database images. At test time, we reconstruct the full token features from the stored codebook indices and the corresponding codebooks. Localization results after quantization are reported in Tab. 9. For 7-Scenes and Cambridge-Landmarks, we report the average median translation error over the same challenging subsets than used in the ablative experiments from the main paper (*i.e.*, street+pumpkin for 7-Scenes and great-court+oldhospital for Cambridge). We observe that a quantization with a block size  $B = 2$  has a limited impact on localization accuracy, but that performance quickly degrades when using more aggressive schemes.

To alleviate the performance drop due to quantization, we slightly finetune the model for one additional epoch using compressed database features as inputs (considered as fixed), with a learning rate of  $10^{-4}$  and a cosine-decay scheduler. We observe large performance improvements thanks to this fine-tuning. In particular, we observe minimal performance drops compared to the original model when using quantization with blocks of size up to  $B = 8$ , which leads to a database compression factor of 32. Beyond this point, the performance gracefully degrades, such that for  $B = 32$  (compression factor of 128) our method is still able to obtain more than 80% accuracy at 50cm&5° error threshold on Aachen-Night.



**Figure 7: Regression examples on Aachen-Day.** Our model predicts a dense 3D coordinates point map and a confidence map (5<sup>th</sup> and 6<sup>th</sup> columns) for a given query image (4<sup>th</sup> column) using reference images retrieved from a *SfM* database (1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup> columns). Only the first 3 reference images (out of  $K = 8$ ) are depicted. 3D coordinates and confidence are colorized for visualization purposes, and areas with a confidence below  $\tau = 1$  are not displayed. Best viewed in color.

**Comparison with NeuMap.** In Table 10, we compare our approach with other scene-compression methods such as NeuMap [62], which directly regress the 3D coordinates of a given set of 2D keypoints using learned neural codes, and other scene compression methods such as Cascaded [11] and Squeezer [79], which are based on feature matching. Our approach achieves similar or better results compared to all other methods under similar compression settings (*e.g.* with  $B = 32$ ). Compared to NeuMap, which uses a similar learning- and transformer-based approach, our method’s inference is also much faster, because NeuMap’s decoder must be run multiple times with the retrieved voxels’ codes, which takes several seconds. Detailed timings of our model

are described in Section D. Additionally, it is noteworthy that, unlike NeuMap, we did not train our model on the Aachen dataset at all.

### C. Point encoding

We provide in this section further information regarding the point encoding  $\phi$  presented in the paper. First recall that we have defined  $\phi(v) = [\psi(x), \psi(y), \psi(z)]$  for  $v = (x, y, z) \in \mathbb{R}^3$ , *i.e.* simply concatenating the encodings of the  $x$ ,  $y$  and  $z$  channels, by a mapping  $\psi$ . For the sake of brevity, most of the discussion will thus focus on this



Figure 8: **Large-scale 3D reconstruction.** Dense point clouds regressed for different query images can be merged into a globally consistent 3D point cloud of a scene at scale. Example for query images from *Aachen-Day* (see also the supplementary video).

Quantization	FT	kB/img ↓	Camb. ↓	7scenes ↓	Aachen-Day ↑	Aachen-Night ↑
Raw		3686	0.13	0.07	86.2 / 93.8 / 99.6	63.4 / 86.9 / 100.0
PQ (B=2)		461	0.13	0.07	84.5 / 93.8 / 99.6	63.9 / 86.9 / 100.0
PQ (B=4)		230	0.14	0.07	84.6 / 93.9 / 99.6	58.6 / 82.2 / 100.0
PQ (B=6)		154	0.15	0.07	82.5 / 92.1 / 99.6	56.0 / 82.7 / 99.5
PQ (B=8)		115	0.18	0.08	80.6 / 91.4 / 99.6	46.1 / 77.0 / 98.4
PQ (B=16)		58	0.30	0.09	48.9 / 74.3 / 96.7	8.4 / 24.6 / 67.0
PQ (B=32)		29	24.3	1.67	0.4 / 1.2 / 8.7	0.0 / 0.0 / 3.1
PQ (B=64)		14	45.1	2.06	0.0 / 0.0 / 0.1	0.0 / 0.0 / 0.0
PQ (B=128)		7	53.8	2.23	0.0 / 0.0 / 0.0	0.0 / 0.0 / 0.0
PQ (B=2)	✓	461	0.13	0.07	85.1 / 93.3 / 99.5	61.8 / 85.9 / 100.0
PQ (B=4)	✓	230	0.13	0.07	84.5 / 93.0 / 99.5	63.9 / 89.0 / 100.0
PQ (B=6)	✓	154	0.13	0.07	84.5 / 93.7 / 99.5	63.9 / 87.4 / 99.5
PQ (B=8)	✓	115	0.14	0.07	83.5 / 92.8 / 99.4	63.4 / 85.9 / 100.0
PQ (B=16)	✓	58	0.15	0.07	81.3 / 91.5 / 99.3	59.7 / 84.3 / 100.0
PQ (B=32)	✓	29	0.20	0.08	75.6 / 89.6 / 99.5	56.5 / 81.2 / 100.0
PQ (B=64)	✓	14	0.34	0.08	60.4 / 82.8 / 98.8	40.3 / 65.4 / 97.9
PQ (B=128)	✓	7	2.75	0.14	24.9 / 51.3 / 91.1	17.3 / 36.1 / 90.1

Table 9: **Product quantization (PQ)** [27] enables to compress database tokens with minimal performance losses. Results obtained for  $640 \times 480$  resolution images, with late fusion (LF) and  $K=20$  retrieved images, for different block sizes  $B$ . The compression factor with respect to storing raw representations is  $4B$ . kB/img represents the number of kilobytes required to store the compressed representations of a single database image. FT denotes one more epoch of fine-tuning using the fixed compressed database representations.

mapping:

$$\begin{aligned} \psi : t \in \mathbb{R} &\rightarrow [\cos(f_1 x), \sin(f_1 x), \dots, \cos(f_F x), \sin(f_F x)] \\ &\in (\mathbb{S}^1)^F \subset [-1, 1]^{2F}, \end{aligned} \quad (6)$$

defined for some positive frequencies  $\{f_i\}_{i=1\dots F}$ , with  $F = d/6 \in \mathbb{N}^*$ , where  $d$  is the dimension of the output point encoding  $\phi(\mathbf{v})$ . We furthermore typically consider harmoni-

cally distributed frequencies  $f_i = f_1 \gamma^{i-1}$  for  $i = 1 \dots F$ , where  $\gamma > 1$ .

### C.1. Injectivity / periodicity

**Lemma 1.**  $\gamma \in \mathbb{R} \setminus \mathbb{Q} \Rightarrow \psi(x)$  is injective, or equivalently, if  $\gamma$  is irrational then there is no  $x$  and  $y$  with  $x \neq y$  such that  $\psi(x) = \psi(y)$ .

**Proof by contraposition.** We will prove that  $\psi(x) =$

Method	Size ↓	Aachen-Day ↑	Aachen-Night ↑
NeuMap (8m, 100) [62]	1.26	80.8 / 90.9 / 95.6	48.0 / 67.3 / 87.8
<b>SACReg PQ (B=6)</b>	0.96	<b>84.5 / 93.7 / 99.5</b>	<b>63.9 / 87.4 / 99.5</b>
SACReg PQ (B=16)	0.36	<b>81.3 / 91.5 / 99.3</b>	<b>59.7 / 84.3 / 100.0</b>
Squeezer [79]	0.24	75.5 / 89.7 / 96.2	50.0 / 67.3 / 78.6
SACReg PQ (B=32)	0.18	75.6 / <b>89.6 / 99.5</b>	<b>56.5 / 81.2 / 100.0</b>
NeuMap (10m, *) [62]	0.17	76.2 / 88.5 / 95.5	37.8 / 62.2 / 87.8
Cascaded [11]	0.14	<b>76.7 / 88.6 / 95.8</b>	33.7 / 48.0 / 62.2
<b>SACReg PQ (B=64)</b>	0.09	<b>60.4 / 82.8 / 98.8</b>	<b>40.3 / 65.4 / 97.9</b>

Table 10: **Results with compression compared to the state of the art on Aachen Day-Night.** Data sizes in the ‘Size’ column are reported in Gigabytes. We highlight in **bold** optimal values lying on an accuracy-versus-compression Pareto front.

$\psi(y)$  with  $x \neq y \Rightarrow \gamma \in \mathbb{Q}$ . For simplicity and without loss of generality, assume  $F = 2$  frequencies. Let  $x, y \in \mathbb{R}$  two distinct numbers ( $x \neq y$ ). The proposition  $\psi(x) = \psi(y)$  is equivalent to:

$$\begin{cases} \cos(f_1 x) = \cos(f_1 y) \\ \sin(f_1 x) = \sin(f_1 y) \\ \cos(f_2 x) = \cos(f_2 y) \\ \sin(f_2 x) = \sin(f_2 y), \end{cases} \quad (7)$$

which can be rewritten as:

$$\begin{cases} e^{if_1 x} = e^{if_1 y} \\ e^{if_2 x} = e^{if_2 y}, \end{cases} \quad (8)$$

i.e., that there exist  $k_1, k_2 \in \mathbb{Z}^*$  such that:

$$\begin{cases} f_1 x = f_1 y + 2\pi k_1 \\ f_2 x = f_2 y + 2\pi k_2. \end{cases} \quad (9)$$

Rearranging the equations, we get:

$$\begin{cases} \frac{f_1}{2\pi}(x - y) = k_1 \\ \frac{f_2}{2\pi}(x - y) = k_2. \end{cases} \quad (10)$$

Dividing the second term by the first one, which is possible as  $k_1, k_2 \neq 0$ , yields:

$$\frac{f_2}{f_1} = \gamma = \frac{k_2}{k_1} \in \mathbb{Q} \quad \blacksquare \quad (11)$$

While any irrational  $\gamma$  fulfills the injectivity lemma, we note that algebraic numbers should ideally be avoided. Indeed they introduce a risk that  $\psi$  would be periodic over a subset of dimensions  $\xi \subseteq \{1, \dots, F\}$ , with  $|\xi| > 1$ . Take, for instance, an algebraic  $\gamma = \sqrt{2}$  and assume  $F = 3$  frequencies. Then we have  $f_2 = \sqrt{2}f_1$  and  $f_3 = 2f_1$ . In this case, one can trivially show that  $\psi_\xi = [\psi_i]_{i \in \xi}$  is periodic

for  $\xi = \{1, 3\}$ . In the presence of noise, this can severely impair the robustness of the inverse mapping. Hence, in general, we recommend choosing transcendental values for  $\gamma$ , which is not really a problem as *almost all real* numbers are transcendental, i.e.,  $p(X \text{ is transcendental}) = 1$  for a uniformly random real  $X$  in any given range  $[a, b]$  with  $a < b$ .

**Numerical considerations.** Modern representations of real numbers on computers are limited by hardware constraints. In practice, a real number  $x \in \mathbb{R}$  is stored as a fixed-length bit vector (64 bits in our case), which roughly amounts to storing  $x$  as a rational  $\frac{a}{b}$ , with  $a, b \in \mathbb{Z}^*$ . Let then denote all frequencies as rational numbers  $f_i = \frac{a_i}{b_i}$ , with coprime  $a_i, b_i \in \mathbb{N}^*$ . In this case,  $\psi$  is periodic, with a period  $P \neq 0$  multiple of all individual periods  $2\pi/f_i$ , for  $i = 1, \dots, F$ . Equivalently, there exist some non-zero integers  $k_i \in \mathbb{N}^*$  such that:

$$P = k_i \frac{2\pi}{f_i}, \quad \forall i = 1 \dots F \quad (12)$$

i.e.:

$$P = 2\pi k_i \frac{b_i}{a_i}, \quad \forall i = 1 \dots F. \quad (13)$$

The smallest period  $P$  satisfying all these  $F$  equations is thus  $P = 2\pi \text{lcm}\{b_i\}/\text{gcd}\{a_i\}$ , defined by the least common multiple of all denominators (`lcm`) divided by the greatest common divisor (`gcd`) of all numerators. Such ratio is in general hard to predict for arbitrary frequencies  $\{f_i\}$ , but it can be extremely large when considering irreducible fractions of large terms  $a_i, b_i \simeq 10^{15}$ . For our use-case, we want the period to be a large as possible, and in practice, using double floating-point representations, we can find values for  $\gamma$  yielding periods in the order of billions of kilometers, which is more than sufficient.

**Practical choice.** We thus adopt an empirical stance and evaluate several random values for the frequency-generator parameters  $f_1$  and  $\gamma$ . In particular, we impose that they satisfy the metric scale conditions, i.e., the smallest and largest periods must lie in the following metric ranges:

$$\begin{aligned} P_1 &= \frac{2\pi}{f_1} \in [200\text{m}, 500\text{m}] \\ P_F &= \frac{2\pi}{f_F} \in [20\text{cm}, 2\text{m}]. \end{aligned} \quad (14)$$

Note that the 200-500 meters range specified for  $P_1$  does not relate to the scale of the full dataset, but to the scale of the point-cloud that can be observed through a single query view. In the unlikely case that this observable point-cloud is larger than 200 meters, our theoretical results above still guarantees that any encoding admits a unique pre-image in

the absence of noise. In the presence of noise, however, it could become more likely that large errors would happen during the inverse mapping.

To select the best frequency parameters, we train the network using the step II recipe on a small subset of 20K training pairs for 20 epochs and choose the parameters that yield the maximum accuracy over a small validation set of the same size. We give in Table 11 the resulting values that we have used in our experiments for different number of frequencies  $F$ .

## C.2. Inverse projection

We formulate the inverse mapping problem as a non-linear least-square projection:

$$\psi^{-1}(\mathbf{y}) = \arg \min_{t \in B} \|\psi(t) - \mathbf{y}\|^2, \quad (15)$$

where  $B$  is a search domain constructed as the union of point-clouds ranges for each input database image  $I_k$ . For instance, and without loss of generality, when considering the  $x$  channel of some 3D coordinates,  $B$  is formally defined

$$B = \bigcup_{k=1 \dots K} [\min\{x_j^k\}, \max\{x_j^k\}], \quad (16)$$

where the set  $\{x_j^k\}$  belongs to the sparse 2D-3D annotation set  $\mathcal{V}_k = \{(p_j^k, v_j^k)\}$  associated with each database image  $I_k$ . While  $B$  should ideally be the infinite real space in theory, we found important in practice to narrow down the search range for efficiency reasons as well as noise robustness.

The minimization objective of Eq. (15) can be expanded

$$\begin{aligned} S(t) &= \|\psi(t) - \mathbf{y}\|^2 \\ &= \sum_{i=1 \dots F} [(\cos(f_i t) - y_{2i-1})^2 + (\sin(f_i t) - y_{2i})^2] \\ &= \sum_{i=1 \dots F} [\cos^2(f_i t) - 2y_{2i-1} \cos(f_i t) + y_{2i-1}^2 \\ &\quad + \sin^2(f_i t) - 2y_{2i} \sin(f_i t) + y_{2i}^2] \\ &= F + \|\mathbf{y}\|^2 - 2 \sum_{i=1 \dots F} [y_{2i-1} \cos(f_i t) + y_{2i} \sin(f_i t)]. \end{aligned} \quad (17)$$

Using the trigonometric identity

$$A \cos(x) + B \sin(x) = \sqrt{A^2 + B^2} \cos\left(x - \arctan \frac{B}{A}\right) \quad (18)$$

with the convention  $\arctan(\pm\infty) = \pm\pi/2$ , we can simplify Eq. (17) into:

$$\begin{aligned} S(t) &= F + \|\mathbf{y}\|^2 - 2 \sum_{i=1 \dots F} \sqrt{y_{2i-1}^2 + y_{2i}^2} \\ &\quad \times \cos\left(f_i t - \arctan \frac{y_{2i}}{y_{2i-1}}\right). \end{aligned} \quad (19)$$

In other words, the inverse projection consists in finding the minimum of a weighted sum of cosines with different frequencies. This problem has no analytic solution in general for  $F > 2$ . It can however be approximately solved when considering inputs normalized by pairs (*i.e.*  $\mathbf{y} \in (\mathbb{S}^1)^F$ , or  $y_{2i-1}^2 + y_{2i}^2 = 1$ ).

Inspired by [47], we efficiently probe the search space at regular intervals  $t_n = \frac{n\pi}{2f_F} \in B$ , with  $n \in \mathbb{Z}$  a signed integer and where  $f_F$  is the highest frequency. A reasoning similar to the one developed in [47] suggests that at most one minima can lie in  $[t_n, t_{n+1}]$ , in which case we have  $S'(t_n) < 0$  and  $S'(t_{n+1}) > 0$ , denoting  $S'$  the derivative of  $S$ . The location of the corresponding minima can be approximated as

$$t = t_n - (t_{n+1} - t_n) \frac{S'(t_n, \mathbf{y})}{S'(t_{n+1}, \mathbf{y}) - S'(t_n, \mathbf{y})}. \quad (20)$$

Finally, we return the local minimum approximation  $t$  for which  $S(t)$  is smallest over all  $t_n$ , which in practice is a good estimate of the global minimum. Typically, for  $f_F \simeq 10m^{-1}$ , this amounts to probe every  $\simeq 10$  centimeters, which translates into 200 evaluation of  $S$  per channel and per pixel for a typical 20-meter-wide outdoor scene. Timings for this inverse mapping step are given in Section D.

## D. Detailed timings

In the main paper, we compare the localization accuracy as a function of the inference time. We measure the inference time on a A100 GPU for the model to forward one query image through the encoder, decoder and head, given a set of already-computed relevant database image representations.

In Figure 9 we plot the detailed timings for each step, this time including the post-processing steps to output the camera pose: (1) inverting the output encoding into an actual 3D dense map with the inverse mapping  $\psi^{-1}$  (see Section C); (2) computing the pose using PnP-RANSAC [63]. The forward pass in the network overall dominates the total time. This is especially true considering that the inverse mapping could be much further optimized (*e.g.* using float32 instead of float64, and computing it only for the 50% most confident pixels). Likewise, using 1,000 iterations in PnP-RANSAC instead of 10,000 would result in  $\simeq 10\times$  faster computing time for this step without noticeable loss of performance, according to preliminary experiments. Since the main focus of this paper is not speed optimization, we leave these modifications to future work.

## E. Regression head architecture

We now give more details about the regression head. In Section 4.2 of the main paper, we experiment with different

$F$	$d=6F$	$f_1$	$\gamma$	$P_1$	$P_F$
4	24	0.020772487794205544	5.7561020938998690	302m	1.59m
6	36	0.017903170262351338	3.7079736887249526	351m	50cm
8	48	0.031278470093268460	2.5735254599557535	201m	27cm

Table 11: **Frequency parameters**  $f_1$  and  $\gamma$  used in our experiments.  $P_1$  and  $P_F$  are respectively the periods of the lowest and highest frequencies.

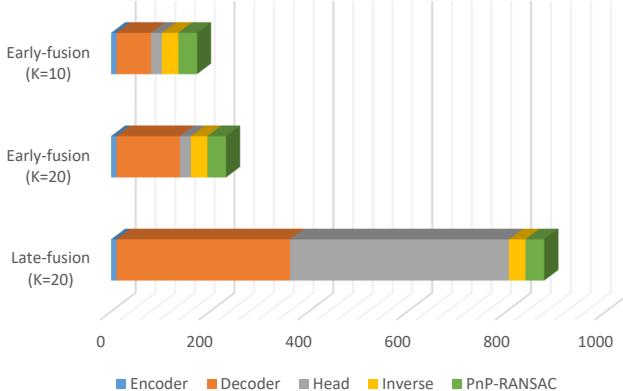


Figure 9: **Detailed timings** in milliseconds for a given image query, measured on a A100 GPU. ‘Inverse’ denotes the inverse mapping step with  $\psi^{-1}$ .

architectures and finally retain the one that regresses each of the  $4 x, y, z, \tau$  channels separately. In this case, we share the prediction head between all spatial x, y and z components as shown in Figure 10, and another head serves to predict the pixel-wise confidence.

All heads (*i.e.*, 3D regression heads and confidence head) share the same architecture, as specified in the main paper and illustrated in Figure 11. Specifically, the head consists of a sequence of 6 ConvNeXT [35] blocks interleaved with 3 PixelShuffle [54] operations to increase the resolution while simultaneously halving the channel dimension.

## F. Implementation details

We report the detailed hyperparameter settings we use for our training recipe in Table 12. Step I corresponds to the pre-training performed in CroCo [73, 74] and we refer to these papers for detailed settings.

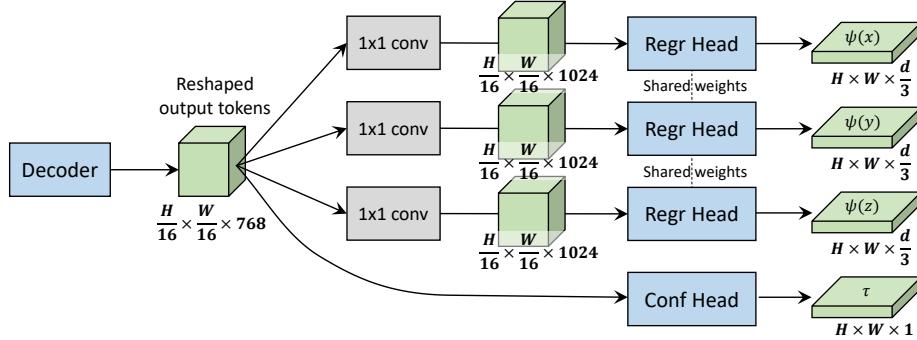


Figure 10: **Detailed pipeline of the decoupled regression head.** After reshaping the output tokens from the decoder into a dense feature map, 3 distinct  $1 \times 1$  convolutions extract channel-wise features from the same feature map. Then, a regression head with shared weight is applied to regress 3D point encodings for each channel. In parallel, another head takes care of predicting the confidence from the initial feature map.

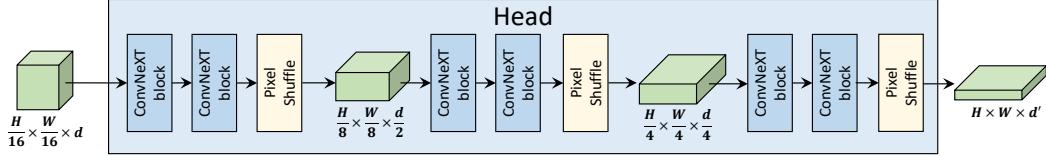


Figure 11: **Internal architecture of the regression head.** It consists of a sequence of 6 ConvNeXT [35] blocks interleaved with 3 PixelShuffle [54] operations to increase the resolution while simultaneously halving the channel dimension. In this figure, we denote the input feature dimension as  $d$  and the output dimension as  $d'$ .

Hyperparameters	Step II	Step III	Step IV
Optimizer	AdamW [36]	AdamW [36]	AdamW [36]
Base learning rate	1e-4	1e-4	1e-4
Weight decay	0.05	0.05	0.05
Adam $\beta$	(0.9, 0.95)	(0.9, 0.95)	(0.9, 0.95)
Batch size	128	128	128
Learning rate scheduler	Fixed LR	Cosine decay	Cosine decay
Training epochs	100	20	20
Warmup epochs	10	2	2
Frozen modules	Encoder	-	Encoder, 3D mixer
Number of database views	1	1	8
Input resolution	$224 \times 224$	$512 \times 384$	$512 \times 384$
Image Augmentations	Random crop, color jitter	Random crop, color jitter	Random crop, color jitter
Sampled 3D points	1024 points per DB image	1024 points per DB image	1024 points per DB image
3D Augmentations	5% triangulation noise rotations, translations in $[-1000, 1000]^3$ , rescale in $[0.5, 2]$	5% triangulation noise rotations, translations in $[-1000, 1000]^3$ , rescale in $[0.5, 2]$	5% triangulation noise rotations, translations in $[-1000, 1000]^3$ , rescale in $[0.5, 2]$
Retrieval Augmentations	Gaussian noise (std=0.05) on FiRE retrieval scores	Gaussian noise (std=0.05) on FiRE retrieval scores	Gaussian noise (std=0.05) on FiRE retrieval scores

Table 12: **Detailed hyper-parameters** for each training step.