ACE-G: Improving Generalization of Scene Coordinate Regression Through Query Pre-Training

Leonard Bruns[2,*] Axel Barroso-Laguna[1] Tommaso Cavallari[1] Áron Monszpart[4,*]
Sowmya Munukutla[1] Victor Adrian Prisacariu[1,3] Eric Brachmann[1]
[1]Niantic Spatial [2]KTH Royal Institute of Technology [3]University of Oxford [4]Third Dimension AI

https://nianticspatial.github.io/ace-g/

Abstract

Scene coordinate regression (SCR) has established itself as a promising learning-based approach to visual relocalization. After mere minutes of scene-specific training, SCR models estimate camera poses of query images with high accuracy. Still, SCR methods fall short of the generalization capabilities of more classical feature-matching approaches. When imaging conditions of query images, such as lighting or viewpoint, are too different from the training views, SCR models fail. Failing to generalize is an inherent limitation of previous SCR frameworks, since their training objective is to encode the training views in the weights of the coordinate regressor itself. The regressor essentially overfits to the training views, by design. We propose to separate the coordinate regressor and the map representation into a generic transformer and a scene-specific map code. This separation allows us to pre-train the transformer on tens of thousands of scenes. More importantly, it allows us to train the transformer to generalize from mapping images to unseen query images during pre-training. We demonstrate on multiple challenging relocalization datasets that our method, ACE-G, leads to significantly increased robustness while keeping the computational footprint attractive.

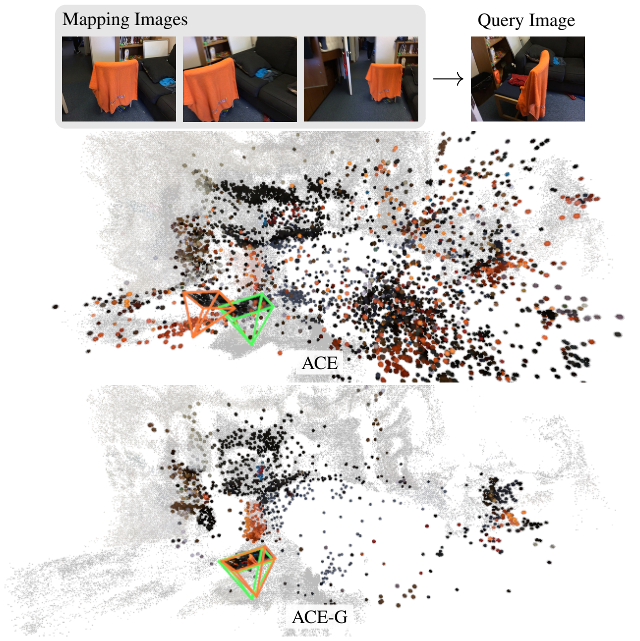


Figure 1. ACE-G is trained with separate mapping and query splits explicitly optimizing scene coordinate regression for unseen views including changing conditions. The estimated scene coordinates for the query image and the estimated and ground-truth camera poses are shown. ACE-G estimates less noisy coordinates resulting in a more accurate pose estimate compared to ACE, which degrades for larger viewpoint or scene condition changes.

1. Introduction

The limits of our training data mean the limits of our world. This statement, freely adapted from Wittgenstein, has been the guiding principle of machine learning and computer vision in recent years. Scaling architectures and training data has led to astonishing successes in language models [1], image and video synthesis [16], and more recently 3D vision [63]. Yet, there are particular tasks that are seemingly bound to small-scale learning problems. One such task is scene coordinate regression (SCR) [53].

SCR has been proposed for the task of visual relocalization. Given a set of RGB mapping images with known camera poses, one builds a visual map of an environment. After the mapping stage, the system is presented with new query images from the same environment and asked to estimate their camera poses relative to the map. SCR models solve this problem by training a scene-specific coordinate regressor on the mapping images. The model learns to as-

[*] Work done at Niantic.

sociate 2D pixels in the image with 3D coordinates in scene space. Applied to a query image, the model's prediction induces 2D-3D correspondences between image and map that yield the desired camera pose.

Leading SCR frameworks, such as DSAC* [11] or ACE [14], train scene-specific networks to encode the mapping images and to represent the map. Their capability to generalize relies on the shift-invariance of their fully-convolutional networks, and on simplistic data augmentation, such as color jitter, applied to the mapping images [11]. They excel in entirely static datasets, where there is no significant domain shift between mapping and query images [14]. However, they struggle in more realistic scenarios when revisiting the same environment later, when lighting and other factors might have changed [61]. We present a two-fold solution to the restricted generalization capabilities of previous SCR models.

Firstly, we separate the map representation from the model predicting scene coordinates. Our coordinate regressor is scene-agnostic. Its output depends on the query image, and a scene-specific map code. The map code is our scene representation, kept separate from the coordinate regressor, and trained via back-propagation through the coordinate regressor at mapping time. Keeping the coordinate regressor scene-agnostic enables pre-training it on large-scale data. This allows us to switch to more powerful transformer architectures, such as ViT [27], and to leverage more expressive features, such as DINO's [41], without the danger of overfitting to scarce scene-specific data.

Secondly, we separately pre-train the coordinate regressor with mapping images as well as with query images. When trained with mapping images, the regressor learns to produce sensible map codes that compress all scene-specific information. When training with query images, we keep the map codes fixed, requiring the transformer to bridge any gap between mapping and query views.

Our method, ACE-G (*cf*. Fig. 1), effectively produces small map codes with fast mapping times, keeping the main advantages of previous SCR methods. Extending on previous capabilities, our system learns to generalize to unseen image conditions via pre-training on 120 000 mapping-query splits. ACE-G demonstrates superior robustness on two challenging indoor datasets with long term changes between mapping and query images. We also show that the coordinate regressor generalizes well to environments outside of its training domain, making it practical and versatile.

Our contributions:

- A framework, which we call ACE-G, which combines a scene-agnostic coordinate regressor with scene-specific map codes. Map codes are few MB large, and trainable in minutes via back-propagation from posed RGB images.
- A pre-training procedure for ACE-G. We intertwine training both map codes and the transformer from mapping views, then training the transformer with separate query views while keeping the map codes fixed. This encourages the transformer to learn to generalize.
- A scalable implementation that lets us pre-train on mapping-query splits of tens of thousands of scenes.

2. Related Work

Visual Relocalization has traditionally been solved via matching of sparse features [17, 38, 44, 46–48]. Posed mapping images are triangulated using structure-from-motion software [32, 51] to yield sparse point clouds where each 3D point is associated with high dimensional descriptors. Features of the query image are matched to the point cloud resulting in 2D-3D correspondences. Finally, the query camera pose is optimized using RANSAC [30] and PnP [31].

Feature matching solutions still offer state-of-the-art accuracy and robustness when conditions between mapping and query images differ [33, 49]. The latest generation of feature matchers exhibit astonishing invariance to scale and viewpoint changes, as well as lighting changes up to day versus night [5, 24, 28, 29, 45, 54]. These recent methods are learning-based and were trained on diverse image sets.

For example, MASt3R [35, 63], a recent 3D foundation model showing state-of-the-art robustness, has been trained on a combination of multiple challenging datasets with hundreds of thousands of image pairs. While MASt3R is able to estimate metric-scale poses between pairs of images directly, its main operation mode for accurate visual relocalisation has been that of a feature matcher [35].

The main disadvantage of feature-based relocalization is the considerable mapping time needed for triangulation, and the significant memory demand to store high dimensional descriptors for the scene point cloud. While compression strategies have been proposed [17, 69], they usually come with reduced performance on challenging datasets.

Absolute pose regression has been proposed as an alternative to feature-based relocalization [34]. A neural network is trained on the mapping set to learn the association between images and camera poses [19, 52, 60]. These methods initially suffered from low accuracy [50] but recently improvements have been reported based on scene-specific data synthesis at the expense of very long mapping times [20, 39]. Relative pose regression networks learn to predict the relative pose between a query and a retrieved mapping image [2, 4, 26, 56] or a panorama image capturing the scene [68]. These networks can be scene-agnostic and pre-trained but still show comparatively low accuracy.

Scene Coordinate Regression (SCR) is related to feature-based approaches but establishes 2D-3D correspondences via dense, direct regression. Initially proposed for RGB-D inputs (using random forests [18, 53, 57]), SCR methods have later been adopted for RGB-only inputs, using neural networks [9, 11, 12, 36, 37, 40, 62]. In most works, the network is trained per scene to predict correspondences for that scene. In terms of accuracy, SCR rivals feature-matching for small to medium sized scenes [13], and some progress has been made towards larger environments [10, 61].

Initial incarnations of SCR required extensive training times on mapping data, similar or worse compared to the long mapping times of feature-based approaches. ACE [14] introduced a training procedure to reduce mapping time to mere minutes, a speed-up making SCR even suitable for iterated training in a structure-from-motion setting [15].

Relatively few works aim at scene-agnostic coordinate regression. SANet [65] and SACReg [43] have a scene-agnostic coordinate regressor that learns to interpolate 3D points of mapping images. As such, they require an external 3D reconstruction or RGB-D data as a foundation. Marepo [21] couples SCR with a scene-agnostic pose regressor. Since the scene-specific SCR component is the bottleneck in terms of generalization, Marepo fails to outperform the SCR baseline in challenging situations.

Our work shares some conceptional similarity with NeuMap [55]. Like us, they utilize a transformer coordinate regressor with scene-specific map codes. Their coordinate regressor is not fully scene-agnostic but trained per evaluation dataset and shared across those scenes. Training NeuMap further requires a full (sparse) 3D reconstruction for each evaluation scene. In contrast, our framework allows pre-training of a fully scene-agnostic coordinate regressor, and optimization of map codes from posed RGB images. Importantly, NeuMap misses any notion of query pre-training and aims primarily at scene compression at a dataset-level rather than generalization.

3. Background

Before diving into the architecture and training protocol we adopt for ACE-G, we briefly summarize the main components of the Accelerated Coordinate Encoding (ACE) SCR.

The ACE scene coordinate regression model (*cf*. Fig. 2, left) is formed by a scene-agnostic, fully-convolutional, image encoder and a scene-specific regression head (implemented as an MLP). The image encoder maps image patches to high-dimensional feature vectors, and the regression head maps these features to scene coordinates.

ACE's training protocol involves first, as a preparation step, passing all mapping images through the image encoder in order to fill a training buffer with features for a large number of randomly sampled patches together with their corresponding metadata, that is, original 2D location in the input image, camera pose, and intrinsic parameters.

Then, in each training iteration, a random batch of N features is sampled from the feature buffer and passed through the scene-specific MLP head to estimate the 3D scene points for the corresponding 2D pixels.

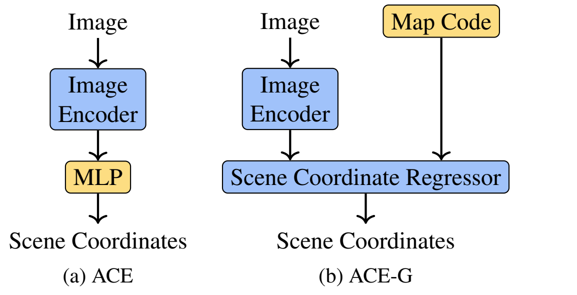


Figure 2. **High-level comparison to ACE.** The scene-specific MLP in ACE is replaced by a scene-agnostic coordinate regressor and a scene-specific map code.

The ACE training loss used to update the weights of the head is then implemented as a pixel-wise projection loss applied to the 3D scene predictions in each training iteration. For further details, please refer to the original ACE paper [14].

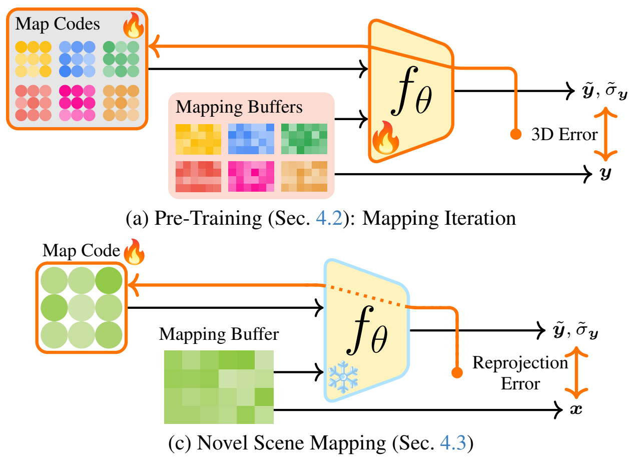
4. Method

We propose to replace the scene-specific MLP used in ACE with a scene-agnostic coordinate regressor that takes as input a latent map code and an image patch embedding and estimates the corresponding scene coordinate. See Fig. 2 for a high-level description of the architecture. A map code describes the feature $\rightarrow$ coordinate mapping for a scene. Specifically, let a map code $\mathcal{C} = \{\boldsymbol{c}_i \in \mathbb{R}^{D_\text{map}} \mid i = 1, \dots, N_\mathcal{C}\}$ denote a scene-specific set of D_map-dimensional map embeddings, $\boldsymbol{e} \in \mathbb{R}^{D_\text{feat}}$ a D_feat-dimensional patch embedding; and $\boldsymbol{y} \in \mathbb{R}^3$ the estimated scene coordinate. Our network can then be formally defined as

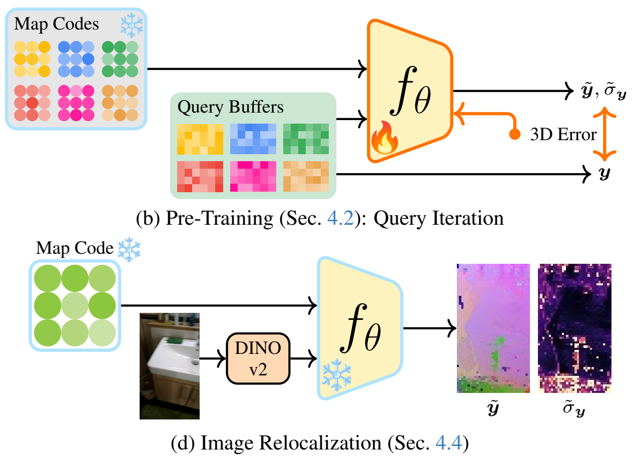
$$\begin{aligned} f_\theta : \mathbb{R}^{D_\text{feat}} \times \mathcal{P}\left(\mathbb{R}^{D_\text{map}}\right) &\rightarrow \mathbb{R}^3 \\ (\boldsymbol{e}, \mathcal{C}) &\mapsto \boldsymbol{y}, \end{aligned} \tag{1}$$

where $\mathcal{P}(\cdot)$ denotes the power set, and θ the parameters.

Figure 3 provides an overview of the different stages of optimization and inference. During pre-training of the scene-agnostic regressor (Sec. 4.2), the network is simultaneously optimized across multiple scenes using disjoint sets of mapping and query images. The mapping images are used to infer scene-specific map codes, while the query images are used to optimize the scene-agnostic network aiming to improve generalization for novel views, given the previously optimized map codes. Once the scene-agnostic coordinate regressor has been pre-trained, the latent map code for a new scene can be optimized within minutes (Sec. 4.3), akin to the per-scene optimization in ACE. To relocalize a query image, the scene-specific map code is used to condition the prediction of the 3D coordinates for all image patches, and PnP with RANSAC can then be used to estimate the camera pose (Sec. 4.4).



Map Codes
Mapping Buffers
f_θ
$\tilde{y}, \tilde{\sigma}_y$
3D Error
y
(a) Pre-Training (Sec. 4.2): Mapping Iteration
Map Code
Mapping Buffer
Reprojection Error
x
(c) Novel Scene Mapping (Sec. 4.3)



Map Codes
Query Buffers
f_θ
$\tilde{y}, \tilde{\sigma}_y$
3D Error
y
(b) Pre-Training (Sec. 4.2): Query Iteration
Map Code
DINO v2
$\tilde{y}$
$\tilde{\sigma}_y$
(d) Image Relocalization (Sec. 4.4)

Figure 3. **Overview of ACE-G**. The scene-agnostic coordinate regressor f_θ is pre-trained by alternating between mapping iterations (a) and query iterations (b). (a) During pre-training mapping iterations, both map codes and network weights are optimized 🔥 using precomputed buffers ▦ storing shuffled DINOv2 features and meta data necessary for supervision (*cf*. [14]). For each mapping buffer a corresponding map code ● is optimized. (b) During pre-training query iterations, the map codes are frozen ❄ and the network is trained to estimate scene coordinates for query buffers made up of viewpoints or scene conditions different from the mapping buffers'. (c) Once the regressor has been pre-trained, a novel scene can be encoded in a new map code by minimizing the reprojection error. (d) Given such an optimized map code and a new query image, scene coordinates and uncertainty can be estimated via a forward pass. The resulting 2D-3D correspondences can then be used to estimate the camera pose.

4.1. Architecture

At a high-level, the architecture of ACE-G consists of a pre-trained, generic, image encoder and a scene-agnostic coordinate regressor (Fig. 2).

Image Encoder As the image encoder we use DINOv2 [23, 41] hypothesizing that higher-level image understanding embedded in the resulting features might be beneficial for more difficult query images (*cf*. *e.g.*, [5]). Given an input image $\mathbf{I}$, the encoder f_{enc} predicts a set of patch embeddings $\mathcal{E} = \{\boldsymbol{e}_i \in \mathbb{R}^{D_{\text{feat}}} \mid i = 1, \ldots, N_{\mathcal{E}}\}$.

Scene-Agnostic Coordinate Regressor Our scene-agnostic coordinate regressor is shown in Fig. 4. It consists of N cross-attention-only vision transformer blocks [27]. Patch embeddings $\boldsymbol{e}$ are used as query tokens and the individual map embeddings in the map code $\mathcal{C}$ are used as key and value tokens. The map embeddings use no additional positional encodings and are therefore seen as permutation-invariant by the transformer. Following the transformer, a 2-layer MLP without layer normalization [3] is used to regress the scene coordinate $\boldsymbol{y}$ and standard deviation $\sigma_{\boldsymbol{y}}$ of a Laplace distribution (see loss below).

4.2. Mapping/Query Pre-Training

Our goal is to train a scene-agnostic coordinate regressor which, after finding the scene-specific map code, generalizes further than simply optimizing an MLP for the same training data. To achieve this, we train the scene-agnostic coordinate regressor with query images separate from the mapping images that were used to find the map codes. This

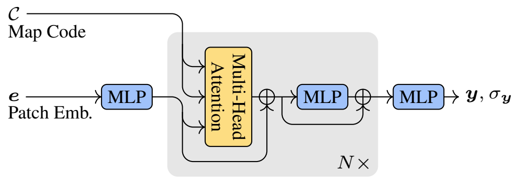


Figure 4. **Network architecture.** The scene-agnostic coordinate regressor consists of N cross-attention-only blocks. Given a patch embedding $\boldsymbol{e}$ and a map code $\mathcal{C}$ it estimates the 3D scene coordinate $\boldsymbol{y}$ and uncertainty $\sigma_{\boldsymbol{y}}$.

is akin to how the network is later used to map novel scenes and relocalize query images: the map code is inferred from one sequence of images (*i.e*., the mapping images), and a query image taken potentially during a different day or from a different viewing angle is to be localized relative to the mapping images given only the map code.

For pre-training we consider datasets for which ground-truth scene coordinates are available. Either from an RGB-D sensor, synthetic data generation, or generated from RGB through multi-view stereo (*cf*. Tab. 2). Note that our architecture does not require ground-truth scene coordinates when optimizing map codes on novel scenes (see Sec. 4.3).

Multi-Buffer Dataset Previously, it has been shown that precomputing image features and storing them as a shuffled buffer can significantly speed up the optimization of scene coordinate regressors as repeated feature extraction is avoided and patches from all views are shuffled, decorrelating the gradients within each batch [14]. Following this approach, we precompute and store the buffers for all mapping and query training sequences.

Let $\mathcal{B} = \{(\mathcal{M}_i, \mathcal{Q}_i) \mid i = 1, \ldots, N_{\mathcal{B}})\}$ denote the dataset of $N_{\mathcal{B}}$ mapping-query tuples. Each tuple i consists of a mapping buffer $\mathcal{M}_i = \{(\boldsymbol{e}_j, \boldsymbol{y}_j) \mid j = 1, \ldots, M_i\}$ and query buffer $\mathcal{Q}_i = \{(\boldsymbol{e}_j, \boldsymbol{y}_j) \mid j = 1, \ldots, Q_i\}$ made up of patch embeddings $\boldsymbol{e}_j$ and corresponding ground-truth scene coordinates $\boldsymbol{y}_j$. During the optimization, each mapping-query tuple i is associated with a map code $\mathcal{C}_i \in \mathcal{G}$[1] that will be optimized on the corresponding mapping buffer $\mathcal{M}_i$.

Mapping and Query Optimization Our pre-training alternates between "mapping" and "query" optimization iterations. During mapping iterations, the scene-agnostic network parameters and map codes are optimized jointly:

$$\theta^*, \mathcal{G}^* = \arg\min_{\theta, \mathcal{G}} \mathbb{E}_{\mathcal{M}} \left[\log \tilde{\sigma}_{\boldsymbol{y}} + \sqrt{2} \frac{\|\tilde{\boldsymbol{y}} - \boldsymbol{y}\|}{\tilde{\sigma}_{\boldsymbol{y}}} \right]. \quad (2)$$

Here, $(\tilde{y}, \tilde{\sigma}_{\boldsymbol{y}}) = f_\theta(\boldsymbol{e}, \mathcal{C}_i)$ denote the estimates for a patch embedding $\boldsymbol{e}$ with ground-truth coordinate $\boldsymbol{y}$ sampled from a mapping buffer $\mathcal{M}_i \sim \mathcal{M}$. Intuitively, this stage ensures that the network has the capacity to fit various types of scenes and move information from the buffers into the map codes.

Conversely, during query iterations, only the scene-agnostic network parameters are updated, keeping the map codes $\mathcal{C}^*$ fixed:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{\mathcal{Q}} \left[\log \tilde{\sigma}_{\boldsymbol{y}} + \sqrt{2} \frac{\|\tilde{\boldsymbol{y}} - \boldsymbol{y}\|}{\tilde{\sigma}_{\boldsymbol{y}}} \right], \quad (3)$$

where $(\tilde{y}, \tilde{\sigma}_{\boldsymbol{y}}) = f_\theta(\boldsymbol{e}, \mathcal{C}_i^*)$. This stage aims to improve generalization beyond previous scene coordinate regressors that are optimized just on mapping images.

For both mapping and query iterations, our optimization objective is the negative log-likelihood of the ground-truth scene coordinates under an estimated Laplace distribution. That is, after pre-training, $\tilde{\sigma}_{\boldsymbol{y}}$ can be interpreted as the estimated standard deviation of the scene coordinate in 3D.

Implementation Instead of optimizing the network and map codes using all scenes simultaneously until convergence, similar to an autodecoder [8, 42, 55], we opt to repeatedly optimize scenes from scratch throughout the training process. Specifically, we only maintain a subset of N_{active} scenes $\mathcal{B}_{\text{active}} \subset \mathcal{B}$ and a map code for each. Each of the active map codes is updated until a randomized number of iterations is reached, at which point it is reset (we initialize map codes by sampling from a Gaussian distribution with $\sigma = 0.01$) and a new scene configuration is sampled from $\mathcal{B}$ and added to the pool, replacing the previous one.

During pre-training, we iteratively perform mapping iterations followed by query iterations. Mapping iterations optimize both the network parameters and the map codes, using Eq. (2) and batches sampled from the $\mathcal{M}$ buffers. Query iterations then use Eq. (3), keeping the map codes frozen and sampling batches from the $\mathcal{Q}$ buffers. To reduce the chance of overfitting the network to the currently active scenes, we further only perform network updates in every 10th iteration, that is, only the map codes are updated in 9 out of 10 optimization iterations. Further, during query optimization we skip scenes when their corresponding map code has undergone less than N_{qstandby} mapping iterations. Intuitively, we want the scene-agnostic network to learn generalizing from map codes that are well initialized, instead of spending capacity on estimating the uncertainty for insufficiently optimized map codes. Each iteration is based on randomly sampled batches of the active scenes' buffers composed of N_{spb} scenes per batch and N_{pps} patches per scene.

[1]Here, $\mathcal{G} = \{\mathcal{C}_i \mid i = 1, \ldots, N_{\mathcal{B}}\}$ denotes a set of map codes $\mathcal{C}_i$.

4.3. Novel Scene Mapping

To find the map code for a new scene we mainly follow the approach used by ACE [14]: given a new set of posed images $\{\mathbf{I}_i, {}^{\text{w}}\mathbf{T}_{\text{c}i} \mid i = 1, \ldots, N\}$, the mapping buffer $\mathcal{M}$ is prepared ahead of training. However, instead of the ground-truth scene coordinate $\boldsymbol{y}$, the image coordinate $\boldsymbol{x}$, camera pose ${}^{\text{w}}\mathbf{T}_{\text{c}}$, and intrinsic matrix $\mathbf{K}$ are stored for each patch embedding $\boldsymbol{e}$. The map code for a novel scene is then optimized by minimizing the negative log-likelihood in 2D:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \mathbb{E}_{\mathcal{M}} \left[\log \tilde{\sigma}_{\boldsymbol{x}} + \sqrt{2} \frac{\|\tilde{\boldsymbol{x}} - \boldsymbol{x}\|}{\tilde{\sigma}_{\boldsymbol{x}}} \right], \quad (4)$$

where $\tilde{\boldsymbol{x}}$ and $\tilde{\sigma}_{\boldsymbol{x}}$ are derived from $(\tilde{\boldsymbol{y}}, \tilde{\sigma}_{\boldsymbol{y}}) = f_\theta(\boldsymbol{e}, \mathcal{C})$ via pinhole projection. For invalid estimates (behind the camera or with large reprojection errors), the constant depth prior of ACE is used but modified to take uncertainty into account, similarly to Eq. (2).

4.4. Image Relocalization

Given a new image $\mathbf{I}$ for a previously mapped scene with map code $\mathcal{C}^*$, the scene coordinates are estimated using a simple forward pass through the image encoder and the scene-agnostic coordinate regressor. Specifically, each image patch (with its known 2D position $\boldsymbol{x}_i$) is first mapped to a patch embedding $\boldsymbol{e}_i \in \mathcal{E} = f_{\text{enc}}(\mathbf{I})$ via the image encoder and passed to the pre-trained scene-agnostic regressor to estimate the corresponding 3D coordinate and uncertainty $\tilde{\boldsymbol{y}}_i, \tilde{\sigma}_{\boldsymbol{y},i} = f_\theta(\boldsymbol{e}_i, \mathcal{C}^*)$. The resulting set of 2D-3D correspondences $\{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i) \mid i = 1, \ldots, N_{\text{patches}}\}$ can directly be used with PnP and RANSAC to estimate the camera pose ${}^{\text{w}}\widetilde{\mathbf{T}}_{\text{c}}$ (see [11] for further details on camera pose estimation from the set of 2D-3D correspondences).

We find that the estimated uncertainty $\tilde{\sigma}_{\boldsymbol{y}}$ can be used to prefilter the correspondences fed to the RANSAC algorithm. Specifically, we use an adaptive uncertainty threshold based on the lowest p-quantile of uncertainties, that is,

Table 1. **Results on Indoor-6.** Each cell contains in order: percent of correctly localized frames under $(5^\circ, 25\,\mathrm{cm})$, median translation error in cm, and median rotation error in degrees. Best in each SCR group **highlighted**.

	(% / cm / °)	Map. Time	Map Size	Indoor-6						
				Scene 1	Scene 2a	Scene 3	Scene 4a	Scene 5	Scene 6	Avg.
Non SCR	Reloc3r [26]	< 5 min.	∼500 MB	76 / 8 / 0.6	76 / 10 / 0.9	87 / 4 / 0.5	82 / 9 / 0.8	50 / 22 / 1.4	86 / 3 / 0.6	76 / 9 / 0.8
Non SCR	MASt3R+Kapture [35]	5-10 h	∼5 GB	92 / 2.4 / 0.4	97 / 2.6 / 0.3	96 / 1.8 / 0.4	96 / 2.6 / 0.5	85 / 4.3 / 0.7	95 / 1.3 / 0.3	94 / 2.5 / 0.5
SCR 25 min.	GLACE [61]	25 min.	6 MB	90 / 3.8 / 0.7	**100 / 3.9 / 0.4**	93 / **2.9 / 0.6**	**99 / 2.3 / 0.5**	79 / 6.1 / 0.9	**97 / 2.1 / 0.4**	93 / **3.5 / 0.6**
SCR 25 min.	Ours (25 min.)	25 min.	12 MB	**96 / 3.5 / 0.6**	98 / **3.9 / 0.4**	**98** / 3.8 / 0.8	**99** / 3.2 / 0.8	**97 / 5.1 / 0.8**	91 / 2.7 / 0.6	**96** / 3.7 / 0.7
SCR 5 min.	ACE [14]	5 min.	4 MB	52 / 17.7 / 2.4	86 / 8.0 / 0.8	66 / 9.0 / 1.6	94 / 4.6 / 0.9	51 / 21.6 / 4.0	70 / 5.2 / 0.9	70 / 11.0 / 1.8
SCR 5 min.	DINO-ACE	5 min.	4 MB	90 / 5.4 / 0.9	97 / 5.2 / 0.6	91 / 6.0 / 1.2	96 / 4.7 / 1.2	89 / 7.5 / 1.2	85 / 4.8 / 1.0	91 / 5.6 / 1.0
SCR 5 min.	Ours (5 min.)	5 min.	12 MB	**94 / 4.3 / 0.7**	**98 / 4.8 / 0.4**	**96 / 4.7 / 0.9**	**98 / 3.7 / 0.9**	**93 / 5.9 / 1.0**	**92 / 3.9 / 0.8**	**95 / 4.5 / 0.8**

Table 2. **Training datasets.** (# scenes: number of scans included in the training; Cov.: uses image pairs by [63]; Sep. Q.: separate query sequence; 3D: source of 3D information)

	# Scenes	Cov.	Sep. Q.	Metric	3D
ScanNet [22]	1201	✗	✗	✓	RGB-D
ScanNet++ v1 [67]	199	✓	✗	✓	RGB-D
ARKitScenes [6]	4520	✓	✗	✓	RGB-D
MapFree [2]	2×400	✗	✓	✓	MVS [7]
BlendedMVS [66]	462	✗	✗	✗	Synth.
WildRGBD [64]	22359	✗	✗	✓	RGB-D

$\sigma_{\text{thresh}} = f \cdot Q_p(\Sigma)$, where Q_p denotes the p-quantile and Σ is the set of estimated uncertainties. A reduced set of correspondences $\{(\boldsymbol{x}_i, \tilde{\boldsymbol{y}}_i) \mid i = 1, \dots, N_{\text{patches}} \wedge \tilde{\sigma}_{\boldsymbol{y},i} < \sigma_{\text{thresh}}\}$ is then used to estimate the camera pose. In all experiments we use $p = 0.1$ and $f = 2.0$.

5. Experiments

Pre-Training Datasets Table 2 provides an overview of the datasets used to pre-train the scene-agnostic coordinate regressor. For all datasets, for each scene, we select "mapping" and "query" chunks by splitting the sequences into disjoint – likely solvable – mapping and query sections. The distributions are adjusted per dataset to account for differences in camera trajectories and scene content. For some datasets, the information about co-visible training image pairs published by [63] is used to inform the sampling procedure. For each dataset we sample 20 000 map/query configurations, then pre-compute a resulting total of 240 000 $\mathcal{M}$ and $\mathcal{Q}$ buffers. See supplementary material for further information and qualitative training data samples.

Evaluation Datasets We focus our evaluation on challenging datasets with varying lighting conditions, strong view point differences, and long-term changes. We report the performance of ACE-G on the Indoor-6 [25], RIO10 [59], and Cambridge Landmarks [34] datasets. Indoor-6 depicts large multi-room indoor environments, including significant lighting variations between mapping and query scans. RIO10 challenges the algorithms with scans captured over the course of several weeks and months, showing significant day-to-day variations in the positions of objects between scans, in addition to variable lighting conditions. For this dataset we train the map codes on the "mapping" scans and compute the error metrics on the "validation" scans. Cambridge Landmarks is used to assess the performance in larger outdoor environments, and puts to the test the capabilities of ACE-G as none of the training datasets contain similarly large, outdoor environments.

Baselines We compare ACE-G with five primary baselines: ACE [14] in its default setting, using its fully-convolutional encoder; a modified version of ACE, "DINO-ACE", where we replaced the encoder with DINOv2 [41], *i.e.*, the same encoder that we are using; GLACE [61], which improves over ACE by adding a noise-augmented global encoding among other modifications; Reloc3r [26], and MASt3R [35] which both rely on image retrieval and – just for the latter – SfM models generated via Kapture [32] for map-relative localization [63]. Reliance on SfM reconstructions leads to significant mapping times and memory demands for MASt3R. Reloc3r maps quickly by building a retrieval index, but needs to store the mapping images. We present results for two variants of ACE-G: the default one that can perform mapping of novel scenes in ≈5 min on a single GPU, similarly to the default configuration of ACE; and one configuration tuned to optimize map codes in ≈25 min, to be fairly comparable with GLACE. ACE-G maps consume 12 MB of memory stored using full precision ($N_C = 4096, D_{\text{map}} = 768$).

Metrics We report the median translation and rotation error of the estimated poses, as well as the percentage of correctly localized frames under different translation and rotation error thresholds.

Hyperparameters We pre-train for 4.4M iterations on 8 A100 GPUs with a scene batch size $N_{\text{spb}} = 200$ and patch batch size $N_{\text{pps}} = 512$. Each map code is optimized between 6 000 and 10 000 iterations with $N_{\text{qstandby}} = 5000$. See supplementary material for further details.

Table 3. **Results on RIO10.** Each cell contains: median translation error and median rotation error. Best in each SCR group **highlighted**.

$(\mathrm{cm} / {}^\circ)$	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5	Scene 6	Scene 7	Scene 8	Scene 9	Scene 10	Avg.
Reloc3r [26]	29 / 3.8	23 / 3.2	45 / 4.1	86 / 53.4	35 / 3.8	10 / 2.1	20 / 2.5	95 / 7.7	76 / 7.4	51 / 6.3	47 / 9.4
MASt3R+Kapture [35]	69 / 26.4	44 / 14.9	61 / 22.2	N/A	68 / 22.9	7.0 / 2.7	8.9 / 2.9	N/A	N/A	106 / 46.9	N/A
GLACE [61]	22.1 / 7.8	**21 / 6.1**	**34 / 10.5**	266.2 / 77.9	127.8 / 45.6	18.8 / 6.9	**19.9 / 5.2**	221.1 / 53.4	323.3 / 89.8	85.5 / 30.9	114 / 33.4
Ours (25 min.)	**20.9 / 6.8**	25.8 / 7.6	40.2 / 10.9	**88.2 / 32.9**	**17.6 / 6.4**	**14.4 / 4.7**	26.5 / 8.6	**68.3 / 16.7**	**29.2 / 9.6**	**39.1 / 17.3**	**37.0 / 12.1**
ACE [14]	97.1 / 25.8	95.1 / 22.6	159.8 / 35.1	390.3 / 91.2	915.9 / 95.0	170.5 / 57.1	57.7 / 13.5	481.8 / 57.1	361.6 / 100.3	854.8 / 89.2	358.4 / 58.7
DINO-ACE	48.3 / 13.0	45.1 / 11.6	63.5 / 13.2	108.9 / **32.7**	118.2 / 10.6	21.6 / 7.0	53.0 / 12.8	233.9 / 21.2	53.6 / 16.9	92.1 / 20.2	83.8 / 15.9
Ours (5 min.)	**25.2 / 8.3**	**29.7 / 8.5**	**45.6 / 12.4**	**89.3** / 37.8	**20.3 / 7.4**	**17.8 / 5.6**	**30 / 9.7**	**75.3 / 18.7**	**32.9 / 11.0**	**44.5 / 18.0**	**41.1 / 13.8**

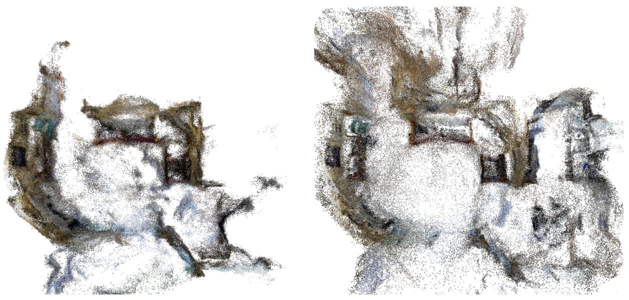


Figure 5. **Scene reconstruction.** Comparing the estimated scene coordinates of ACE (left) and ACE-G (right) on Scene 1 of Indoor-6 shows that ACE fails to reconstruct some parts of the apartment.

5.1. Evaluation Results

Indoor-6 Table 1 shows results on Indoor-6. Our method performs on par with GLACE, and outperforms the baselines in the 5 minute SCR group. Comparing ACE, DINO-ACE, and ACE-G it can be seen that the DINO features improve the performance significantly over the ACE features, and a further improvement across metrics is achieved thanks to ACE-G's query-optimized, scene-agnostic coordinate regressor. Comparing the scene coordinates predicted by ACE and ACE-G in Fig. 5 indicates that ACE fails to cover some parts of the scene, suggesting that some features become too ambiguous to handle the multi-room scenes in Indoor-6. GLACE, which addresses this issue through an added global encoding, performs similarly to our method.

RIO10 Table 3 shows a significant improvement in terms of robustness to changing object arrangements, compared to other SCR methods (*e.g.*, note that for all scenes the median error is consistently lower than $< 1\,\mathrm{m}$). MASt3R fails to compute a pose for more than 50% of images for some of the scenes (marked as N/A) highlighting the challenging nature of this dataset. Reloc3r, the only method achieving results similar to ours, requires access to the mapping images to perform relocalization. Figure 6 further highlights ACE-G's improved generalization ability.

Cambridge Landmarks Table 4 reports median errors for Cambridge Landmarks. The results show that among the SCR methods, GLACE, which aims for scalability to larger scenes performs best. Our method performs slightly worse than ACE on average. One reason for the different trend of results compared to Indoor-6 and RIO10 is the relatively small difference between mapping and query present in this dataset. In addition, note that our pre-training did not include scenes comparable in size and content to the ones present in this dataset. Interestingly, our method still outperforms DINO-ACE overall, that is, our pre-training still improved performance.

Table 4. **Results on Cambridge Landmarks.** Each cell contains in order: median translation error in cm and median rotation error in degrees. Best in each SCR group **highlighted**. Despite this dataset being out-of-distribution with respect to our training datasets, ACE-G achieves competitive results. Our architecture still improves notably over the DINO-ACE baseline.

$(\mathrm{cm} / {}^\circ)$	GC	KC	OH	SF	SMC	Avg.
Reloc3R [26]	122 / 0.7	42 / 0.4	62 / 0.6	13 / 0.6	34 / 0.6	55 / 0.6
MASt3R+K. [35]	12.6 / 0.6	7.3 / 0.1	15.2 / 0.3 /	3.8 / 0.2	4.1 / 0.1	9.1 / 0.2
GLACE	**18.0 / 0.1**	**18.0 / 0.3**	**17.9 / 0.4**	**4.5 / 0.2**	**8.4 / 0.3**	**13.5 / 0.3**
Ours (25 min.)	50.6 / 0.3	21.1 / 0.3	22.5 / 0.5	7.4 / 0.3	24.9 / 0.8	25.3 / 0.4
ACE	**42.2 / 0.2**	26.7 / **0.4**	30.7 / 0.6	**5.3** / 0.3	**20.6 / 0.6**	**25.1 / 0.4**
DINO-ACE	61.4 / 0.3	38.9 / 0.4	39.8 / 0.6	5.3 / **0.3**	74.0 / 1.8	43.9 / 0.7
Ours (5 min.)	68.2 / 0.4	**26.3** / 0.4	**28.7 / 0.5**	7.8 / 0.3	44.2 / 1.4	35.1 / 0.6

Table 5. **Accuracy under fine thresholds.** Proportion of frames correctly localized under ($5°$, $5\,\mathrm{cm}$) error threshold for Indoor-6 and RIO10, and ($10°$, $10\,\mathrm{cm}$) error for Cambridge. Best in each SCR group **highlighted**.

(%)	Indoor-6	RIO10	Cambridge
Reloc3R [26]	37.8	9.7	11.8
MASt3R+Kapture [35]	77.0	13.0	65.0
GLACE	**69.6**	**9.8**	**42.5**
Ours (25 min.)	64.7	7.4	24.6
ACE	33.5	4.2	**26.3**
DINO-ACE	44.4	2.9	17.3
Ours (5 min.)	**54.7**	**5.7**	17.9

Fine Accuracy Table 5 reports results for finer accuracy thresholds across datasets. Compared to the previously reported, more lenient thresholds and mean median metrics, ACE-G performs slightly worse than GLACE (compare relative performance to GLACE in Tab. 1 and Tab. 3). We hypothesize that the slightly worse performance for fine accuracies can be attributed to the lower resolution of DINO features – patch size of 14 pixels across vs. 8 pixels for ACE and GLACE's convolutional image encoder – resulting in fewer estimated correspondences per image and pos-

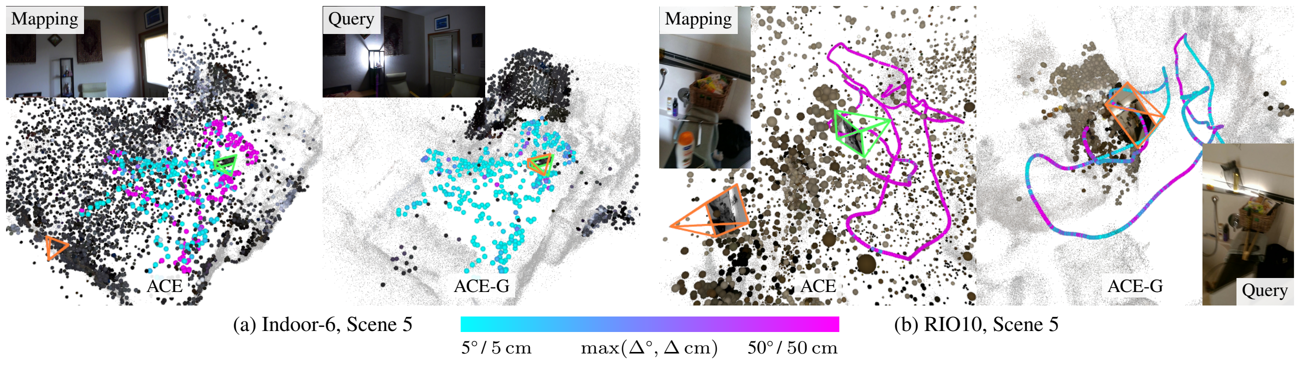


Figure 6. **Qualitative comparison for ACE and ACE-G.** The full ground-truth trajectory is shown as dots with color indicating the error. The estimated scene coordinates for one query image are shown along with the estimated and ground-truth camera pose (mapping image shows a close mapping image chosen based on manual inspection). ACE fails to generalize to changing lighting conditions (a) and changing objects (b), while ACE-G's predictions remain sensible resulting in a correct pose estimate (best viewed digitally).

(%)	SN	SN++	ARK	MF	BMVG	WR
Mapping-only	48.2	39.8	39.4	38.4	28.9	26.3
Mapping+query	56.2	49.2	49.3	42.3	32.7	32.7
Change	+8.0	+9.4	+9.9	+3.9	+3.8	+6.4

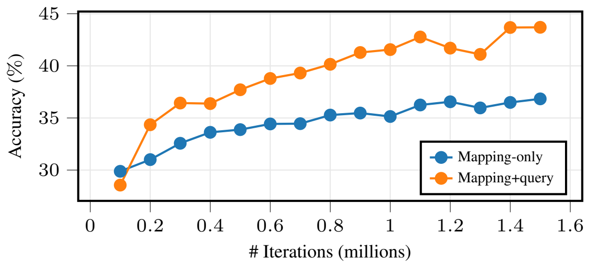


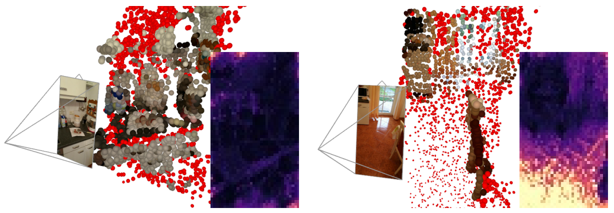
Figure 7. **Query training.** Top: mean accuracy at the $(20°, 20\,\mathrm{cm})$ error threshold after 1.5M iterations, and relative change attributable to query training. Bottom: mean accuracy on the validation splits as training progresses.

sibly less accurate geometry estimation. Nevertheless, for indoor datasets ACE-G remains better than ACE and DINO-ACE. Notably, on RIO10, DINO-ACE performs worse than ACE, but ACE-G performs better than both, highlighting the advantage of the pre-trained coordinate regressor.

5.2. Analysis

Query Training In Fig. 7 we show the performance on the validation sets with and without query iterations during the pre-training of the scene-agnostic coordinate regressor. The "mapping+query" model alternates between mapping and query optimization iterations, whereas the "mapping" model only performs mapping iterations (*cf*. Sec. 4.2). By including the query training optimization steps we can see a clear improvement across dataset validation performance, increasing the robustness and generalization of ACE-G.

Uncertainty-Based Prefiltering As described in Sec. 4.4, we use the uncertainty predicted by the model to prefilter the scene coordinates used for registration. Figure 8 shows the filtered and raw coordinates for two examples. The adaptive threshold adjusts the number of estimated correspondences used in PnP based on the image difficulty. For a query image with small mapping-query gap and large visual overlap, most estimates will have similar uncertainty and will pass the quantile-based filter. For a query image with larger mapping-query gap the uncertainty varies depending on image content and previously seen mapping data and the filter will remove the high uncertainty estimates.



(%)	SN	SN++	ARK	MF	BMVG	WR
No prefiltering	73.2	65.6	63.4	57.3	56.6	42.8
With prefiltering	77.0	70.3	65.3	63.5	57.2	51.8
Change	+3.8	+4.7	+1.9	+6.2	+0.6	+9.0

Figure 8. **Uncertainty prefiltering.** Quantile-based filtering removes points (highlighted) based on the difficulty of the query image, improving results.

6. Conclusion

We presented ACE-G, a transformer-based approach to scene coordinate regression explicitly optimized to generalize to novel views and changing conditions. Built on top of DINO features, our approach demonstrates improved invariance to changing lighting conditions and robustness to environment changes. It outperforms state-of-the-art scene coordinate regression methods on difficult datasets and achieves competitive performance on large-scale datasets without major mapping-query differences. Overall, ACE-G is a strong alternative to traditional SCRs that are optimized per-scene, and an important conceptual step towards enabling further learning-based approaches.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] Eduardo Arnold, Jamie Wynn, Sara Vicente, Guillermo Garcia-Hernando, Aron Monszpart, Victor Prisacariu, Daniyar Turmukhambetov, and Eric Brachmann. Map-free visual relocalization: Metric pose relative to a single image. In *ECCV*, pages 690–708, 2022. 2, 6

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

[4] V. Balntas, S. Li, and V. Prisacariu. RelocNet: Continuous metric learning relocalisation using neural nets. In *ECCV*, 2018. 2

[5] Axel Barroso-Laguna, Sowmya Munukutla, Victor Adrian Prisacariu, and Eric Brachmann. Matching 2D images in 3D: Metric relative pose from metric correspondences. In *CVPR*, 2024. 2, 4

[6] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Yuri Feigin, Peter Fu, Thomas Gebauer, Daniel Kurz, Tal Dimry, Brandon Joffe, Arik Schwartz, and Elad Shulman. ARKitScenes: A diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data. In *NeurIPS*, 2021. 6, 2

[7] Michael Bleyer, Christoph Rhemann, and Carsten Rother. PatchMatch stereo-stereo matching with slanted support windows. In *BMVC*, 2011. 6

[8] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In *ICML*, pages 600–609. PMLR, 2018. 5

[9] Eric Brachmann and Carsten Rother. Learning less is more - 6D camera localization via 3D surface regression. In *CVPR*, 2018. 3

[10] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *ICCV*, 2019. 3

[11] Eric Brachmann and Carsten Rother. Visual camera re-localization from RGB and RGB-D images using DSAC. *TPAMI*, 44(9):5847–5865, 2021. 2, 3, 5

[12] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-differentiable RANSAC for camera localization. In *CVPR*, 2017. 3

[13] Eric Brachmann, Martin Humenberger, Carsten Rother, and Torsten Sattler. On the limits of pseudo ground truth in visual camera re-localisation. In *ICCV*, 2021. 3

[14] Eric Brachmann, Tommaso Cavallari, and Victor Adrian Prisacariu. Accelerated coordinate encoding: Learning to relocalize in minutes using RGB and poses. In *CVPR*, 2023. 2, 3, 4, 5, 6, 7

[15] Eric Brachmann, Jamie Wynn, Shuai Chen, Tommaso Cavallari, Áron Monszpart, Daniyar Turmukhambetov, and Victor Adrian Prisacariu. Scene coordinate reconstruction: Posing of image collections via incremental learning of a relocalizer. In *ECCV*, 2024. 3

[16] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024. 1

[17] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid scene compression for visual localization. In *CVPR*, 2019. 2

[18] Tommaso Cavallari, Stuart Golodetz, Nicholas A Lord, Julien Valentin, Luigi Di Stefano, and Philip HS Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *CVPR*, 2017. 2

[19] Shuai Chen, Zirui Wang, and Victor Prisacariu. Direct-PoseNet: Absolute pose regression with photometric consistency. In *3DV*, 2021. 2

[20] Shuai Chen, Xinghui Li, Zirui Wang, and Victor Prisacariu. DFNet: Enhance absolute pose regression with direct feature matching. In *ECCV*, 2022. 2

[21] Shuai Chen, Tommaso Cavallari, Victor Adrian Prisacariu, and Eric Brachmann. Map-relative pose regression for visual re-localization. In *CVPR*, 2024. 3

[22] 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. 6, 2

[23] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024. 4, 1

[24] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 2

[25] Tien Do, Ondrej Miksik, Joseph DeGol, Hyun Soo Park, and Sudipta N. Sinha. Learning to detect scene landmarks for camera localization. In *CVPR*, 2022. 6

[26] Siyan Dong, Shuzhe Wang, Shaohui Liu, Lulu Cai, Qingnan Fan, Juho Kannala, and Yanchao Yang. Reloc3r: Large-scale training of relative camera pose regression for generalizable, fast, and accurate visual localization. In *CVPR*, 2025. 2, 6, 7

[27] 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. 2, 4

[28] Johan Edstedt, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. DeDoDe: Detect, don't describe – describe, don't detect for local feature matching. In *3DV*, 2024. 2

[29] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Revisiting robust losses for dense feature matching. In *CVPR*, 2024. 2

[30] 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. 2

[31] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the

perspective-three-point problem. *TPAMI*, 25(8):930–943, 2003. 2

[32] 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. 2, 6

[33] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *IJCV*, 129(2):517–547, 2021. 2

[34] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *ICCV*, 2015. 2, 6

[35] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3D with MASt3R. In *ECCV*, pages 71–91, 2024. 2, 6, 7

[36] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *CVPR*, 2020. 3

[37] Ting-Ru Liu, Hsuan-Kung Yang, Jou-Min Liu, Chun-Wei Huang, Tsung-Chih Chiang, Quan Kong, Norimasa Kobori, and Chun-Yi Lee. Reprojection errors as prompts for efficient scene coordinate regression. In *ECCV*, 2024. 3

[38] Simon Lynen, Bernhard Zeisl, Dror Aiger, Michael Bosse, Joel Hesch, Marc Pollefeys, Roland Siegwart, and Torsten Sattler. Large-scale, real-time visual-inertial localization revisited. *IJRR*, 39(9), 2019. 2

[39] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. LENS: Localization enhanced by NeRF synthesis. In *CoRL*, 2021. 2

[40] Son Tung Nguyen, Alejandro Fontan, Michael Milford, and Tobias Fischer. FocusTune: Tuning visual localization through focus-guided sampling. In *WACV*, pages 3606–3615, 2024. 3

[41] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. 2, 4, 6, 1

[42] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 5

[43] Jerome Revaud, Yohann Cabon, Romain Brégier, JongMin Lee, and Philippe Weinzaepfel. SACReg: Scene-agnostic coordinate regression for visual localization. In *CVPRW*, 2024. 3

[44] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 2

[45] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2

[46] Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. Back to the feature: Learning robust camera localization from pixels to pose. In *CVPR*, 2021. 2

[47] T. Sattler, B. Leibe, and L. Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012.

[48] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *TPAMI*, 39(9):1744–1756, 2016. 2

[49] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DOF outdoor visual localization in changing conditions. In *CVPR*, 2018. 2

[50] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding the limitations of CNN-based absolute camera pose regression. In *CVPR*, 2019. 2

[51] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2

[52] Yoli Shavit, Ron Ferens, and Yosi Keller. Learning multi-scene absolute pose regression with transformers. In *ICCV*, 2021. 2

[53] 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. 1, 2

[54] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, 2021. 2

[55] Shitao Tang, Sicong Tang, Andrea Tagliasacchi, Ping Tan, and Yasutaka Furukawa. NeuMap: Neural coordinate mapping by auto-transdecoder for camera localization. In *CVPR*, 2023. 3, 5

[56] Mehmet Özgür Türkoğlu, Eric Brachmann, Konrad Schindler, Gabriel Brostow, and Áron Monszpart. Visual camera re-localization using graph neural networks and relative pose supervision. In *3DV*, 2021. 2

[57] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip H. S. Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015. 2

[58] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *3DV*, 2016. 1

[59] Johanna Wald, Torsten Sattler, Stuart Golodetz, Tommaso Cavallari, and Federico Tombari. Beyond controlled environments: 3D camera re-localization in changing indoor scenes. In *ECCV*, pages 467–487, 2020. 6

[60] Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. AtLoc: Attention guided camera localization. In *AAAI*, pages 10393–10401, 2020. 2

[61] Fangjinhua Wang, Xudong Jiang, Silvano Galliani, Christoph Vogel, and Marc Pollefeys. GLACE: Global local accelerated coordinate encoding. In *CVPR*, 2024. 2, 3, 6, 7

[62] Shuzhe Wang, Zakaria Laskar, Iaroslav Melekhov, Xiaotian Li, Yi Zhao, Giorgos Tolias, and Juho Kannala. HSCNet++: Hierarchical scene coordinate classification and regression for visual localization with transformer. *IJCV*, 132(7):2530–2550, 2024. 3

[63] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024. 1, 2, 6

[64] Hongchi Xia, Yang Fu, Sifei Liu, and Xiaolong Wang. RGBD objects in the wild: Scaling real-world 3D object learning from RGB-D videos. In *CVPR*, 2024. 6, 2

[65] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. SANet: Scene agnostic network for camera localization. In *ICCV*, 2019. 3

[66] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A large-scale dataset for generalized multi-view stereo networks. In *CVPR*, 2020. 6, 2

[67] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. ScanNet++: A high-fidelity dataset of 3D indoor scenes. In *CVPR*, 2023. 6, 2

[68] Junwei Zheng, Ruiping Liu, Yufan Chen, Zhenfang Chen, Kailun Yang, Jiaming Zhang, and Rainer Stiefelhagen. Scene-agnostic pose regression for visual localization. In *CVPR*, pages 27092–27102, 2025. 2

[69] Qunjie Zhou, Sérgio Agostinho, Aljoša Ošep, and Laura Leal-Taixé. Is geometry enough for matching in visual localization? In *ECCV*, 2022. 2

ACE-G: Improving Generalization of Scene Coordinate Regression Through Query Pre-Training

Supplementary Material

7. Hyperparameters

Architecture We use DINOv2-L with registers [23, 41] as our image encoder and use $N = 12$ cross-attention-only transformer blocks (*cf*. Fig. 4).

Pre-Training We train our network for 4.4M map code iterations (resulting in 440 000 mapping and 440 000 query iterations for the head, because we are only updating the head in every 10th iteration, *cf*. Sec. 4.2) on 8 A100 GPUs with a scene batch size $N_{\mathrm{spb}} = 200$ and patch batch size $N_{\mathrm{pps}} = 512$. Each map code is optimized between 6000 and 10000 iterations with $N_{\mathrm{qstandby}} = 5000$. To focus optimization on solvable patches we found it beneficial to only use the lowest 30% losses in each batch. We use AdamW for map codes and network weights with a learning rate of 0.0001 without learning rate scheduler. During pre-training we use a map code size of $N_{\mathcal{C}} = 1024$.

Novel Scene Mapping We use slightly varying parameters for our 5 minute and 25 minute configuration following manual tuning on validation scenes. In the 5 minute setup a maximum buffer size of 4M patches, 1000 iterations, and a batch size of 40960 is used. In the 25 minute setup we spend more time budget on the buffer creation using 8M patches, increase the number of iterations to 4000, and increase the batch size to 51200. In both cases, AdamW with a one cycle learning rate schedule with maximum learning rate 0.002 is used. We use $N_{\mathcal{C}} = 4096$ resulting in 12 MB maps (full precision). During optimization we apply dropout on the image features with a dropout probability of 10%.

Table 6. **Image encoder analysis**. Accuracy, in terms of median position error (in cm), on 7Scenes, 12Scenes, Indoor-6 and RIO10 (top) with per-scene results for 7Scenes (bottom) for different image encoders. **Best** and $\underline{\text{second best}}$ highlighted.

	Static		Dynamic	
	7S	12S	I6	R10
ACE w/ ACE enc.	**1.1**	**0.7**	11.0	358.4
ACE w/ DINOv2 enc. (“DINO-ACE”)	7.2	1.9	$\underline{5.6}$	$\underline{83.8}$
ACE-G w/ ACE enc. (Ours)	$\underline{1.3}$	**0.7**	11.3	144.5
ACE-G w/ DINOv2 enc. (Ours)	4.6	$\underline{1.2}$	**4.5**	**41.1**

	Chess	Fire	Heads	Off.	Pump.	RK	Stairs
ACE w/ ACE enc.	**0.6**	**0.8**	**0.6**	$\underline{1.1}$	**1.2**	**0.8**	**2.8**
ACE w/ DINOv2 enc. (“DINO-ACE”)	$\underline{0.9}$	$\underline{1.4}$	$\underline{0.8}$	1.4	1.8	$\underline{1.1}$	43.9
ACE-G w/ ACE enc. (Ours)	**0.6**	**0.8**	**0.6**	**1.0**	$\underline{1.4}$	**0.8**	$\underline{3.8}$
ACE-G w/ DINOv2 enc. (Ours)	1.0	1.5	0.9	1.4	1.8	$\underline{1.1}$	24.5

8. Datasets

Every combination of mapping images yields a unique map code after optimization and every other image in a sequence can potentially aid in improving the generalization performance of the coordinate regressor. Therefore, we randomly generate multiple mapping-query configurations per scene taking into account specific dataset characteristics. For most datasets, sequences are ordered in time which gives a strong clue for which images are likely covisible. Therefore, we follow an interval-based configuration scheme where the sorted image sequence is split into disjoint subsets serving as the mapping and query portion. For unsorted datasets we adjust parameters such that empirically most query views should still be solvable while also including challenging views with little visual overlap.

We follow two sampling schemes: an interspersed one, in which mapping and query intervals of varying length alternate throughout the sequence; and a query-mapping-query scheme, where a mapping interval of varying length is surrounded by two query intervals of varying length.

For ARKitScenes and ScanNet++, we first sample a mapping interval, then find covisible image pairs given the image pair information published by [63] and sample a short interval around the image known to be covisible.

Beyond this interval-based sampling, we randomly switch mapping and query sequences for the MapFree dataset and randomly mirror scenes. Finally, a random rotation is applied every time a new mapping-query configuration is added to the active scenes.

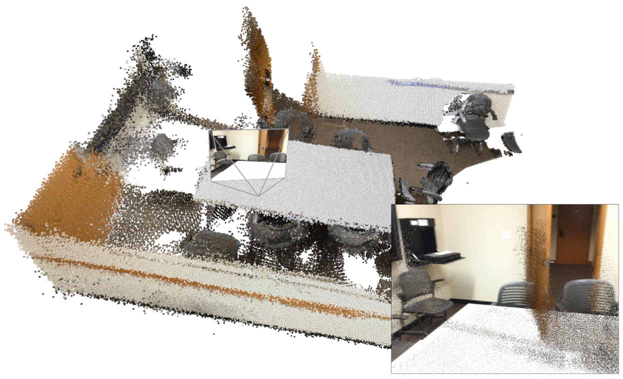
Figure 9 visualizes a mapping-query split for each included dataset.

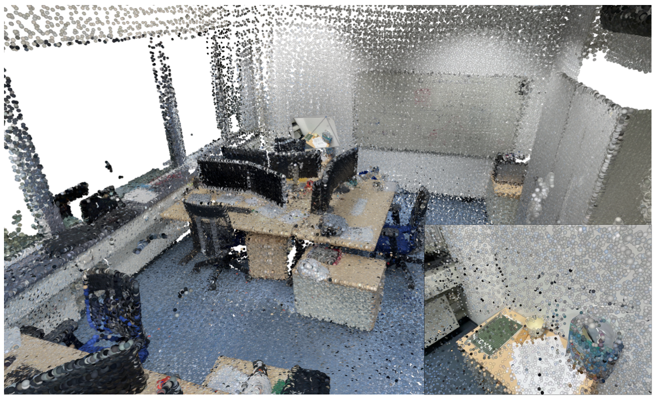
9. Additional Experiments

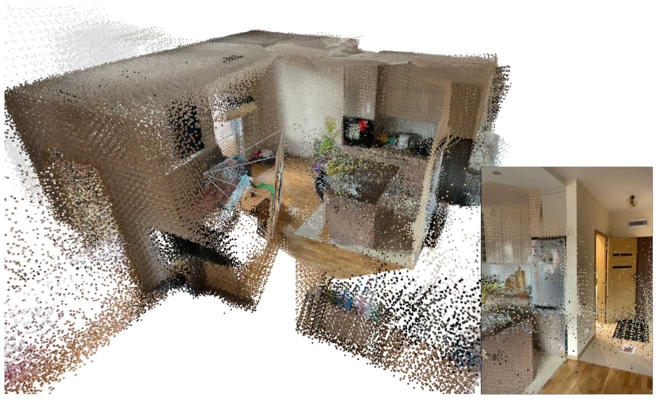
9.1. Image Encoder

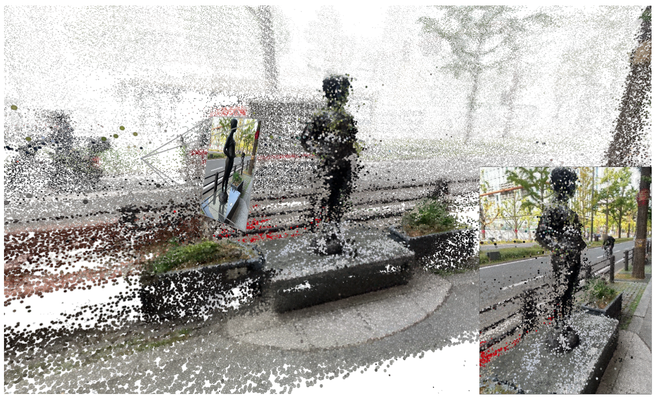
To better understand the interplay of our pre-training and the image encoder, we report additional results of ACE and ACE-G paired with ACE's fully-convolutional image encoder and DINOv2 in Tab. 6. In addition to the datasets reported in the main paper, we include 7Scenes [53] and 12Scenes [58]. Datasets can be grouped into *static* (7Scenes, 12Scenes) and *dynamic* (Indoor-6, RIO10), depending on whether there are environment and lighting changes between mapping and query images.

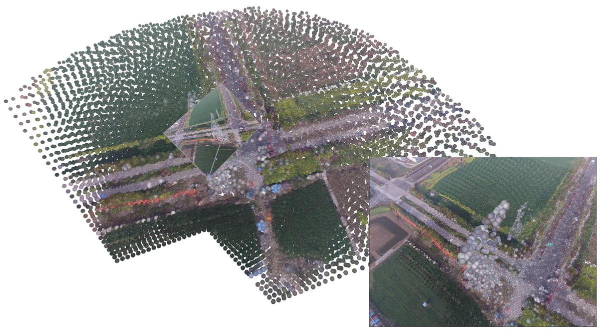
In summary, ACE-G with DINOv2 achieves the most balanced results across datasets. The accuracy of ACE and


(a) ScanNet [22]


(b) ScanNet++ [67]


(c) ARKitScenes [6]


(d) MapFree [2]


(e) BlendedMVG [66]

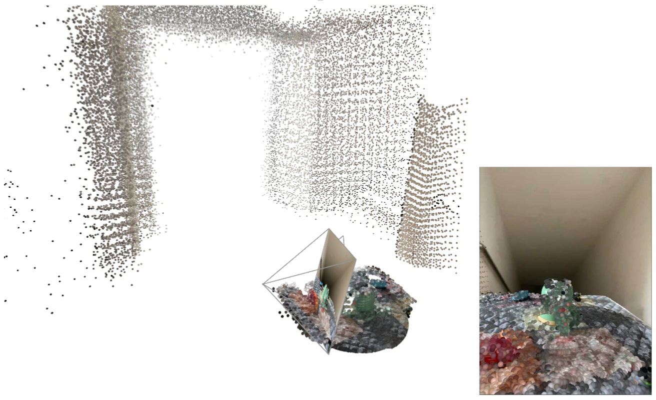

(f) WildRGBD [64]

Figure 9. **Mapping-query splits used for training.** The 3D points show the accumulated scene coordinates from all mapping views. The inset shows projected ground-truth points in a query view. Note that overlap between mapping points and query view varies significantly.

ACE-G depends to some extent on the image features being used. The ACE features works well on static scenes that require little generalization but performs poorly in dynamic scenes. DINOv2 features are less precise compared to ACE features on static scenes, but generalize much better in dynamic scenes.

To further understand the differences on 7Scenes, we also include per-scene results on that dataset (Tab. 6). The performance drop is caused by one scene (Stairs), and can be attributed to DINOv2 features, not to ACE-G's architecture or pre-training.

Notably, ACE-G's architecture and pre-training consistently improves when building on-top of DINOv2 features. The strong performance of ACE on static scenes, comes at the cost of worse performance on dynamic datasets. We believe that the strong performance of ACE-G in dynamic conditions is highly relevant in practice when query images are taken long after an environment has been mapped.

Table 7. **Fine-tuning results.** Accuracy under $(20^\circ, 20\,\mathrm{cm})$ error threshold on the validation splits of the training datasets. Fine-tuning on different dataset combinations can specialize the model for specific conditions.

(%)	SN	SN++	ARK	MF	BMVG	WR
Baseline	63.5	55.5	56.0	47.1	40.6	36.7
Indoor	+1.9	+2.6	+1.2	-3.4	-7.5	-9.2
Outdoor	-8.7	-10.6	-9.6	+0.4	+3.4	-9.0
MapFree	-8.5	-10.1	-9.1	+0.2	-7.1	-8.6

9.2. Fine-Tuning

In Tab. 7 we further show validation results for three specialized models fine-tuned on a subset of datasets for 1M iterations after 4M iterations of pre-training on all 6 datasets: in one case we fine-tune only on indoor datasets, in a second case only on outdoor datasets, and in the final case only on the MapFree dataset. Interestingly, the latter models benefit less from the fine-tuning, which might suggest that the two outdoor training datasets (MapFree and BlendedMVG) are not sufficient for the variety of scenes present in these datasets.