

NeuMap: Neural Coordinate Mapping by Auto-Transdecoder for Camera Localization

Shitao Tang Sicong Tang Andrea Tagliasacchi Ping Tan Yasutaka Furukawa
Simon Fraser University

{shitao.tang, sta105, andrea.tagliasacchi, pingtan, furukawa}@sfu.ca

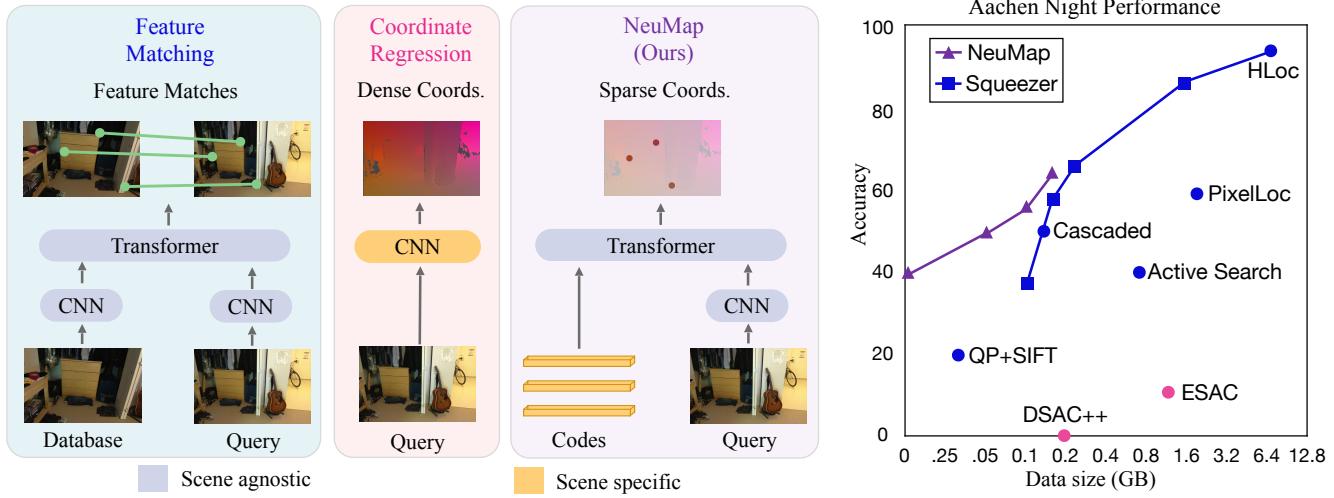


Figure 1. The paper presents neural coordinate mapping (NeuMap) for camera localization. NeuMap encodes a scene into a set of codes and uses a scene-agnostic transformer to decode the coordinates of key-points in a query image. The right compares the localization accuracy and the data size for Aachen Night benchmark. The accuracy is averaged over three translation/rotation error thresholds (0.25m, 2°), (0.5m, 5°), or (5m, 10°). NeuMap and Squeeze control representation sizes and are illustrated by curves with multiple data points.

Abstract

This paper presents an end-to-end neural mapping method for camera localization, encoding a whole scene into a grid of latent codes, with which a Transformer-based auto-decoder regresses 3D coordinates of query pixels. State-of-the-art camera localization methods require each scene to be stored as a 3D point cloud with per-point features, which takes several gigabytes of storage per scene. While compression is possible, the performance drops significantly at high compression rates. NeuMap achieves extremely high compression rates with minimal performance drop by using 1) learnable latent codes to store scene information and 2) a scene-agnostic Transformer-based auto-decoder to infer coordinates for a query pixel. The scene-agnostic network design also learns robust matching priors by training with large-scale data, and further allows us to just optimize the codes quickly for a new scene while fixing the network weights. Extensive evaluations with five benchmarks show that NeuMap outperforms all the other coor-

dinate regression methods significantly and reaches similar performance as the feature matching methods while having a much smaller scene representation size. For example, NeuMap achieves 39.1% accuracy in Aachen night benchmark with only 6MB of data, while other compelling methods require 100MB or a few gigabytes and fail completely under high compression settings. The codes are available at <https://github.com/Tangshitao/NeuMap>.

1. Introduction

Visual localization determines camera position and orientation based on image observations, an essential task for applications such as VR/AR and self-driving cars. Despite significant progress, accurate visual localization remains a challenge, especially when dealing with large viewpoint and illumination changes. Compact map representation is another growing concern, as applications like delivery robots may require extensive maps. Standard visual localization techniques rely on massive databases of keypoints with 3D

coordinates and visual features, posing a significant bottleneck in real-world applications.

Visual localization techniques generally establish 2D-3D correspondences and estimate camera poses using perspective-n-point (PnP) [22] with a sampling method like RANSAC [15]. These methods can be divided into two categories: Feature Matching (FM) [11, 34, 37, 49] and Scene Coordinate Regression (SCR) [3, 5, 7, 40]. FM methods, which are trained on a vast amount of data covering various viewpoint and illumination differences, use sparse robust features extracted from the query image and matched with those in candidate scene images. This approach exploits learning-based feature extraction and correspondence matching methods [13, 35, 42, 44] to achieve robust localization. However, FM methods require large maps, making them impractical for large-scale scenes. Many methods [11, 27, 49] have been proposed to compress this map representation, but often at the cost of degraded performance. On the other hand, SCR methods directly regress a dense scene coordinate map using a compact random forest or neural network, providing accurate results for small-scale indoor scenes. However, their compact models lack generalization capability and are often restricted to limited viewpoint and illumination changes. Approaches such as ESAC [5] handle larger scenes by dividing them into smaller sub-regions, but still struggle with large viewpoint and illumination changes.

We design our method to enjoy the benefits of compact scene representation of SCR methods and the robust performance of FM methods. Similar to FM methods, we also focus on a sparse set of robustly learned features to deal with large viewpoint and illumination changes. On the other hand, we exploit similar ideas to SCR methods to regress the 3D scene coordinates of these sparse features in the query images with a compact map representation. Our method, dubbed neural coordinate mapping (NeuMap), first extracts robust features from images and then applies a transformer-based auto-decoder (i.e., auto-transdecoder) to learn: 1) a grid of scene codes encoding the scene information (including 3D scene coordinates and feature information) and 2) the mapping from query image feature points to 3D scene coordinates. At test time, given a query image, after extracting image features of its key-points, the auto-transdecoder regresses their 3D coordinates via cross attention between image features and latent codes. In our method, the robust feature extractor and the auto-transdecoder are scene-agnostic, where only latent codes are scene specific. This design enables the scene-agnostic parameters to learn matching priors across scenes while maintaining a small data size. To handle large scenes, we divide the scene into smaller sub-regions and process them independently while applying a network pruning technique [25] to drop redundant codes.

We demonstrate the effectiveness of NeuMap with a diverse set of five benchmarks, ranging from indoor to outdoor and small to large-scale: 7scenes (indoor small), ScanNet (indoor small), Cambridge Landmarks (outdoor small), Aachen Day & Night (outdoor large), and NAVER LABS (indoor large). In small-scale datasets (i.e., 7scenes and Cambridge Landmarks), NeuMap compresses the scene representation by around 100-1000 times without any performance drop compared to DSAC++. In large-scale datasets (i.e., Aachen Day & Night and NAVER LABS), NeuMap significantly outperforms the current state-of-the-art at high compression settings, namely, HLoc [34] with a scene-compression technique [49]. In ScanNet dataset, we demonstrate the quick fine-tuning experiments, where we only optimize the codes for a new scene while fixing the scene-agnostic network weights.

2. Related work

Visual localization with feature matching (FM). FM-based localization has achieved state-of-the-art performance [13, 14, 32, 34, 35, 37, 42, 49]. Torsten et al. [37] match query images exhaustively with all 3D points in structure-from-motion models. However, as scenes grow larger, this matching becomes ambiguous, compromising localization robustness. Paul-Edouard [35] proposes a coarse-to-fine strategy: 1) coarsely localizing query images using a global image feature [1], and 2) computing camera poses through local key-point matches. This pipeline has demonstrated significant improvements, with most follow-up works focusing on enhancing feature matching capability [35, 42, 49, 50]. SuperGlue [35] utilizes Transformers to match key-point sets, achieving impressive results. LoFTR [42] and its variants [9, 44] propose dense matching frameworks without key-points. Nevertheless, these methods require storing large databases of key-point features or images. SceneSqueezer [49] offers a compression mechanism for FM-based methods by removing redundant images and key-points, but performance drops significantly at high compression settings. Our approach employs latent codes to store and retrieve key-point information, naturally achieving high compression rates.

Scene coordinate regression (SCR). Convolutional neural networks have been used to regress dense coordinate maps from RGB images for localization [3, 5–7, 23, 47]. Due to limited network capacity, this approach does not scale well to large scenes. ESAC [5] addresses scalability by dividing large scenes into smaller sub-regions, training a separate network for each. HSCNet [23] classifies pixels into corresponding sub-regions to reduce ambiguity. Although SCR-based methods have a smaller representation size than FM-based methods, they are less robust against large viewpoint changes or illumination differences. This is because acquiring SCR training data is expensive, whereas FM train-

ing data is easily accessible from extensive databases of feature correspondences in challenging conditions [12, 24]. Furthermore, SCR approaches require model training for each new scene, which is undesirable in many applications. While SANet [48] and DSM [43] generalize this approach to unseen scenes, they also result in reduced localization accuracy. Our approach is SCR-based, utilizing a scene-agnostic feature extractor and transformer trained as in FM-based methods, offering both robustness and compact representation.

Auto-decoder. An auto-decoder optimizes latent codes directly via back-propagation within a decoder-only framework [2, 30, 41]. Piotr et al. [2] employ an auto-decoder for generative adversarial networks. DeepSDF [30] and SRN [41] store shape representations into latent codes. However, MLP or CNN-based auto-decoders have limited capacity using a single code. Transformer-based auto-decoders (i.e., auto-transdecoders) employ an arbitrary number of codes to increase capacity. Recently, Sandler et al. [33] augment a transformer with learnable memory tokens (codes) that allow the model to adapt to new tasks while optionally preserving its capabilities on previously learned tasks. The transformer weights are first pre-trained with a large-scale dataset, and the tokens (codes) are then fine-tuned in downstream tasks. This memory mechanism resembles our auto-transdecoder, but we optimize both transformer and codes simultaneously during training. To the best of our knowledge, this work is the first to use an auto-transdecoder for mapping and localization tasks.

3. Method

Given a scene represented by a set of reference images $\{\mathbf{I}_n\}$ with known camera calibrations $\{\mathbf{T}_n\}$ expressed in a (scene-specific) coordinate frame, we devise a technique that encodes these images into a *compact* scene representation \mathcal{S} . We employ this scene representation to perform *visual localization*, a task of predicting the camera \mathbf{T}_q of a query image \mathbf{I}_q that was never seen before – with a vantage point and/or appearance different from the one in the reference image set.

3.1. Overview

We achieve visual localization by solving the proxy task of *scene coordinate regression* on *sparse features*, where given a set of 2D key-point $\{\mathbf{k}_i\}$ extracted from \mathbf{I}_q , we predict its corresponding 3D scene coordinate $\{\mathbf{K}_i\}$ (Section 3.2). As shown in Figure 2, our method extracts 2D key-points $\{\mathbf{k}_i\}$ following HLoc [34] with a pre-trained backbone, R2D2 [32]. In order to determine their scene coordinates, we first compute a feature map of the image \mathbf{I}_q with a trainable CNN backbone \mathcal{F} , which is a ResNet18 [17] and bilinearly interpolates the feature, and then solve the scene coordinate by a decoder \mathcal{D} as,

$$\mathbf{K}_i = \mathcal{D}(\mathcal{F}(\mathbf{I}_q, \mathbf{k}_i), \mathcal{S}) \quad (1)$$

Here, \mathcal{S} is the learned scene representation. Finally, we obtain camera localization by solving a perspective-n-point (PnP) problem [22] (Section 3.4) with 2D-3D correspondences from $\{\mathbf{k}_i \leftrightarrow \mathbf{K}_i\}$.

3.2. Sparse Scene Coordinate Regression

Scene Representation. We first build a scene model to facilitate localization. Given a set of reference images, we extract 2D key-points by R2D2 [32] and compute their corresponding 3D scene coordinates by COLMAP (Section 3.3). Some of the key-points are not successfully triangulated, so their 3D coordinates are invalid. We cache 3D scene points in a *sparse grid* with voxels of uniform side length l as shown in Figure 2. We denote the v -th voxel as \mathcal{V}_v . We then model the reference scene as a *set* of latent codes $\mathcal{S} = \{\mathbf{S}_v\}$ – *one per voxel* – and modify Equation 1 to reflect this design choice as:

$$(\mathbf{K}_i^v, c_i^v) = \mathcal{D}(\mathcal{F}(\mathbf{I}_q, \mathbf{k}_i), \mathbf{S}_v) \quad (2)$$

Here, the scalar parameter c_i^v is the confidence that the key-point \mathbf{k}_i is in the voxel \mathcal{V}_v , and \mathbf{K}_i^v is the scene coordinate under the voxel attached coordinate frame, i.e., $\mathbf{K}_i = \mathbf{K}_i^v + \mathbf{O}_v$ and \mathbf{O}_v is the mean of all the coordinates in \mathcal{V}_v .

This formulation simultaneously solves a classification and regression problem: 1) classifying whether a 2D key-point belongs to a 3D voxel and 2) regressing its 3D position *locally* within the voxel.

Sparsity. Inspired by network pruning [25], we apply code pruning to remove redundant codes. Specifically, we multiply each voxel code in \mathbf{S}_v with a scaling factor w_v , and jointly learn the codes with these scaling factors, with $L1$ loss imposed on w_v . After finishing training, we prune codes whose weights are below a threshold and finetune the remaining codes.

Scene Coordinate Regression. We use a set of per-voxel latent codes \mathbf{S}_v to facilitate the learning of scene coordinate regression. The decoder \mathcal{D} is a stacked transformer to regress the scene coordinates of the 2D image key-points. We include T transformer blocks, (6 blocks in our implementation), defined by the inductive relationship (see Figure 2) as,

$$\mathbf{f}_i^{(t+1)} = \text{CrAtt}(\mathbf{f}_i^t, \tilde{\mathbf{s}}_v^t) \quad (3)$$

$$\tilde{\mathbf{s}}_v^t = w_v^t \mathbf{s}_v^t \quad (4)$$

where the feature $\mathbf{f}_i^{(1)} = \mathcal{F}(\mathbf{I}_q, \mathbf{k}_i)$, w_v^t is the scaling factor enforcing sparsity. Each transformer block contains a set of codes \mathbf{s}_v^t , and the final per voxel codes are $\mathbf{S}_v = \{\mathbf{s}_v^t, 1 \leq t \leq T\}$. The function $\text{CrAtt}(\cdot, \cdot)$ is classical

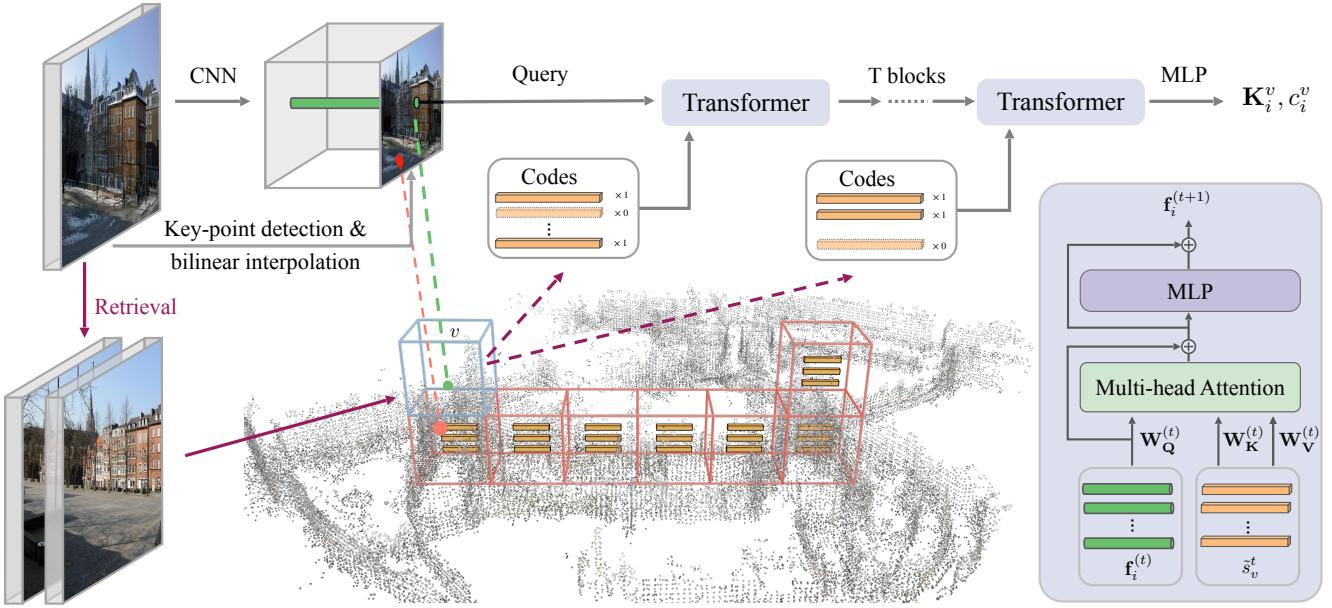


Figure 2. **Overview** – We divide a scene into sub-regions and assign codes to each region. A shared convolutional neural network extracts image features, and a Transformer network decodes coordinates from the codes. Code-pruning further reduces the data size.

cross-attention from transformers [45]:

$$\mathbf{Q} = \mathbf{W}_\mathbf{Q}^{(t)} \cdot \mathbf{f}_i^t, \mathbf{K} = \mathbf{W}_\mathbf{K}^{(t)} \cdot \tilde{\mathbf{s}}_v^t, \mathbf{V} = \mathbf{W}_\mathbf{V}^{(t)} \cdot \tilde{\mathbf{s}}_v^t \quad (5)$$

$$\mathbf{f}_i^{(t+1)} = \text{MLP}(\text{softmax}(\mathbf{Q} \cdot \mathbf{K}) \cdot \mathbf{V}) \quad (6)$$

At the end of the stacked transformer, we apply another MLP to compute the scene coordinate and confidence as,

$$(\mathbf{K}_i^v, c_i^v) = \text{MLP}(\mathbf{f}_i^{(T)}) \quad (7)$$

3.3. Training

At training time, we learn the weights of the feature encoder \mathcal{F} , the decoder \mathcal{D} , and the *compressed* scene representation for all training scenes $\{\mathcal{S}_s\}$ via generative latent optimization / auto-decoding [30] with the following loss:

$$\mathcal{L} = \lambda_x \cdot \mathcal{L}_x + \lambda_c \cdot \mathcal{L}_c + \lambda_{L1} \cdot \mathcal{L}_{L1}. \quad (8)$$

We set the loss weights λ_x , λ_c , and λ_{L1} to 1.

The scene coordinate loss is defined as,

$$\mathcal{L}_x = \sum_v \sum_i (y_i^v) \cdot \|(\mathbf{K}_i^v + \mathbf{O}_v) - \mathbf{K}_i^*\|_2 \quad (9)$$

where \mathbf{K}_i^* is the scene coordinate triangulated by COLMAP for key-point \mathbf{k}_i , the binary variable y_i^v indicates whether the triangulated scene coordinate \mathbf{K}_i^* belongs to the voxel \mathcal{V}_v . The key-points with no valid 3D coordinates (not successfully triangulated) do not belong to any voxel.

The second term is a classification loss, i.e., a binary cross entropy, for the confidence c_i^v ,

$$\mathcal{L}_c = \sum_v \sum_i BCE(c_i^v, y_i^v) \quad (10)$$

The third term enforces *sparsity* and produces a *compressed* representation, which is defined as,

$$\mathcal{L}_{L1} = \sum_v \sum_{t=1}^T \|w_v^t\|_1, \quad (11)$$

where w_v^t is the sparsity factor in Equation 4.

Training strategy. We learn the decoder \mathcal{D} , the CNN backbone \mathcal{F} , and the scene representation \mathcal{S} with voxel sampling. At each iteration, we randomly choose B voxels, where B is the batch size. Each voxel \mathcal{V}_v has a set of reference images \mathbf{I}_v , each of which contains at least 20 scene points in \mathcal{V}_v . We then sample one training image for each voxel and optimize \mathcal{D} , \mathcal{F} , and the scene codes of sampled voxels by minimizing the training loss in Equation 8. We sample voxels without replacement, so all scene codes are updated once at each epoch.

Similarly to network pruning [25], we minimize the training loss to convergence, set sparsity factors w_v^t whose values are below a certain threshold to zero, and fine-tune our model while keeping w_v^t frozen.

Scene adaption. Given a new scene (i.e., not in the training data), we simply optimize the scene code \mathcal{S} , while keeping decoder \mathcal{D} and CNN backbone \mathcal{F} fixed. In this way, our scene representation \mathcal{S} is scene specific, but the decoder \mathcal{D} and feature extractor \mathcal{F} are scene agnostic.

3.4. Inference

Given a query image \mathbf{I}_q , we use an existing deep image retrieval method [1,31] to retrieve the most similar reference

images, which activate a subset of voxels; see Figure 2. A voxel \mathcal{V}_v is activated if one of the retrieved reference images contains at least 20 scene points in \mathcal{V}_v . For large-scale scenes, we typically get around 100-200 voxels, while for small-scale scenes, we consider all the voxels without image retrieval. We then extract a set of 2D key-points $\{\mathbf{k}_i\}$ within \mathbf{I}_q , and for each of them, regress their per-voxel confidence and positions via Equation 2. We discard points with confidence $c < 0.5$. All the remaining points are used to compute the camera pose with the PnP algorithm combined with RANSAC, implemented in Pycolmap [28].

4. Experiments

We conduct extensive evaluations with twelve competing methods on five benchmarks.

4.1. Datasets

We use 7scenes [40], Cambridge landmarks [20], Aachen Day & Night [39], NAVER LABS datasets [21], and ScanNet [12] (See Fig. 3). For 7scenes, Cambridge, Aachen, and ScanNet, we train a separate model for each dataset. Within each dataset, all the scenes share the same network parameters. For NAVER, we train one model for one scene due to the large size of the scene codes, which cannot fit in GPU memories.

Additionally, for 7scenes, Cambridge, and NAVER, we train models from scratch. For Aachen, with images that have significant view and illumination differences, we use a feature extractor from LoFTR [42], pretrained with images containing such variations. Although ScanNet is not a standard localization dataset, it shares similarities with 7scenes. To illustrate our scene-agnostic network and scene-specific code design, we use 80 scenes to train the scene-agnostic network and fine-tune the codes on 1 scene (scene0708) for ScanNet.

We demonstrate code pruning for 7scenes, Cambridge, and Aachen, with pruning thresholds of 0.00003, 0.05, and 0.2, respectively. We use median translation/rotation errors to evaluate 7scenes, Cambridge, and ScanNet. For Aachen and NAVER, the accuracy under different thresholds is the standard metric used.

4.2. Implementation details

We employ ResNet-18 [17] to extract image features and train models with 8 V100 GPUs using a batch size of 256. We set the initial learning rate to 0.002 for training the scene-agnostic parameters and 0.0001 for training the scene-specific codes. After every 30 epochs, we multiply the learning rate by 0.5. We train the model for 200 epochs in the first stage and fine-tune the model for 100 epochs in the second stage. For training data generation, we use r2d2 [32] to extract keypoints and perform triangulation to obtain coordinates in Cambridge, Aachen, and NAVER. We

	Size	Aachen Day	Aachen Night
FM	AS [38]	0.75	57.3 / 83.7 / 96.6
	HLoc [34]	7.82	89.6 / 95.4 / 98.8
	Cascaded [11]	0.14	76.7 / 88.6 / 95.8
	QP+R.Sift [27]	0.03	62.6 / 76.3 / 84.7
	Squeezer [49]	0.24	75.5 / 89.7 / 96.2
E2E	PixLoc [36]	2.13	64.3 / 69.3 / 77.4
	ESAC(50) [5]	1.31	42.6 / 59.6 / 75.5
	Ours (8m, 100)	1.26	80.8 / 90.9 / 95.6
	Ours (10m, 1)	.006	49.3 / 75.2 / 93.9
	Ours (10m, *)	0.17	76.2 / 88.5 / 95.5

Table 1. Results of Aachen Day & Night. We report accuracy under different error thresholds. The first number in the bracket is the sub-region length. The second number is the number of codes for each transformer block, and the notation (*) indicates the application of code pruning. Data sizes in the "Size" column are reported in Gigabytes. The red texts show methods with scene compression. FM means feature matching. E2E means end-to-end. The same applies to the following figures. We outperform all the other E2E methods by a large margin, achieving similar performance as Squeezer, which applies scene compression to HLoc. The cyan and orange colors indicate the best and the second best methods, respectively, in each column.

use depth images to acquire 3D coordinates in 7scenes and ScanNet.

4.3. Main results with code-pruning

We present the results after code-pruning for 7scenes, Cambridge, and Aachen datasets.

Aachen Day & Night. Table 1 compares NeuMap with other methods on this large-scale outdoor dataset. We implement three versions of NeuMap by varying the size of voxels (either 8 or 10 meters) with or without code-pruning. The version with 8 meters without code-pruning serves as the performance upper-bound.

Existing coordinate regression methods (i.e., ESAC or DSAC++) perform poorly on this dataset. DSAC++ cannot fit the whole scene into a single network. ESAC is not robust to view and illumination differences. NeuMap is the first coordinate regression method with competitive results in large-scale outdoor datasets. NeuMap also outperforms another non-feature matching-based method, PixLoc, by a significant margin. Cascaded [11], QP+R.Sift [27], and Squeezer [49] are scene compression methods based on feature matching. NeuMap outperforms Cascaded and QP+R.Sift by a considerable margin. Both methods utilize numeric quantization for compression (converting 32-bit floats to 8-bit integers), which is orthogonal to our scheme and would likely further improve our performance once combined.

Squeezer [49] is built on HLoc [34], which extracts and matches keypoints using SuperPoint [13] and SuperGlue [35]. Squeezer removes redundant keypoints by solv-

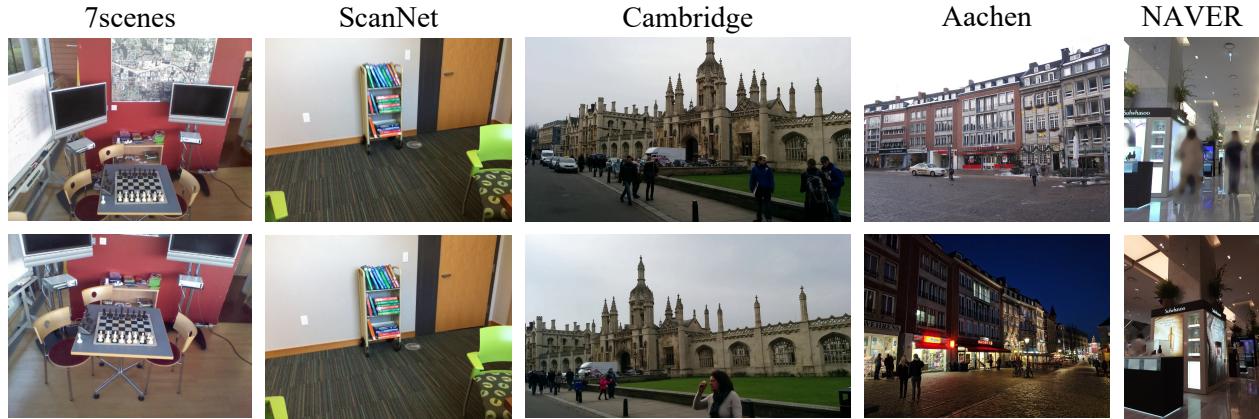


Figure 3. Sample images from 7scenes, ScanNet, Cambridge, Aachen Day & Night, and NAVER LABS datasets.

	7scenes	size (GB)	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs
FM	Active Search [37]	>0.5	1.96, 0.04	1.53, 0.03	1.45, 0.02	3.61, 0.09	3.10, 0.08	3.37, 0.07	2.22, 0.03
	HLoc [34]	>1.0	0.79, 0.02	0.87, 0.02	0.92, 0.02	0.91, 0.03	1.12, 0.05	1.25, 0.04	1.62, 0.06
E2E	DSAC++ [4]	>0.2	0.5 , 0.02	0.9 , 0.02	0.8 , 0.01	0.7 , 0.03	1.1 , 0.04	1.1 , 0.04	2.6 , 0.09
	SANet [48]	>1.0	0.88, 0.03	1.12, 0.03	1.48, 0.02	1.00, 0.03	1.21, 0.04	1.40, 0.04	4.59, 0.16
	DSM [43]	>1.0	0.68, 0.02	0.80, 0.02	0.80, 0.01	0.78, 0.03	1.11, 0.04	1.11, 0.03	1.16, 0.04
	Ours	< 0.002	0.81, 0.02	1.11, 0.03	1.17, 0.02	0.98, 0.03	1.11, 0.04	1.33, 0.04	1.12, 0.04

Table 2. Results of 7scenes. We report the median translation and rotation errors in ($^{\circ}$, m). NeuMap achieves similar performance as other methods with significantly smaller representation sizes.

ing quadratic programming and achieves impressive compression at the expense of performance. We use SuperPoint [13] features (float32) for Squeezer. The NeuMap with code pruning (10 meters) achieves similar performance as Squeezer with a 70 Megabytes smaller representation size. Even at extremely high compression ratios, NeuMap obtains good results, where the total mapping size is only 6 Megabytes. Squeezer fails at such ratios, as shown in Fig. 1.

Cambridge. Table 3 compares performance on the Cambridge dataset. We use triangulated points for supervision and divide each scene into voxels of 200 meters in length. NeuMap achieves similar performance as DSAC++ with a 100 to 1000 times smaller representation size. Our compression design is simple (i.e., latent codes with auto-transdecoder) compared to Squeezer, which has a complex pipeline.

7scenes. Table 2 shows the results, where we use depth images for supervision and train models from scratch. A scene is divided into voxels of 3 meters in length. NeuMap achieves the same performance as DSAC++, HLoc, and DSM with a 200-1000 times smaller data size.

4.4. Main results without code-pruning

NAVER is a large-scale indoor localization dataset. Each scene contains approximately 20,000 images. We use 3 scenes (Dept. B1), (Dept. 1F), and (Dept. 4F) for exper-

iments. We divide each scene into voxels of 4 meters in length, which results in approximately 2,000 voxels, and assign 50×6 codes to each voxel, and train a model from scratch without code pruning. Table 4 compares NeuMap against three methods. Each method is trained with "training images" and evaluated with "validation images" of their dataset. NeuMap outperforms the state-of-the-art coordinate regression method ESAC by a large margin with a much smaller representation size. In comparison to feature matching methods (D2Net and R2D2), NeuMap has similar accuracy when the error thresholds are (0.25m, 2°) and (1m, 5°), while the representation size is 200 times more compact even without code pruning.

4.5. Evaluating code fine-tuning

Our scene-agnostic network and scene-specific code design allows us to fine-tune only the codes for a new scene while fixing the network weights. We use the ScanNet dataset for experiments. We train NeuMap with 1, 10, 20, 40, or 80 scenes and choose 1 new testing scene for fine-tuning and evaluation. For each scene, we randomly sample 10% of the frames for testing, while the rest becomes the training set.

Figure 4 shows the median translation and the median rotation errors for the fully-trained (i.e., optimizing both network weights and codes) and the fine-tuned NeuMap

	ShopFacade	OldHospital	College	Church	Court
FM	Active search [38]	(1.12, 0.12) / 38.7	(1.12, 0.52) / 140	(0.70, 0.57) / 275	(0.62, 0.22) / 359
	HLoc [34]	(0.20 , 0.04) / 316	(0.30 , 0.15) / 1335	(0.20 , 0.12) / 1877	(0.21 , 0.07) / 2007
	Hybrid [8]	(0.54, 0.19) / 0.16	(1.01, 0.75) / 0.62	(0.59, 0.81) / 1.01	(0.49, 0.50) / 1.34
	QP+RootSIFT [27]	(1.40, 0.72) / 0.41	(2.17, 0.90) / 1.1	(1.09, 1.53) / 2.20	(0.89, 0.56) / 3.30
	Squeezer [49]	(0.38, 0.11) / 1.04	(0.57, 0.37) / 4.03	(0.38, 0.27) / 2.4	(0.37, 0.15) / 7.97
E2E	DSAC++ [4]	(0.3 , 0.06) / 207	(0.3 , 0.2) / 207	(0.3 , 0.18) / 207	(0.4 , 0.13) / 207
	HSCNet [4]	(0.3 , 0.06) / 207	(0.3 , 0.19) / 207	(0.3 , 0.18) / 207	(0.30, 0.09) / 207
	DSM [43]	(0.30, 0.06) / 27	(0.23, 0.38) / 105	(0.35, 0.19) / 143	(0.34, 0.11) / 174
	Ours	(0.25 , 0.06) / 0.3	(0.36 , 0.19) / 0.2	(0.19 , 0.14) / 0.3	(0.53, 0.17) / 0.4

Table 3. Results of Cambridge. Each cell reports the median translation error, the median rotation error, and the scene representation size in $\{(\text{°}, \text{m}) / \text{MB}\}$. Red indicates scene compression methods. NeuMap achieves similar accuracy as coordinate regression and feature matching-based methods but with significantly smaller representation sizes.

		Dept. B1	Dept. 1F	Dept. 4F
FM	D2Net [14]	(70.2 / 78.0 / (86.1) / 505GB	(83.2 / 89.2 / (94.5) / 398GB	(72.1 / 85.3 / 98.5 / 183GB
	R2D2 [32]	(71.9 / (77.8 / 87.9) / 210GB	(85.8 / (89.9 / 94.4) / 166GB	(72.6 / 84.6 / 98.3) / 76GB
E2E	ESAC (50) [5]	(5.4 / 9.1 / 14.2) / 1.3GB	(49.7 / 71.5 / 84.1) / 1.3GB	(45.2 / 69.9 / 85.1) / 1.3GB
	Ours	(46.0 / 66.5 / 79.8) / 0.8GB	(75.5 / 88.2 / 95.8) / 0.7GB	(70.4 / 85.4 / 99.0) / 0.4GB

Table 4. Results of NAVER LAB. We report accuracy under different error thresholds (0.1m, 1°), (0.25m, 2°), (1m, 5°).

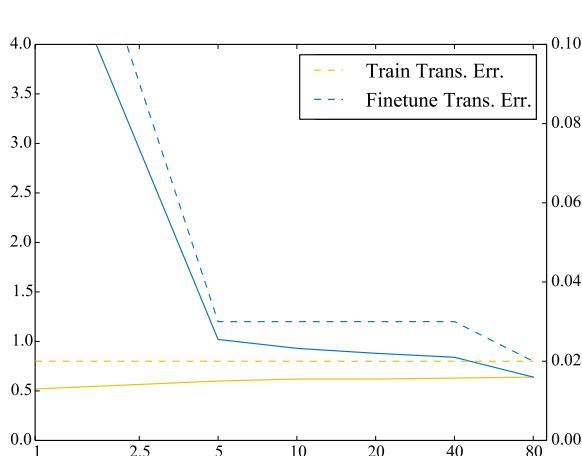


Figure 4. ScanNet results in terms of the median translation and rotation errors (m, °). We train our model with different numbers of training scenes and finetune tokens on one new testing scene.

models. The fully-trained models’ accuracy does not drop as the number of training scenes increases, showing that NeuMap handles an arbitrary number of scenes without a performance drop. On the other hand, fine-tuned models improve accuracy as more training scenes are used to learn a generalizable scene-agnostic network.

4.6. Ablation studies and visualization

We conduct several ablation studies and show visualization results for NeuMap.

Code num.	Day	Night
1	49.3 / 75.2 / 93.9	14.3 / 26.5 / 76.5
25	72.2 / 85.4 / 95.4	21.4 / 49.0 / 82.7
50	75.6 / 88.8 / 95.5	33.6 / 59.1 / 83.7
100	78.0 / 89.0 / 95.2	42.9 / 61.2 / 80.6
Pruned (25)	76.2 / 88.5 / 95.5	37.8 / 62.2 / 87.8

Table 5. Performance under different code numbers. We use a voxel of 10 meters to divide the scene on Aachen day & night dataset and list the code number per transformer block. There are 933 voxels and 6 transformer blocks. It is noted that our network can output good results even with 1 code. For code pruning, there are 25 tokens on average for each transformer block.

Block num.	Day	Night
1	62.1 / 82.8 / 94.4	17.3 / 36.7 / 76.5
3	68.9 / 85.7 / 94.5	20.4 / 40.8 / 79.6
6	78.0 / 89.0 / 95.2	42.9 / 61.2 / 80.6

Table 6. Localization accuracy under different transformer blocks. The performance improves as more blocks are utilized.

Code number and code-pruning. Table 5 shows how the number of codes influences the accuracy of the Aachen Day & Night dataset. NeuMap obtains good results even with 1 code, while the performance improves as we add more codes. The table also shows the results of code pruning, which reduces the number of codes from 100 to 25 on average while maintaining similar accuracy. The pruned version with 25 codes is far superior to the counterpart without pruning, which was trained with 25 codes from the start.

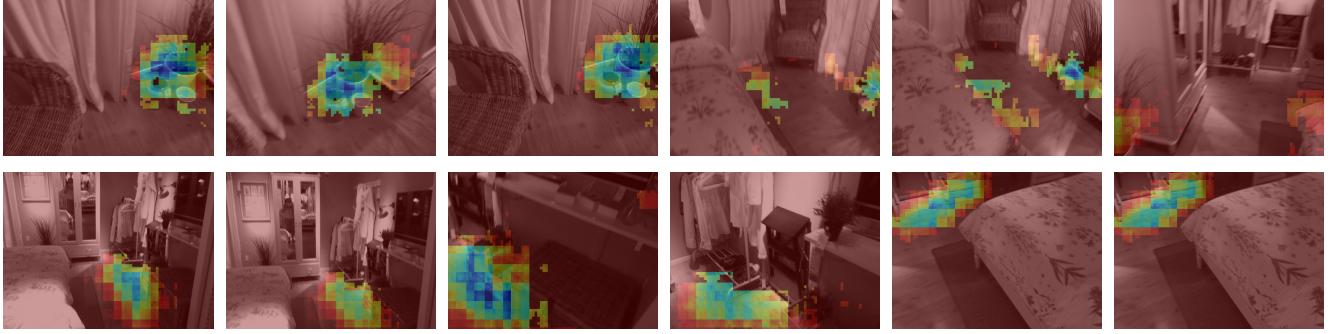


Figure 5. Attention score visualization. We choose one specific code, compute the attention scores for all pixels, and visualize scores by normalizing all the pixels. Each row is the visualization results for the same code. We can see that each code memorizes a specific area.

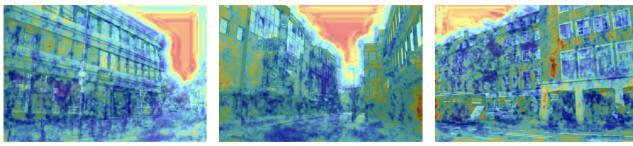


Figure 6. Visualization of activated regions. Deep blue delineates larger attention scores, while orange corresponds to smaller scores.

Attention score visualization. Figure 5 shows the attention scores between one code and image features in different views. The score for pixel i and code j is computed as

$$s_i^j = [\text{softmax}(\mathbf{Q} \cdot \mathbf{K})]^j. \quad (12)$$

$[.]^j$ is the j^{th} element of the vector. \mathbf{Q}, \mathbf{K} is the same as Equation 5. We normalize the scores across an image by

$$\tilde{s}_i^j = \frac{((s_i^j) - a)}{a - b}. \quad (13)$$

a and b are the minimum and maximum values in an image. A code activates and stores information for consistent pixels corresponding to the same 3D scene structures.

Correlated region visualization. In Fig. 6, we visualize the most correlated regions using a heat map. We computed attention scores for each feature point on a dense feature map. The visualization score for feature point i is determined by the maximum score among all codes, expressed as $\max_j s_i^j$, where j represents the codes. After normalizing the scores across the entire feature map, we blended them with the RGB image for a comprehensive visualization. This heat map highlights the most significant regions in the scene.

Transformer block number. Table 6 shows the results under different transformer block numbers. The performance improves as we use more transformer blocks.

5. Future Work and Limitations

NeuMap is a simple yet effective localization method with many potentials. Since our framework is end-to-end

and fully differentiable, conventional network acceleration technology can be applied for better efficiency, such as code pruning [26, 29], code quantization [46, 51], and knowledge distillation [16, 18]. Besides, the accuracy could also be further boosted by pose loss, commonly used in object pose estimation [10, 19].

A major limitation is the inference speed for large scenes. For high accuracy, a voxel needs to be smaller, increasing the number of voxels to be retrieved. While the coordinate regression is end-to-end, our approach runs the decoder many times for the retrieved voxels and is not necessarily fast (e.g., 5 seconds per image for Aachen). Our future work is better voxel division and retrieval methods for speed up. Another future work is cross-dataset training of NeuMap, which currently degrades performance.

6. Conclusions

This paper proposes a novel camera localization method that encodes a scene into a grid of latent codes. Our framework consists of a scene-agnostic neural network and scene-specific latent code. To handle large scenes, we divide a scene into a grid of voxels and assign codes to each. Our method outperforms all the other end-to-end methods by a large margin and achieves similar performance as the feature-matching methods with a much smaller scene representation size. We will share all our code and models.

Acknowledgement. We thank Martin Humenberger and Philippe Weinzaepfel for their codes of ESAC in NAVER LAB datasets. We thank Luwei Yang to evaluate Squeezed under different data sizes. We thank Jiahui Zhang and Jiacheng Chen for proof reading. The research is supported by NSERC Discovery Grants, NSERC Discovery Grants Accelerator Supplements, DND/NSERC Discovery Grant Supplement, and John R. Evans Leaders Fund (JELF).

References

- [1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016. 2, 4
- [2] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017. 3
- [3] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6684–6692, 2017. 2
- [4] Eric Brachmann and Carsten Rother. Learning less is more: 6d camera localization via 3d surface regression. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4654–4662, 2018. 6, 7
- [5] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7525–7534, 2019. 2, 5, 7
- [6] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4322–4331, 2019. 2
- [7] Eric Brachmann and Carsten Rother. Visual camera re-localization from rgb and rgb-d images using dsac. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2
- [8] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid scene compression for visual localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7653–7662, 2019. 7
- [9] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David Mckinnon, Yanghai Tsin, and Long Quan. Aspanformer: Detector-free image matching with adaptive span transformer. *arXiv preprint arXiv:2208.14201*, 2022. 2
- [10] Hansheng Chen, Pichao Wang, Fan Wang, Wei Tian, Lu Xiong, and Hao Li. Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2781–2790, 2022. 8
- [11] Wentao Cheng, Weisi Lin, Kan Chen, and Xinfeng Zhang. Cascaded parallel filtering for memory-efficient image-based localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1032–1041, 2019. 2, 5
- [12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 3, 5
- [13] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabenovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018. 2, 5, 6
- [14] 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. 2, 7
- [15] 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*, 24(6):381–395, 1981. 2
- [16] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021. 8
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3, 5
- [18] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 8
- [19] Shun Iwase, Xingyu Liu, Rawal Khirodkar, Rio Yokota, and Kris M Kitani. Repose: Fast 6d object pose refinement via deep texture rendering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3303–3312, 2021. 8
- [20] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Poseonet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 5
- [21] Donghwan Lee, Soohyun Ryu, Suyong Yeon, Yonghan Lee, Deokhwa Kim, Cheolho Han, Yohann Cabon, Philippe Weinzaepfel, Nicolas Guérin, Gabriela Csurka, et al. Large-scale localization datasets in crowded indoor spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3236, 2021. 5
- [22] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155–166, 2009. 2, 3
- [23] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11983–11992, 2020. 2
- [24] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050, 2018. 3
- [25] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017. 2, 3, 4

- [26] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018. 8
- [27] Marcela Mera-Trujillo, Benjamin Smith, and Victor Fragoso. Efficient scene compression for visual-based localization. In *2020 International Conference on 3D Vision (3DV)*, pages 1–10. IEEE, 2020. 2, 5, 7
- [28] Dusmanu Mihai, Lindenberger Philipp, Lambert John, and Sarlin Paul-Edouard. Pycolmap. <https://github.com/colmap/pycolmap>, 2013. 5
- [29] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Froissard, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 8
- [30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 3, 4
- [31] Jerome Revaud, Jon Almazán, Rafael S Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5107–5116, 2019. 4
- [32] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: repeatable and reliable detector and descriptor. *arXiv preprint arXiv:1906.06195*, 2019. 2, 3, 5, 7
- [33] Mark Sandler, Andrey Zhmoginov, Max Vladymyrov, and Andrew Jackson. Fine-tuning image transformers using learnable memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12155–12164, 2022. 3
- [34] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. 2, 3, 5, 6, 7
- [35] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 2, 5
- [36] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3247–3257, 2021. 5
- [37] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In *European conference on computer vision*, pages 752–765. Springer, 2012. 2, 6
- [38] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1744–1756, 2016. 5, 7
- [39] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *BMVC*, volume 1, page 4, 2012. 5
- [40] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgbd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2930–2937, 2013. 2, 5
- [41] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *Advances in Neural Information Processing Systems*, 32, 2019. 3
- [42] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021. 2, 5
- [43] Shitao Tang, Chengzhou Tang, Rui Huang, Siyu Zhu, and Ping Tan. Learning camera localization via dense scene matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1831–1841, 2021. 3, 6, 7
- [44] Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers. *arXiv preprint arXiv:2201.02767*, 2022. 2
- [45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4
- [46] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019. 8
- [47] Xin Wu, Hao Zhao, Shunkai Li, Yingdian Cao, and Hongbin Zha. Sc-wls: Towards interpretable feed-forward camera re-localization. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I*, pages 585–601. Springer, 2022. 2
- [48] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. Sanet: Scene agnostic network for camera localization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 42–51, 2019. 3, 6
- [49] Luwei Yang, Rakesh Shrestha, Wenbo Li, Shuaicheng Liu, Guofeng Zhang, Zhaopeng Cui, and Ping Tan. Squeezee: Learning to compress scene for camera relocalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8259–8268, 2022. 2, 5, 7
- [50] Jiahui Zhang, Shitao Tang, Kejie Qiu, Rui Huang, Chuan Fang, Le Cui, Zilong Dong, Siyu Zhu, and Ping Tan. Rendernet: Visual relocalization using virtual viewpoints in large-scale indoor environments. *arXiv preprint arXiv:2207.12579*, 2022. 2

- [51] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 8