Accelerated Coordinate Encoding: Learning to Relocalize in Minutes using RGB and Poses

Eric Brachmann
Niantic

Tommaso Cavallari
Niantic

Victor Adrian Prisacariu
Niantic, University of Oxford

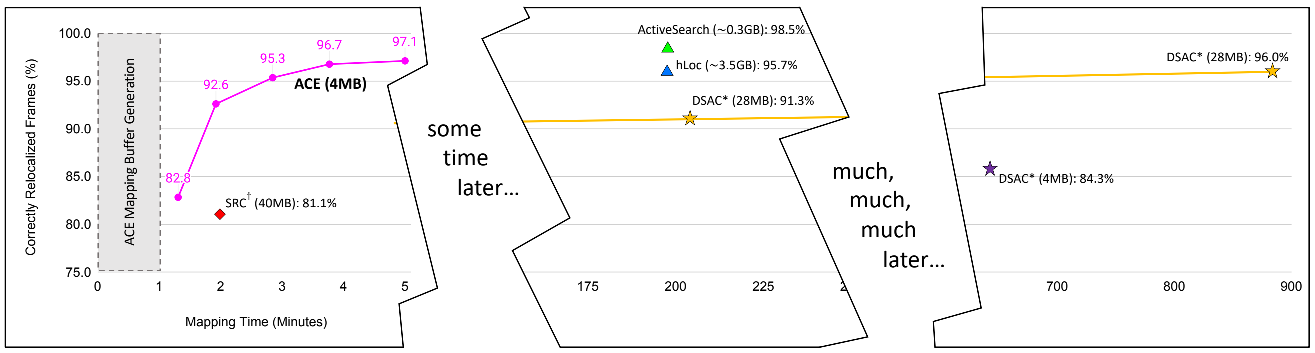


Figure 1. **Mapping Time vs. Relocalization Rate.** We show the mapping time of multiple state-of-the-art relocalizers on a standard dataset, 7Scenes [53]. We measure accuracy as the percentage of frames with a pose error below 5cm and 5°. Our approach, ACE, maps a new environment two orders of magnitude faster than the baseline, DSAC* [10], while being as accurate. We also report the average map size of each approach in brackets. † signifies that mapping needs depth, while other methods only need RGB and poses.

Abstract

Learning-based visual relocalizers exhibit leading pose accuracy, but require hours or days of training. Since training needs to happen on each new scene again, long training times make learning-based relocalization impractical for most applications, despite its promise of high accuracy. In this paper we show how such a system can actually achieve the same accuracy in less than 5 minutes. We start from the obvious: a relocalization network can be split in a scene-agnostic feature backbone, and a scene-specific prediction head. Less obvious: using an MLP prediction head allows us to optimize across thousands of view points simultaneously in each single training iteration. This leads to stable and extremely fast convergence. Furthermore, we substitute effective but slow end-to-end training using a robust pose solver with a curriculum over a reprojection loss. Our approach does not require privileged knowledge, such a depth maps or a 3D model, for speedy training. Overall, our approach is up to 300x faster in mapping than state-of-the-art scene coordinate regression, while keeping accuracy on par. Code is available: https://nianticlabs.github.io/ace

1. Introduction

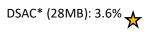
Time is really the only capital that any human being has, and the only thing he can't afford to lose.

Thomas Edison

Time is relative. Time spent waiting can stretch to infinity. Imagine waiting for a visual relocalizer to finally work in a new environment. It can take hours – and feel like days – until the relocalizer has finished its pre-processing of the scene. Only then can it estimate the camera's position and orientation to support real-time applications like navigation or augmented reality (AR).

Relocalizers need that extensive pre-processing to build a map of the environment that defines the coordinate space we want to relocalize in. Visual relocalizers typically build maps from sets of images of the environment, for each of which the camera pose is known. There are two prevalent families of structure-based relocalizers that meet the high accuracy requirements of applications like AR.

Sparse feature-matching approaches [12, 25, 40, 44, 48, 49, 67] need to build an explicit 3D reconstruction of a scene using structure-from-motion (SfM) software [51, 55, 63]. Even when poses of mapping images are known, the run-



time of SfM for scene triangulation varies a lot, and can lie anywhere between 10 minutes and 10 hours depending on how many mapping frames are used. When mapping succeeds, feature-based relocalizers are fast at query time and accurate [44,49]. Less refined maps can be built in real time using SLAM, if one is willing to accept the detrimental effect on accuracy [4]. In either case, the underlying maps can consume vast amounts of storage, and can reveal private information that was present in the mapping images [16,56].

On the other hand, scene coordinate regression [5, 7, 10, 20, 31, 53, 64] *learns* an implicit representation of the scene via gradient descent. The resulting maps can be as small as 4MB [10], and privacy preserving [67]. But, while scene coordinate regression is on-par with feature-matching in terms of accuracy and relocalization time [4], the fact that they map an environment via hours-long *training of a network* makes them unattractive for most applications. The state-of-the-art scene coordinate regression pipeline, DSAC* [10], requires 15 hours to reach top accuracy on a premium GPU, see Fig. 1. We can stop training any time, and see which accuracy we get but, after 5 minutes mapping time, DSAC* has a relocalization rate in the single digits. In fact, the corresponding data point for the plot in Fig. 1 can be found at the bottom of the previous page.

The aim of this work is summarized quickly: we take a scene coordinate regression-based relocalizer, the slowest approach in terms of mapping time, and make it one of the fastest. In particular, we present *Accelerated Coordinate Encoding* (ACE), a schema to train scene coordinate regression in 5 minutes to state-of-the-art accuracy.

Speeding up training time normally causes moderate interest in our community, at best. This is somewhat justified in train-once-deploy-often settings. Still, learning-based visual relocalization does not fall within that category, as training needs to happen on each new scene, again. Therefore, fast training has a range of important implications:

- **Mapping delay.** We reduce the time between collecting mapping data, and having a top-performing relocalizer for that environment.
- **Cost.** Computation time is expensive. Our approach maps a scene within minutes on a budget GPU.
- **Energy consumption.** Extensive computation is an environmental burden. We significantly reduce the resource footprint of learning-based relocalization.
- **Reproducibility.** Using ACE to map all scenes of the datasets used in this paper can be done almost five times over on a budget GPU, in the time it takes DSAC* to map a single scene on a premium GPU.

We show that a thoughtful split of a standard scene coordinate regression network allows for more efficient training. In particular, we regard scene coordinate regression as a mapping from a high-dimensional feature vector to a 3D point in scene space. We show that a multi-layer perceptron (MLP) can represent that mapping well, as opposed to convolutional networks normally deployed [7,10,20]. Training a scene-specific MLP allows us to optimize over many (oftentimes all available) mapping views at once in each single training iteration. This leads to very stable gradients that allow us to operate in very aggressive, high-learning rate regimes. We couple this with a curriculum over a reprojection loss that lets the network *burn in* on reliable scene structures at later stages of training. This mimics end-to-end training schemes that involve differentiating through robust pose estimation during training [10], but are much slower than our approach. We summarize our **contributions**:

- *Accelerated Coordinate Encoding* (ACE), a scene coordinate regression system that maps a new scene in 5 minutes. Previous state-of-the-art scene coordinate regression systems require hours of mapping to achieve comparable relocalization accuracy.
- ACE compiles a scene into 4MB worth of network weights. Previous scene coordinate regression systems required 7-times more storage, or had to sacrifice accuracy for scene compression.
- Our approach requires only posed RGB images for mapping. Previous fast mapping relocalizers relied on priviledged knowledge like depth maps or a scene mesh for speedy mapping.

2. Related Work

Visual relocalization requires some representation of the environment we want to relocalize in. We refer to these representations as “maps”, and the process of creating them as “mapping”. Our work is predominately concerned with the time needed for mapping, and, secondly, the storage demand of the maps created.

Image Retrieval and Pose Regression. Arguably the simplest form of a map is a database of mapping images and their poses. Given a query image, we look for the most similar mapping images using image retrieval [2, 42, 57], and approximate the query pose with the top retrieved mapping pose [12, 50]. Pose regression uses neural networks to either predict the absolute pose from a query image directly, or predict the relative pose between the query image and the top retrieved mapping image. All absolute and most relative pose regression methods [11, 28, 29, 52, 58, 62, 68] train scene-specific networks which can take significant time, *e.g.* [28] reports multiple hours per scene for PoseNet. Some relative pose regression works report results with generalist, scene-agnostic networks that do not incur additional mapping time on top of building the retrieval index [58,62]. Map-free relocalization [3] is an extreme variation that couples scene-agnostic relative pose regression with a single reference frame for practically instant relocalization. Recently, some authors use neural radiance fields (NeRFs) [38] for camera pose estimation [36, 66]. In its early stage, this family of methods has yet to demonstrate its merits against the corpus of existing relocalisers and on standard benchmarks. Some of the aforementioned approaches have attractive mapping times, *i.e.* require only little scene-specific pre-processing. But their pose accuracy falls far behind structure-based approaches that we discuss next.

Feature Matching. Feature matching-based relocalizers [12,25,40,44,49] calculate the camera pose from correspondences between the query image and 3D scene space. They establish correspondences via discrete matching of local feature descriptors. Thus, they require a 3D point cloud of an environment where each 3D point stores one or multiple feature descriptors for matching. These point clouds can be created by running SfM software, such as COLMAP [51]. Even if poses of mapping images are known in advance, *e.g.* from on-device visual odometry [1, 24, 27, 39], feature triangulation with SfM can take several hours, depending on the number of mapping frames. Also, the storage requirements can be significant, mainly due to the need for storing hundreds of thousands of descriptor vectors for matching. Strategies exist to alleviate the storage burden, such as storing fewer descriptors per 3D point [26,47,49], compressing descriptors [35, 65] or removing 3D points [65]. More recently, GoMatch [67] and MeshLoc [40] removed the need to store descriptors entirely by matching against the scene geometry. None of the aforementioned strategies reduce the mapping time – on the converse, often they incur additional post-processing costs for the SfM point clouds. To reduce mapping time, one could use only a fraction of all mapping images for SfM, or reduce the image resolution. However, this would likely also affect the pose estimation accuracy.

Scene Coordinate Regression. Relocalizers in this family regress 3D coordinates in scene space for a given 2D pixel position in the query image [53]. Robust optimization over scene-to-image correspondences yields the desired query camera pose. To regress correspondences, most works rely on random forests [6, 14, 15, 53, 61] or, more recently, convolutional neural networks [5, 7, 8, 10, 13, 20, 31]. Thus, the scene representation is implicit, and the map is encoded in the weights of the neural network. This has advantages as the implicit map is privacy-preserving [56, 67]: an explicit scene representation can only be re-generated with images of the environment. Also, scene coordinate regression has small storage requirements. DSAC* [10] achieves state-of-the-art accuracy with 28MB networks, and acceptable accuracy with 4MB networks. Relocalization in large-scale environments can be challenging, but strategies exist that rely on network ensembles [8].

The main drawback of scene coordinate regression is its long mapping time, since mapping entails training a neural network for each specific scene. DSAC++ [7] reported 6 days of training for a single scene. DSAC* reduced the training time to 15 hours – given a powerful GPU. This is still one order of magnitude slower than typical feature matching approaches need to reconstruct a scene. In our work, we show how few conceptual changes to a scene coordinate regression pipeline result in a speedup of two orders of magnitude. Thus, we pave the way for deep scene coordinate regression to be useful in practical applications.

A variety of recipes allow for fast mapping if depth, rendered or measured, is given. Indeed, the original SCoRF paper [53] reported to train their random forest with RGB-D images under 10 minutes. Cavallari *et al.* [15] show how to adapt a pre-trained neural scene representation in real time for a new scene, but their approach requires depth inputs for the adaptation, and for relocalization. Dong *et al.* [20] use very few mapping frames with depth to achieve a mapping time of 2 minutes. The architecture described in [20] consists of a scene-agnostic feature backbone, and a scene-specific region classification head – very similar to our setup. However, their prediction head is convolutional, and thus misses the opportunity for highly efficient training as we will show. SANet [64] is a scene coordinate regression variant that builds on image retrieval. A scene-agnostic network interpolates the coordinate maps of the top retrieved mapping frames to yield the query scene coordinates. None of the aforementioned approaches is applicable when mapping images are RGB only. Depth channels for mapping can be rendered from a dense scene mesh [7], but mesh creation would increase the mapping time. Our work is the first to show fast scene coordinate regression mapping from RGB and poses alone.

3. Method

Our goal is to estimate a camera pose $\mathbf{h}$ given a single RGB image I.We define the camera pose as the rigid body transformation that maps coordinates in camera space $\mathbf{e}_i$ to coordinates in scene space $\mathbf{y}_i$, therefore $\mathbf{y}_i = \mathbf{h}\mathbf{e}_i$. We can estimate the pose from image-to-scene correspondences:

$$\mathbf{h} = g(\mathcal{C}), \text{ with } \mathcal{C} = \{(\mathbf{x}_i, \mathbf{y}_i)\}, \tag{1}$$

where $\mathcal{C}$ is the set of correspondences between 2D pixel positions $\mathbf{x}_i$ and 3D scene coordinates $\mathbf{y}_i$. Function g denotes a robust pose solver. Usually g consists of a PnP minimal solver [22] in a RANSAC [21] loop, followed by refinement. Refinement consists of iterative optimization of the reprojection error over all RANSAC inliers using Levenberg–Marquardt [30, 37]. For more details concerning pose solving we refer to [10], as our focus is on correspondence prediction. To obtain correspondences, we follow the ap-

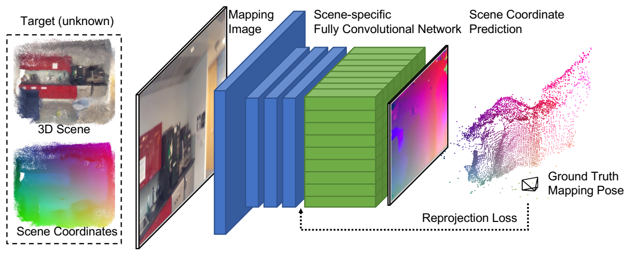

Target (unknown)
3D Scene
Scene Coordinates
Mapping Image
Scene-specific Fully Convolutional Network
Scene Coordinate Prediction
Ground Truth Mapping Pose
Reprojection Loss

Figure 2. **Standard Training Loop [10].** Previous works train a coordinate regression network with one mapping image at a time. The network predicts dense scene coordinates, and is supervised with the ground truth camera pose and a reprojection loss.

proach of scene coordinate regression [53]. We learn a function to predict 3D scene points for any 2D image location:

$$\mathbf{y}_i = f(\mathbf{p}_i; \mathbf{w}), \text{ with } \mathbf{p}_i = \mathcal{P}(\mathbf{x}_i, I), \tag{2}$$

where f is a neural network parameterized by learnable weights $\mathbf{w}$, and $\mathbf{p}_i$ is an image patch extracted around pixel position $\mathbf{x}_i$ from image I. Therefore, f implements a mapping from patches to coordinates, $f : \mathbb{R}^{C_\text{I} \times H_\text{P} \times W_\text{P}} \to \mathbb{R}^3$. We have RGB images but usually take grayscale inputs with $C_\text{I} = 1$. Typical patch dimensions are $H_\text{P} = W_\text{P} = 81$px [7–10]. For state-of-the-art architectures there is no explicit patch extraction. A fully convolutional neural network [33] with limited receptive field slides over the input image to efficiently predict dense outputs while reusing computation of neighbouring pixels. However, for our subsequent discussion, the explicit patch notation will prove useful.

We learn the function f by optimizing over all mapping images $\mathcal{I}_\text{M}$ with their ground truth poses $\mathbf{h}_i^*$ as supervision:

$$\underset{\mathbf{w}}{\text{argmin}} \sum_{I \in \mathcal{I}_\text{M}} \sum_i \ell_{\boldsymbol{\pi}}[\mathbf{x}_i, \overbrace{f(\mathbf{p}_i; \mathbf{w})}^{\mathbf{y}_i}, \mathbf{h}_i^*], \tag{3}$$

where $\ell_{\boldsymbol{\pi}}$ is a reprojection loss that we discuss in Sec. 3.2. We optimize Eq. 3 using minibatch stochastic gradient descent. The network predicts dense scene coordinates from one mapping image at a time, and all predictions are supervised using the ground truth mapping pose, see Fig. 2.

3.1. Efficient Training by Gradient Decorrelation

With the standard training, we optimize over predictions for thousands of patches in each training iteration – but they all come from the same image. Hence, their loss and their gradients will be highly correlated. A prediction $\mathbf{y}_i$ and the prediction for the pixel next to it will be very similar, so will be the pixel loss and its gradient.

Our key idea is to randomize patches over the entire training set, and construct training batches from many different mapping views. This decorrelates gradients within a batch and leads to a very stable training signal, robustness to high learning rates, and, ultimately, fast convergence.

A naive implementation of this idea would be slow if it resorted to explicit patch extraction [5]. The expressive power of convolutional layers, and their efficient computation using fully convolutional architectures is key for state-of-the-art scene coordinate regression. Therefore, we propose to split the regression network into a convolutional backbone, and a multi-layer perceptron (MLP) head:

$$f(\mathbf{p}_i; \mathbf{w}) = f_\text{H}(\mathbf{f}_i; \mathbf{w}_\text{H}), \text{ with } \mathbf{f}_i = f_\text{B}(\mathbf{p}_i; \mathbf{w}_\text{B}), \tag{4}$$

where f_B is the backbone that predicts a high-dimensional feature $\mathbf{f}_i$ with dimensionality $C_\mathbf{f}$, and f_H is the regression head that predicts scene coordinates:

$$f_\text{B} : \mathbb{R}^{C_\text{I} \times H_\text{P} \times W_\text{P}} \to \mathbb{R}^{C_\mathbf{f}} \text{ and } f_\text{H} : \mathbb{R}^{C_\mathbf{f}} \to \mathbb{R}^3. \tag{5}$$

Similar to [20], we argue that f_B can be implemented using a scene-agnostic convolutional network - a generic feature extractor. In addition to [20], we argue that f_H can be implemented using a MLP instead of another convolutional network. Fig. 2 signifies our network split. Convolution layers with 3×3 kernels are blue, and 1×1 convolutions are green. The latter are MLPs with shared weights. This standard network design is used in pipelines like DSAC* [10].

Note how function f_H needs no spatial context, *i.e.* differently from the backbone, f_H does not need access to neighbouring pixels for its computation. Therefore, we can easily construct training batches for f_H with random samples across all mapping images. Specifically, we construct a fixed size training buffer by running the pre-trained backbone f_B over the mapping images. This buffer contains millions of features $\mathbf{f}_i$ with their associated pixels positions $\mathbf{x}_i$, camera intrinsics $\mathbf{K}_i$ and ground truth mapping poses $\mathbf{h}_i^*$. We generate this buffer once, in the first minute of training. Afterwards, we start the main training loop that iterates over the buffer. At the beginning of each epoch, we shuffle the buffer to mix features (essentially patches) across all mapping data. In each training step, we construct batches of several thousand features, potentially computing a parameter update over thousands of mapping views at once. Not only is the gradient computation extremely efficient for our MLP regression head, but the gradients are also decorrelated which allows us to use high learning rates for fast convergence. Fig. 3 shows our training procedure.

3.2. Curriculum Training

Previous state-of-the-art scene coordinate regression pipelines use a multi-stage training process. Firstly, they optimize a pixel-level reprojection loss. Secondly, they do end-to-end training, where they propagate a pose error back through a differentiable pose solver [5, 7]. End-to-end training lets the network focus on reliable scene structures while ignoring outlier predictions. However, end-to-end training is extremely costly. For example, in [10], end-to-end training incurs half of the training time for 10% of the parameter

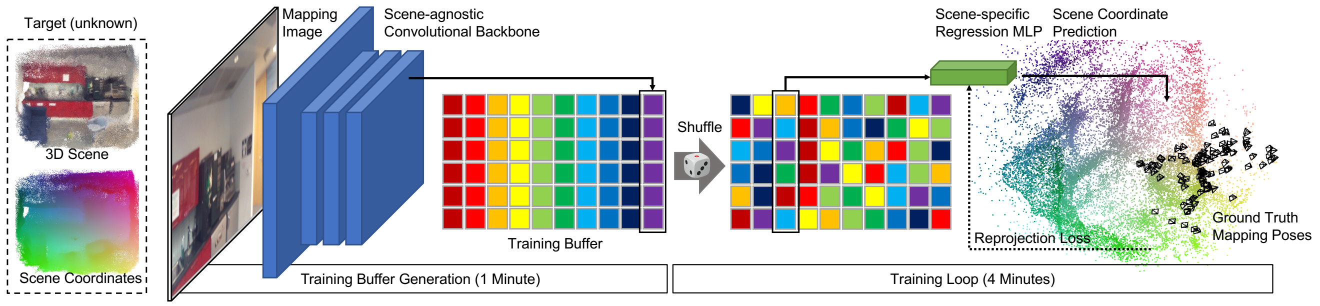

Target (unknown)
3D Scene
Scene Coordinates
Mapping Image
Scene-agnostic Convolutional Backbone
Training Buffer
Training Buffer Generation (1 Minute)
Shuffle
Scene-specific Regression MLP
Scene Coordinate Prediction
Reprojection Loss
Ground Truth Mapping Poses
Training Loop (4 Minutes)

Figure 3. **ACE Training Loop.** Training consists of two stages: Buffer generation (left) and the main training loop (right). To create a training buffer, we pass mapping images through a scene-agnostic backbone that extracts high-dimensional feature vectors. Each colored box in the buffer represents one such feature, and features with the same color came from the same mapping image. In the main loop, we train a scene-specific MLP that predicts scene coordinates from backbone features. We assemble training batches from random features and their associated mapping poses. Thus, we supervise the scene-specific MLP with many, diverse mapping views in each training iteration.

updates. To mimic the effects of end-to-end training, we construct a curriculum over a much simpler pixel-wise reprojection loss. We use a moving inlier threshold throughout the training process that starts loose, and gets more restrictive as training progresses. Therefore, the network can focus on predictions that are already good, and neglect less precise predictions that would be filtered by RANSAC during pose estimation. Our training loss is based on the pixel-wise reprojection loss of DSAC* [10]:

$$\ell_{\boldsymbol{\pi}}[\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*] = \begin{cases} \hat{e}_{\boldsymbol{\pi}}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) & \text{if } \mathbf{y}_i \in \mathcal{V} \\ ||\mathbf{y}_i - \bar{\mathbf{y}}_i||_0 & \text{otherwise.} \end{cases} \quad (6)$$

This loss optimizes a robust reprojection error $\hat{e}_{\boldsymbol{\pi}}$ for all *valid* coordinate predictions $\mathcal{V}$. Valid predictions are between 10cm and 1000m in front of the image plane, and have a reprojection error below 1000px. For invalid predictions, the loss optimizes the distance to a dummy scene coordinate $\bar{\mathbf{y}}_i$ that is calculated from the ground truth camera pose assuming a fixed image depth of 10m. The main difference between DSAC* and our approach is in the definition of the robust reprojection error $\hat{e}_{\boldsymbol{\pi}}$. DSAC* uses the reprojection error $e_{\boldsymbol{\pi}}$ up to a threshold τ, and the square root of the reprojection error beyond. Instead, we use $\tanh$ clamping of the reprojection error:

$$\hat{e}_{\boldsymbol{\pi}}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*) = \tau(t)\ \tanh\left(\frac{e_{\boldsymbol{\pi}}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}_i^*)}{\tau(t)}\right) \quad (7)$$

We dynamically re-scale the $\tanh$ according to a threshold τ that varies throughout training:

$$\tau(t) = w(t)\,\tau_{\text{max}} + \tau_{\text{min}}, \text{ with } w(t) = \sqrt{1 - t^2}, \quad (8)$$

where $t \in (0, 1)$ denotes the relative training progress. This curriculum implements a circular schedule of threshold τ, which remains close to τ_{max} in the beginning of training, and declines towards τ_{min} at the end of training.

3.3. Backbone Training

As backbone, we can use any dense feature description network [19, 32, 43, 59]. However, existing solutions are often optimized towards sparse feature matching. Their descriptors are meant to be informative at key points. In contrast, we need descriptors that are distinctive for any position in the input image. Thus, we present a simple way to train a feature description network tailored towards scene coordinate regression. We adhere to the network architecture of DSAC* [10]. We use the early convolutional layers as our backbone, and split off the subsequent MLP as our scene-specific regression head. To train the backbone, we resort to the image-level training of DSAC* [10] (*cf*. Fig. 2) but couple it with our training curriculum of Eq. 6.

Instead of training the backbone with one regression head for a single scene, we train it with N regression heads for N scenes, in parallel. This bottleneck architecture forces the backbone to predict features that are useful for a wide range of scenes. We train the backbone on 100 scenes from ScanNet [17] for 1 week, resulting in 11MB of weights that can be used to extract dense descriptors on any new scene. See the Supplement for more details on the training process.

3.4. Further Improvements

We train the entire network with half-precision floating-point weights. This gives us an additional speed boost, especially on budget GPUs. We also store our networks with float16 precision. This allows us to increase the depth of our regression heads while maintaining 4MB maps. On top of our loss curriculum (see Sec. 3.2), we use a one cycle learning rate schedule [54], *i.e.* we increase the learning rate in the middle of training, and reduce it towards the end. We found a small but consistent advantage in overparameterizing the scene coordinate representation: we predict homogeneous coordinates $\mathbf{y}' = (x, y, z, w)^\top$ and apply a w-clip, enforcing w to be positive by applying a softplus operation.

		Mapping w/ Mesh/Depth	Mapping Time	Map Size	7 Scenes		12 Scenes	
					SfM poses	D-SLAM poses	SfM poses	D-SLAM poses
FM	AS (SIFT) [49]	No	∼1.5h	∼200MB	98.5%	68.7%	99.8%	99.6%
	D.VLAD+R2D2 [25]	No		∼1GB	95.7%	77.6%	99.9%	99.7%
	hLoc (SP+SG) [44, 45]	No		∼2GB	95.7%	76.8%	100%	99.8%
	pixLoc [46]	No		∼1GB	N/A	75.7%	N/A	N/A
SCR (w/ Depth)	DSAC* (Full) [10]	Yes	15h	28MB	98.2%	84.0%	99.8%	99.2%
	DSAC* (Tiny) [10]	Yes	11h	4MB	85.6%	70.0%	84.4%	83.1%
	SANet [64]	Yes	∼2.3 min	∼550MB	N/A	68.2%	N/A	N/A
	SRC [20]	Yes	2 min†	40MB	81.1%	55.2%	N/A	N/A
SCR	DSAC* (Full) [10]	No	15h	28MB	<u>96.0%</u>	**81.1%**	<u>99.6%</u>	<u>98.8%</u>
	DSAC* (Tiny) [10]	No	11h	4MB	84.3%	69.1%	81.9%	81.6%
	ACE (ours)	No	5 min	4MB	**97.1%**	<u>80.8%</u>	**99.9%**	**99.6%**

Table 1. **Indoor Relocalization Results.** We report the percentage of frames below a 5cm,5° pose error. Best results in **bold** for the "SCR" group, second best results <u>underlined</u>. We list the time needed for mapping, the map size and, whether depth (rendered or measured) is needed for mapping. See the main text and Supp. for details on these numbers. † does not include time needed to pre-cluster the scene.

4. Experiments

We implement our approach in PyTorch [41], based on the public code of DSAC* [10]. We list our main parameter choices here, and refer to the Supplement for more details. We create a training buffer of 8M backbone features in 1 minute, randomly sampled from mapping images. In the 4 minutes training, we do 16 complete passes over the training buffer with a batch size of 5120. Our training curriculum starts with a soft threshold of $\tau_{\text{max}} = 50$px and ends with $\tau_{\text{min}} = 1$px. We optimize using AdamW [34] with a learning rate between $5 \cdot 10^{-4}$ and $5 \cdot 10^{-3}$ and a 1 cycle schedule [54]. We reuse the DSAC* robust pose estimator with 64 RANSAC hypotheses and 10px inlier threshold.

4.1. Indoor Relocalization

We conducted experiments on 7Scenes [53] and 12Scenes [60], two indoor relocalization datasets. They provide mapping and query images for several small-scale indoor rooms. In these environments, speedy mapping is particularly desirable. If mapping takes hours or days, the room might have changed, or the user might have wandered off. Two sets of ground truth poses are available [4], one from running depth-based SLAM [18, 27], one from running SfM [51]. We report results on both because relocalizer performance can be biased towards one or the other [4].

Our main comparison is to DSAC* [10], our baseline. We include other top-performing methods as well, such as feature matching (FM) approaches, and scene coordinate regression (SCR) pipelines that use depth during mapping. We list results in Table 1. Multiple approaches achieve high accuracy, but require vastly different resources. ACE is the only approach that 1) achieves top accuracy 2) in less than 10 minutes per scene 3) with less than 10MB per scene. At the same time, it does not need depth – either measured or rendered from a mesh – for mapping.

Note: Comparing mapping times and map sizes is a difficult endevour. The exact numbers will depend on hardware, implementation details and hyper-parameters. For our main baseline, DSAC*, we made sure that the numbers are accurate by re-running their public code on our hardware. For the other approaches, which we mainly list for context, we resort to numbers given in their respective papers, or estimated numbers based on publicly available statistics. We detail our reasoning for each method in the Supplement. As such, these numbers should be taken with a grain of salt. But we are confident that the order of magnitude is correct, as confirmed by cross-referencing with other publications.

4.2. Outdoor Relocalization

Cambridge Landmarks [29]. This dataset collects mapping and query images of buildings across the old town of Cambridge. Ground truth poses stem from reconstructing mapping and query images jointly using SfM [63]. We report our results in Table 2. Feature matching-approaches do very well on this dataset, presumably due to their similarity to the SfM reference algorithm [4]. ACE performs reasonably well compared to DSAC* considering the spatial extent of each scene and our 4MB memory footprint. We provide a variant of ACE, denoted *Poker*, where we split a scene into four ACE models, and return the pose estimate with the largest inlier count at query time (see Supp. for details). *Poker* outperforms DSAC* on average while still being comparatively lean w.r.t. mapping time and storage.

Wayspots. The previous datasets are not ideal for fully showcasing the advantages of our method. Their mapping poses either stem from D-SLAM, where depth would be available, or SfM reconstruction, where 3D point clouds would be available. However, RGB-based visual odometry runs on millions of phones [1, 24] providing mapping images and poses, but neither depth nor full 3D point clouds

		Mapping w/ Mesh/Depth	Mapping Time	Map Size	Cambridge Landmarks					Average (cm / °)
					Court	King's	Hospital	Shop	St. Mary's	
FM	AS (SIFT) [49]	No		$\sim$200MB	24/0.1	13/0.2	20/0.4	4/0.2	8/0.3	14/0.2
	hLoc (SP+SG) [44, 45]	No		$\sim$800MB	16/0.1	12/0.2	15/0.3	4/0.2	7/0.2	11/0.2
	pixLoc [46]	No	$\sim$35min	$\sim$600MB	30/0.1	14/0.2	16/0.3	5/0.2	10/0.3	15/0.2
	GoMatch [67]	No		$\sim$12MB	N/A	25/0.6	283/8.1	48/4.8	335/9.9	N/A
	HybridSC [12]	No		$\sim$1MB	N/A	81/0.6	75/1.0	19/0.5	50/0.5	N/A
APR	PoseNet17 [28]	No	4 – 24h	50MB	683/3.5	88/1.0	320/3.3	88/3.8	157/3.3	267/3.0
	MS-Transformer [52]	No	$\sim$7h	$\sim$18MB	N/A	83/1.5	181/2.4	86/3.1	162/4.0	N/A
SCR w/ Depth	DSAC* (Full) [10]	Yes	15h	28MB	49/0.3	15/0.3	21/0.4	5/0.3	13/0.4	21/0.3
	SANet [64]	Yes	$\sim$1min	$\sim$260MB	328/2.0	32/0.5	32/0.5	10/0.5	16/0.6	84/0.8
	SRC [20]	Yes	2 min†	40MB	81/0.5	39/0.7	38/0.5	19/1.0	31/1.0	42/0.7
SCR	DSAC* (Full) [10]	No	15h	28MB	34/0.2	**18/0.3**	**21/0.4**	**5/0.3**	15/0.6	19/0.4
	DSAC* (Tiny) [10]	No	11h	4MB	98/0.5	27/0.4	33/0.6	11/0.5	56/1.8	45/0.8
	ACE (ours)	No	5 min	4MB	43/0.2	28/0.4	31/0.6	**5/0.3**	18/0.6	25/0.4
	Poker (Quad ACE Ensemble)	No	20 min	16MB	**28/0.1**	**18/0.3**	25/0.5	**5/0.3**	**9/0.3**	**17/0.3**

Table 2. **Cambridge Landmarks [29] Results.** We report median rotation and position errors. Best results in **bold** for the "SCR" group, second best results underlined. Methods using depth for mapping rely on a dense multi-view stereo mesh that took hours to compute [7, 29]. See the main text and Supp. for details about mapping times and storage demands. † does not include time needed to pre-cluster the scene.

with feature descriptors. Therefore, we curate a new relocalization dataset, denoted *Wayspots*, from a publicly available corpus of phone scans. In particular, we select 10 consecutive scenes from the training split of the MapFree dataset [3]. Each scene depicts a small outdoor place and comes with two full, independent scans. We use one for mapping and one for query. Arnold *et al.* [3] reconstructed ground truth poses of both scans using SfM [51]. We register the original phone trajectories to the SfM poses. In our experiments, we used the original phone trajectories for mapping, and SfM poses solely for evaluation. We refer to the Supplement for details on the dataset and its curation. We show an overview of all scenes, and our main results in Fig. 4. We outperform DSAC* on average while being two orders of magnitude faster in mapping. In Fig. 5, we show a qualitative comparison of ACE and DSAC* on the "Rock" scene, including a variant of DSAC* that was stopped after 5 minutes mapping.

4.3. Analysis

Mapping Time. The speed of training a network depends on the GPU model. In Table 3, we compare mapping times of ACE and DSAC* on a premium GPU (NVIDIA V100) with a more affordable model (NVIDIA T4)[1]. For DSAC*, mapping takes twice as long on the cheaper GPU. The difference is smaller for the "Tiny" version of DSAC* that uses a leaner network, but mapping still takes hours. Conversely, ACE experiences only a 10% slowdown on the T4 GPU, due to having to train just the regression head, in half-precision.

[1]At time of writing, instantiating a virtual machine with a Tesla T4 on a popular cloud computing provider [23] costs USD 0.35/h, whereas the more powerful Tesla V100 GPU costs USD 2.48/h, 7x more expensive.

GPU	Method	Mapping Time	ACE Speed-up
NVIDIA V100	ACE	291s	-
	DSAC* (Tiny)	11h	130x
	DSAC* (Full)	15h	180x
NVIDIA T4	ACE	327s	-
	DSAC* (Tiny)	14h	150x
	DSAC* (Full)	28h	310x

Table 3. **Mapping Times.** Comparison of methods on different GPUs. T4 GPUs offer less compute capabilities but are considerably cheaper. DSAC* (Full) uses a 28MB network. DSAC* (Tiny) uses a smaller, 4MB network.

Map Size. The network architecture of our regression head determines the size of our maps. In particular we can vary the number of layers in our MLP, see Fig. 6 for an analysis. Smaller maps of 2.5MB achieve slightly lower accuracy, but still yield comfortable relocalization rates above 95% on 7Scenes. Using larger maps of 5.5MB does not pay off: because of their increased size, these networks undergo fewer passes over the training buffer in 5 minutes. Note how ACE achieves $\sim$80% accuracy after a single epoch (75s).

More Experiments in the Supplement. We demonstrate the positive impact of gradient de-correlation, predicting homogeneous scene coordinates and our loss curriculum. Furthermore, we substitute the ACE feature backbone with various off-the-shelf networks, including SuperPoint [19] and DISK [59], to find them less suited for our task. We also vary the dimensionality of ACE backbone features to find more dimensions working better. See supplement for details.

Scene	DSAC* (Full)	DSAC* (Tiny)	ACE (Ours)
Cubes	83.8%	68.7%	**97.0%**
Bears	**82.6%**	73.1%	80.7%
Winter Sign	0.2%	0.3%	**1.0%**
Inscription	**54.1%**	41.3%	49.0%
The Rock	**100%**	99.8%	**100%**
Tendrils	25.1%	19.6%	**34.9%**
Map	**56.7%**	53.3%	56.5%
Square Bench	**69.5%**	60.3%	66.7%
Statue	0.0%	0.0%	0.0%
Lawn	34.7%	20.0%	**35.8%**
Average	50.7%	43.6%	**52.2%**

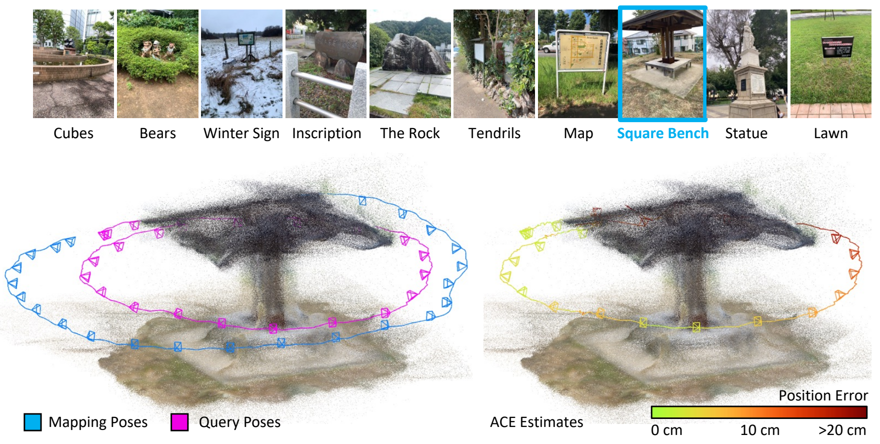


Figure 4. **The Wayspots Dataset.** We curate a new relocalization dataset from a public corpus of phone scans [3]. (left) We show accuracy as the percentage of frames below 10cm,5° pose error. Best results in **bold**. (right) One mapping image of each scene, and a visualisation of ground truth poses and ACE estimates for one scene. The dataset does not provide 3D point clouds, so we show the ACE map instead. See the Supplementary for details of the process we use to render 3D point clouds from the trained ACE maps.

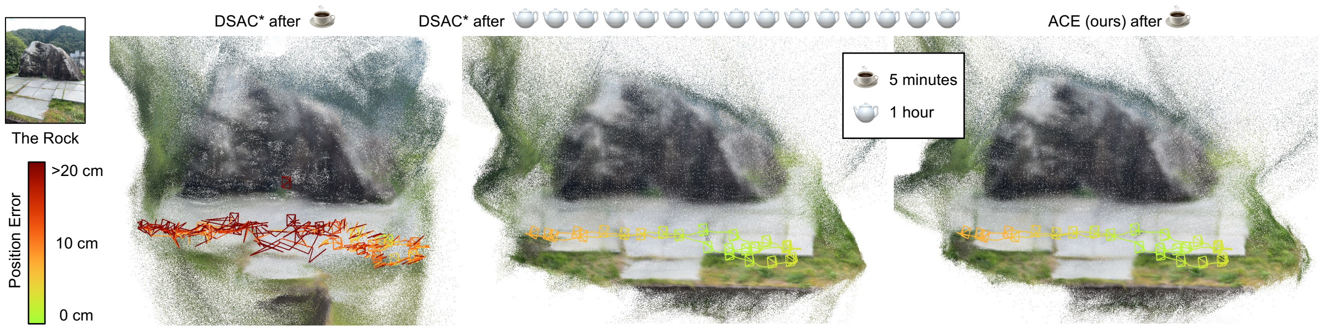


Figure 5. **Qualitative results.** We compare qualitative results of ACE and DSAC* on one scene of the Wayspots dataset. Full training of DSAC* takes 15 hours. When stopped after 5 minutes, relocalization accuracy is poor. We show the learned map for each method.

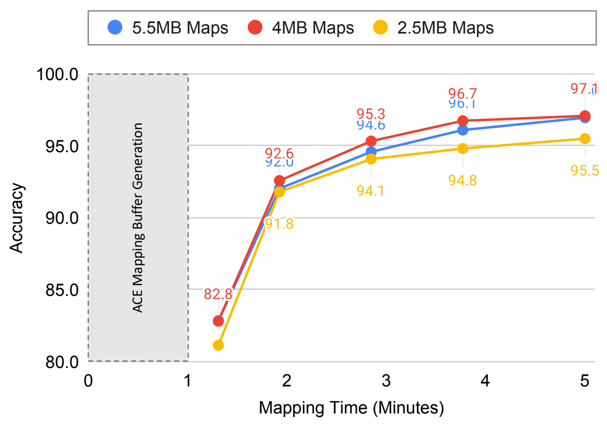


Figure 6. **Map Size.** We vary the map size of ACE by using more or less MLP layers. 4MB maps correspond to our standard settings. Results on 7Scenes with SfM pseudo ground truth poses.

5. Conclusion and Future Work

We have presented ACE, a relocalizer able to map new environments in 5 minutes. ACE reduces cost and energy consumption of mapping by two orders of magnitude compared to previous RGB-based scene coordinate regression approaches, making this family of algorithms practical.

The changes from previous state-of-the-art relocalizers that we propose in this paper are mainly conceptual, leveraging decorrelation of gradients by patch-level training. We see further potential for speedups by clever engineering, such as interleaving buffer creation and training in separate threads, or early stopping for easy scenes.

Acknowledgements: We thank Torsten Sattler for helpful comments about the resource footprint of feature-based relocalizers, Mohamed Sayed for help with the 3D visualisation of ACE, and Clément Godard for early contributions to the code base of ACE.

References

[1] Apple. ARKit. Accessed: 11 November 2022. 3, 6

[2] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *CVPR*, 2016. 2

[3] Eduardo Arnold, Jamie Wynn, Sara Vicente, Guillermo Garcia-Hernando, Áron Monszpart, Victor Adrian Prisacariu, Daniyar Turmukhambetov, and Eric Brachmann. Map-free visual relocalization: Metric pose relative to a single image. In *ECCV*, 2022. 2, 7, 8

[4] Eric Brachmann, Martin Humenberger, Carsten Rother, and Torsten Sattler. On the limits of pseudo ground truth in visual camera re-localisation. In *ICCV*, 2021. 2, 6

[5] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-differentiable ransac for camera localization. In *CVPR*, 2017. 2, 3, 4

[6] Eric Brachmann, Frank Michel, Alexander Krull, Michael Y. Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016. 3

[7] Eric Brachmann and Carsten Rother. Learning Less is More - 6D Camera Localization via 3D Surface Regression. In *CVPR*, 2018. 2, 3, 4, 7

[8] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *ICCV*, 2019. 3, 4

[9] Eric Brachmann and Carsten Rother. Neural-guided RANSAC: Learning where to sample model hypotheses. In *ICCV*, 2019. 4

[10] Eric Brachmann and Carsten Rother. Visual camera re-localization from RGB and RGB-D images using DSAC. *TPAMI*, 2021. 1, 2, 3, 4, 5, 6, 7

[11] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *CVPR*, 2018. 2

[12] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid scene compression for visual localization. In *CVPR*, 2019. 1, 2, 3, 7

[13] Tommaso Cavallari, Luca Bertinetto, Jishnu Mukhoti, Philip Torr, and Stuart Golodetz. Let's take this online: Adapting scene coordinate regression network predictions for online rgb-d camera relocalisation. In *3DV*, 2019. 3

[14] 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 re-localisation. In *CVPR*, 2017. 3

[15] Tommaso Cavallari, Stuart Golodetz, Nicholas A. Lord, Julien Valentin, Victor A. Prisacariu, Luigi Di Stefano, and Philip H. S. Torr. Real-time rgb-d camera pose estimation in novel scenes using a relocalisation cascade. *TPAMI*, 2019. 3

[16] Kunal Chelani, Fredrik Kahl, and Torsten Sattler. How privacy-preserving are line clouds? recovering scene details from 3d lines. In *CVPR*, 2021. 2

[17] 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. 5

[18] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *TOG*, 2017. 6

[19] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 5, 7

[20] Siyan Dong, Shuzhe Wang, Yixin Zhuang, Juho Kannala, Marc Pollefeys, and Baoquan Chen. Visual localization via few-shot scene region classification. In *3DV*, 2022. 2, 3, 4, 6, 7

[21] 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. 3

[22] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *TPAMI*, 2003. 3

[23] Google. Google Compute Engine GPU Pricing. Accessed: 11 November 2022. 7

[24] Google. ARCore. Accessed: 11 November 2022. 3, 6

[25] 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, 2020. 1, 3, 6

[26] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009. 3

[27] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera. In *UIST*, 2011. 3, 6

[28] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. *CVPR*, 2017. 2, 7

[29] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-DOF camera relocalization. In *CVPR*, 2015. 2, 6, 7

[30] K. Levenberg. A method for the solution of certain problems in least squares. *Quaterly Journal on Applied Mathematics*, 1944. 3

[31] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *CVPR*, 2020. 2, 3

[32] Ce Liu, Jenny Yuen, and Antonio Torralba. SIFT flow: Dense correspondence across scenes and its applications. *TPAMI*, 2011. 5

[33] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 4

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6

[35] 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. *Intl. Journal of Robotics Research*, 2020. 3

[36] Dominic Maggio, Marcus Abate, Jingnan Shi, Courtney Mario, and Luca Carlone. Loc-NeRF: Monte carlo localization using neural radiance fields, 2022. 3

[37] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 1963. 3

[38] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3

[39] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 3

[40] Vojtech Panek, Zuzana Kukelova, and Torsten Sattler. MeshLoc: Mesh-Based Visual Localization. In *ECCV*, 2022. 1, 3

[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017. 6

[42] Jerome Revaud, Jon Almazan, Rafael Rezende, and Cesar De Souza. Learning with average precision: Training image retrieval with a listwise loss. In *ICCV*, 2019. 2

[43] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019. 5

[44] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 1, 2, 3, 6, 7

[45] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 6, 7

[46] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Victor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the Feature: Learning Robust Camera Localization from Pixels to Pose. In *CVPR*, 2021. 6, 7

[47] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *ICCV*, 2011. 3

[48] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012. 1

[49] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization. *TPAMI*, 2017. 1, 2, 3, 6, 7

[50] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura 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. 1, 3, 6, 7

[52] Yoli Shavit, Ron Ferens, and Yosi Keller. Learning multi-scene absolute pose regression with transformers. In *ICCV*, 2021. 2, 7

[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, 3, 4, 6

[54] Leslie N. Smith and Nicholay Topin. Super-convergence: very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, 2019. 5, 6

[55] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH*, 2006. 1

[56] Pablo Speciale, Johannes L. Schonberger, Sing Bing Kang, Sudipta N. Sinha, and Marc Pollefeys. Privacy preserving image-based localization. In *CVPR*, June 2019. 2, 3

[57] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *CVPR*, 2015. 2

[58] 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

[59] MichałTyszkiewicz, Pascal Fua, and Eduard Trulls. DISK: Learning local features with policy gradient. In *NeurIPS*, 2020. 5, 7

[60] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *3DV*, 2016. 6

[61] 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. 3

[62] Dominik Winkelbauer, Maximilian Denninger, and Rudolph Triebel. Learning to localize in new environments from synthetic training data. In *ICRA*, 2021. 2

[63] Changchang Wu. VisualSFM: A visual structure from motion system, 2011. 1, 6

[64] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. SANet: Scene agnostic network for camera localization. In *ICCV*, 2019. 2, 3, 6, 7

[65] Luwei Yang, Rakesh Shrestha, Wenbo Li, Shuaicheng Liu, Guofeng Zhang, Zhaopeng Cui, and Ping Tan. Scenesqueezer: Learning to compress scene for camera relocalization. In *CVPR*, 2022. 3

[66] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. iNeRF: Inverting neural radiance fields for pose estimation. In *IROS*, 2021. 3

[67] Qunjie Zhou, Sérgio Agostinho, Aljoša Ošep, and Laura Leal-Taixé. Is geometry enough for matching in visual localization? In *ECCV*, 2022. 1, 2, 3, 7

[68] Qunjie Zhou, Torsten Sattler, Marc Pollefeys, and Laura Leal-Taixe. To learn or not to learn: Visual localization from essential matrices. In *ICRA*, 2020. 2