SANet: Scene Agnostic Network for Camera Localization

Luwei Yang[1,*] Ziqian Bai[1,*] Chengzhou Tang[1] Honghua Li[2] Yasutaka Furukawa[1] Ping Tan[1]
[1] Simon Fraser University [2] Alibaba A.I Labs
{luweiy, ziqianb, cta73, furukawa, pingtan}@sfu.ca, honghua.lhh@alibaba-inc.com

Abstract

This paper presents a scene agnostic neural architecture for camera localization, where model parameters and scenes are independent from each other. Despite recent advancement in learning based methods, most approaches require training for each scene one by one, not applicable for online applications such as SLAM and robotic navigation, where a model must be built on-the-fly. Our approach learns to build a hierarchical scene representation and predicts a dense scene coordinate map of a query RGB image on-the-fly given an arbitrary scene. The 6D camera pose of the query image can be estimated with the predicted scene coordinate map. Additionally, the dense prediction can be used for other online robotic and AR applications such as obstacle avoidance. We demonstrate the effectiveness and efficiency of our method on both indoor and outdoor benchmarks, achieving state-of-the-art performance.

1. Introduction

Camera localization amounts to determining the orientation and position of a camera from the image it captures. It is a critical component for many applications such as SLAM, location recognition, robot navigation, and augmented reality. Conventionally [30, 26, 25, 16, 15, 7], this problem is solved by first finding a set of feature correspondences between the input image and a reference scene (either represented by a point cloud or a set of reference images), and then estimating the camera pose by minimizing some energy function defined over these correspondences. This approach is often brittle due to the long pipeline of hand-crafted components, i.e., image retrieval, feature correspondence matching, and camera pose estimation.

Recently, learning based approaches [18, 4, 11, 3] have advanced the performance of camera localization with random forests (RFs) and convolutional neural networks (CNNs). Some works [11, 10, 32, 2] directly regress the camera poses using CNNs; while others [9, 31, 27, 3, 5] first estimate a scene coordinate map, defining the `xyz` coordinates of each pixel in the query image, then compute camera pose accordingly, which is a well posed optimization problem. These methods often enjoy better robustness than the conventional pipeline. However, the RFs or CNNs are typically learned from images for a specific scene and require retraining or adaptation before they can be applied to a different scene. While online adaptation is possible, it has been only demonstrated with RGBD query images and a RF [6]. CNNs typically produce higher localization accuracy, but it is unknown how a CNN can be quickly adapted to a different scene, which limits their applications.

We follow this learning based approach for camera localization and aim to build a scene agnostic network that works on unseen scenes without retraining or adaptation. This capability is important for online applications such as SLAM and robotic navigation where retraining is impossible. We focus on the problem of scene coordinate map estimation, and adopt a similar method as [5] to compute the camera pose from the estimated scene coordinate map, while striving to make this process scene agnostic. The estimated dense scene coordinate map might be further applied for other applications such as robot obstacle avoidance.

To achieve this goal, we design a network, SANet, to extract a scene representation from some reference scene images and 3D points, instead of encoding specific scene information in network parameters, so that our network can be applied to different scenes without any retraining or adaptation. Specifically, our scene representation is a hierarchical pyramid of features at different resolutions. At querying time, this scene representation is combined with features from the query image to predict a dense scene coordinate map from coarse to fine. Intuitively, the network learns to fit the query image onto the 3D scene surfaces in a visually consistent manner. In order to fuse query image features and scene features, we adopt a similar structure as the PointNet [21, 22], which can handle unordered point clouds, to predict the scene coordinate features.

To demonstrate the effectiveness of the proposed method, we have evaluated our method on several benchmark datasets including indoor scenes (*7Scenes* [27]) and

[*] These authors contributed equally to this work.

outdoor scenes (*Cambridge* [11]). We have achieved state-of-the-art performance, while our method works for arbitrary scenes without retraining or adaptation. The code is available at `https://github.com/sfu-gruvi-3dv/sanet_relocal_demo`.

2. Related Work

Feature Matching & Camera Fitting: In conventional methods, a few neighboring images are first retrieved, a set of 2D-3D correspondences are then matched between the query image and 3D scene points based on some hand-crafted descriptors, and camera poses are finally recovered by the PnP algorithms [8, 13]. These works focus on making the hand-crafted descriptor either more efficient [17, 26], more robust [24, 29], or more scalable to large outdoor scenes [15, 23, 26]. However, the hand-crafted feature detectors and descriptors only work with well textured images. Recently, InLoc [30] pushed forward this direction by replacing the hand-crafted features by CNNs, i.e. NetVLAD [1] for image retrieve and VGG [28] for feature matching. While achieving strong performance, it is still based on the conventional pipeline of correspondence matching and camera model fitting. In comparison, learning based methods (including our work) can regress a scene coordinate map from the query image directly, which has the advantages of exploiting global context information in the image to recover 3D structures at textureless regions. Besides better robustness, the dense scene coordinate map as a dense 3D reconstruction might be used for robot obstacle avoidance or other applications.

Random Forests: Shotton *et al.* [27] proposed to regress the scene coordinates using a Random Forest, and this pipeline was extended in several following works. Guzman-Rivera *et al.* [9] trained a random forest to predict diverse scene coordinates to resolve scene ambiguities. Valentin *et al.* [31] trained a random forest to predict multi-model distributions of scene coordinates for increased pose accuracy. Brachmann *et al.* [4] addressed camera localization from an RGB image instead of RGB-D, utilizing the increased predictive power of an auto-context random forest. None of these works are scene agnostic. A recent work [6] has generalized this approach to unseen scenes with an RGBD camera and online adaptation. Compared with these works, our method is scene agnostic and only requires an RGB image for camera localization, which is appliable in both indoor and outdoor scenes.

Convolutional Neural Networks: CNN based methods have brought major advancements in performance. PoseNet [11] solved camera localization as a classification problem, where the 6 DOF camera poses are regressed directly. Several follow-up works further improved the training losses [10] or utilized the temporal dependency of videos to improve localization accuracy [32]. The recent work [2] learned a continuous metric to measure overlap between images, and the relative camera pose was regressed between the query image and its closest neighbor. Instead of direct camera pose regression, more recent works use CNNs to regress the scene coordinates as intermediate quantities [14, 3, 5] because the following camera pose estimation from a scene coordinate map is a well-behaved optimization problem. Our method belongs to this categories and use CNNs to predict the scene coordinate of an image. However, our network extracts hierarchical features from scenes rather than learn a set of scene-specific network parameters. In this way, our method is scene agnostic and can be applied to unknown scenes.

3. Overview

The overview of our pipeline is illustrated in Figure 1. The inputs to our method are a set of scene images $\{\mathbf{I}_s\}_{all}$ with its associated 3D point cloud $\{\mathbf{X}_s\}_{all}$ and a query image $\mathbf{q}$ captured in the same scene. The output is an estimated 6D camera pose $\mathbf{\Theta}_q = [\mathbf{R}_q|\mathbf{t}_q]$ of the query image.

To narrow down the input space for better efficiency and performance, we first use NetVLAD [1] to retrieve n nearest neighbors of the query image from all scene images. Then, we propose a network to regress the scene coordinate map of the query by interpolating the 3D points associated with the retrieved scene images (Section 4). The interpolation is done by firstly constructing hierarchical representations of the scene and the query image respectively through our network, named as scene pyramid and query feature pyramid (Section 4.1). With two constructed pyramids, we design two modules: Query-Scene Registration (QSR) and Fusing, and apply them iteratively to regress the scene coordinate map in a coarse-to-fine manner (Section 4.2). The architecture is End-To-End trainable to perform both tasks of constructing the pyramids and predicting dense scene coordinate map. In the end, the camera pose of query image can be estimated by RANSAC+PnP as in [5] (Section 4.3).

4. Method

4.1. Constructing Pyramids

Scene Pyramid: A scene retrieved by NetVLAD [1] contains a collection of n reference RGB images $\{\mathbf{I}_s|s = 1, ..., n\}_{vlad}$ (256×192 pixels in our implementation), each of which is associated with a scene coordinate map $\mathbf{X}_s \in \{\mathbf{X}_s\}_{vlad}$ (either dense or sparse) defined in the world coordinate system. We represent a scene as a pyramid that encodes both geometry and appearance information at different scales. Each pyramid level consists of a set of 3D point coordinates appended with their image features extracted by a CNN.

To construct such a scene pyramid, we first extract features from each scene image $\mathbf{I}_s$ by a convolutional neural

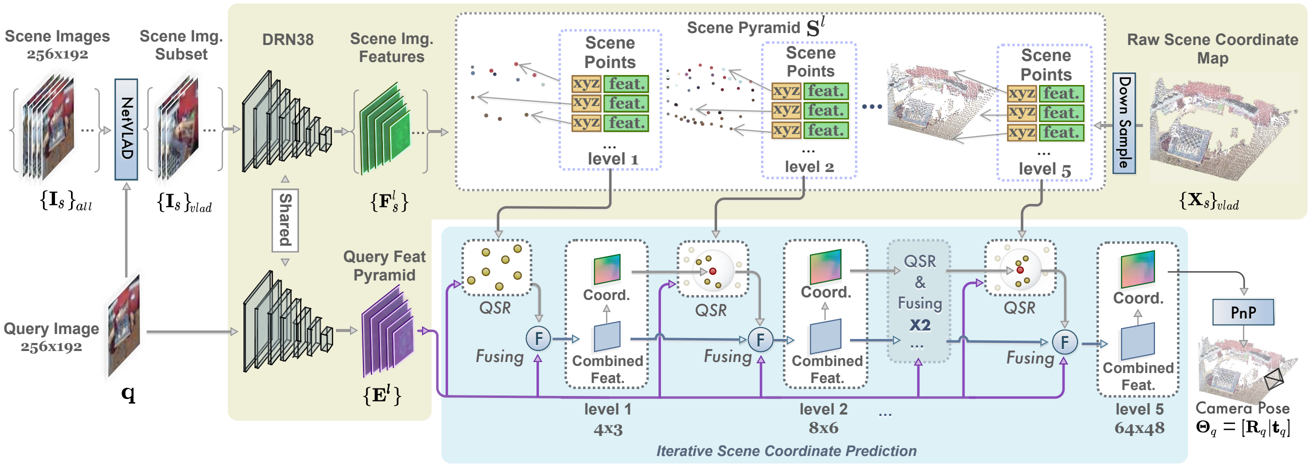


Figure 1: Pipeline of our system. The top branch constructs the scene pyramid, while the bottom branch predict a scene coordinate map for the query image.

network. Specifically, we use the Dilated Residual Network (DRN38) [34] as our feature extractor and obtain feature maps at different resolutions by removing all dilations and applying stride-2 down-sampling at the 1st ResBlock of each resolution level. We extract the feature maps $\{\mathbf{F}_s^l | l = 1, ..., 5\}$ from DRN38 with resolution from 4×3 up to 64×48 by a factor of 2. The weights of the network are shared by all scene frame images.

In addition, the scene coordinate maps $\mathbf{X}_s$ are scaled to match the resolution of feature maps at different levels. Here, we reduce their resolution by applying the Average Pooling filter with a 2×2 kernel, and ignore non-existing points when inputs are sparse. The scaled scene coordinate maps are denoted as $\{\tilde{\mathbf{X}}_s^l | l = 1, ..., 5\}$.

Finally, at each level l, we extract all pixels with valid scene coordinates $\{\tilde{\mathbf{x}}_i^l\}$ and concatenate them with corresponding feature vectors $\{\mathbf{f}_i^l\}$. This way, we get the scene pyramid $\mathbf{S} = \{\mathbf{S}^l | l = 1, ..., 5\} = \{(\mathbf{f}_i^l, \tilde{\mathbf{x}}_i^l) | l = 1, ..., 5; i = 1, ..., m^l\}$, which is a multi-scale point clouds equipped with image feature vectors, where m^l is the number of points at level l.

Query Feature Pyramid: The feature pyramid of query image $\{\mathbf{E}^l | l = 1, ..., 5\}$ is extracted by the same DRN38 network used for constructing the scene pyramid $\mathbf{S}$, thus with identical feature dimensions as $\{\mathbf{F}_s^l | l = 1, ..., 5\}$.

4.2. Predicting Scene Coordinate

Given these two types of pyramids $\mathbf{S}$ and $\mathbf{E}$, we predict the scene coordinate map $\hat{\mathbf{Y}}$ for the query image $\mathbf{q}$. To better encode the global scene context and speedup computation, we take a coarse-to-fine strategy to predict the scene coordinate map. The network first produces a coarse scene coordinate map at resolution of 4×3 as a rough estimation, then finer maps are refined iteratively level by level to yield more detailed predictions.

At each iteration, we sequentially apply two modules to predict the scene coordinate map: (1) The Query-Scene Registration (QSR) module learns to register each query feature pixel into the 3D scene space by interpolating the scene coordinates of scene pyramid points based on visual similarity; (2) The Fusing module fuses cross-pixel image context information to regularize the pixel-wise registration given by the QSR. For simplification, we will treat the QSR module as a black box for now to explain the iterative scene coordinate map prediction. We will discuss the QSR module in detail later. The iterative pipeline is illustrated in Figure 2.

At the initial level $l = 1$, we predict the coarsest scene coordinate map $\hat{\mathbf{Y}}^1$ from the query image feature $\mathbf{E}^1$ and the scene point clouds $\mathbf{S}^1$. First, we register the feature $\mathbf{E}^1$ of each pixel to the 3D scene space by feeding $\mathbf{E}^1$ and $\mathbf{S}^1$ into the QSR module. The registration results are encoded into a *scene reference feature* $\mathbf{R}^1$. Second, since each pixel of $\mathbf{R}^1$ is computed independently and may contain erroneous predictions (e.g. featureless area), we further regularize the QSR registration by fusing with query image feature $\mathbf{E}^1$ that encodes the cross-pixel contextual information as the additional geometry constraint. Specifically, $\mathbf{R}^1$ is concatenated with the query image feature $\mathbf{E}^1$, and send to a ResBlock to yield a *combined feature* $\mathbf{C}^1$, which is decoded by a 1×1 Conv to get the prediction $\hat{\mathbf{Y}}^1$. Note that both $\hat{\mathbf{Y}}^1$ and $\mathbf{C}^1$ will be utilized by the next level.

The finer levels $l > 1$ share a similar process as the initial one but taking in additional inputs, i.e., the combined feature $\mathbf{C}^{l-1}$ and prediction $\hat{\mathbf{Y}}^{l-1}$ from previous level, which are up-sampled by nearest neighbor interpolation yielding $up(\mathbf{C}^{l-1})$ and $up(\hat{\mathbf{Y}}^{l-1})$ to match the resolution of current level. The QSR module takes the query image feature

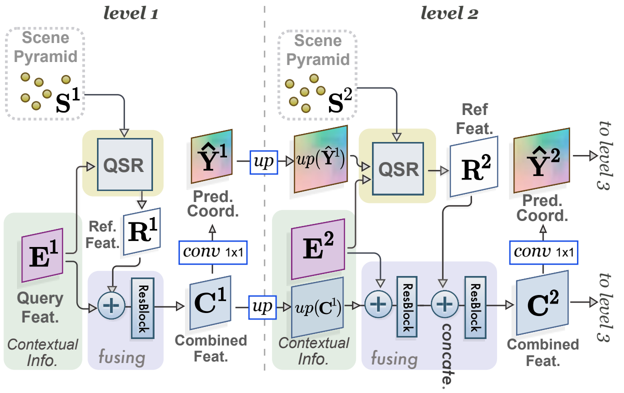


Figure 2: Iterative scene coordinates prediction from level 1 to level 2. For level $l > 2$, the operation is identical to level 2, please refer to main text for details.

$\mathbf{E}^l$, the point cloud $\mathbf{S}^l$, as well as the previous prediction $up(\hat{\mathbf{Y}}^{l-1})$ as inputs to produce a scene reference feature $\mathbf{R}^l$. Then, we fuse $\mathbf{E}^l$ with $up(\mathbf{C}^{l-1})$ from previous level to better leverage high-level image context. The resulting feature map is further fused with $\mathbf{R}^l$ and decoded by 1×1 `Conv`, producing the current level combined feature $\mathbf{C}^l$ and prediction $\hat{\mathbf{Y}}^l$.

Query-Scene Registration (QSR): In QSR, the query image feature $\mathbf{E}^l$ is processed per pixel. For each pixel $\mathbf{p}$ in $\mathbf{E}^l$, the QSR module learns to register the feature vector $\mathbf{E}^l[\mathbf{p}]$ into the 3D scene point cloud $\mathbf{S}^l$, and encodes the registration result as a scene reference feature $\mathbf{R}^l[\mathbf{p}]$. Two sub-steps are involved in this module: (1) sample a subset of the scene point cloud $\mathbf{S}^l_{sub} = \{(\mathbf{f}^l_i, \tilde{\mathbf{x}}^l_i) | i = 1, ..., k^l\} \subseteq \mathbf{S}^l$, where k^l is the number of samples; (2) execute the registration through a PointNet [21]-like architecture with $\mathbf{E}^l[\mathbf{p}]$ and $\mathbf{S}^l_{sub}$ as inputs; see Figure 3 for illustration.

At the level l, to narrow down the solution space and let the network focus on local details, we sample a subset of scene points $\mathbf{S}^l_{sub} \subseteq \mathbf{S}^l$ from the neighbourhood of previous prediction $up(\hat{\mathbf{Y}}^{l-1})[\mathbf{p}]$, which is inspired by the `Sampling & Grouping` strategy in PointNet++ [22]. The neighbourhood is defined as a sphere with radius r^l. When $l = 1$, we simply have $\mathbf{S}^1_{sub} = \mathbf{S}^1$, namely all scene points are used.

Given the query image feature $\mathbf{E}^l[\mathbf{p}]$ and scene points $\mathbf{S}^l_{sub}$, we design a network based on PointNet [21] to process the query-to-scene registration. PointNet only processes the positional information encoded in the `xyz` coordinates. However, our network needs to filter the positional information according to the appearance correlation between the query pixel and the 3D scene. To do so, we adopt an `MLP` to extract the appearance correlation between the query image feature $\mathbf{E}^l[\mathbf{p}]$ and the scene appearance feature $\mathbf{f}^l$ of each scene point. The resulting features are appended with the 3D coordinates $\tilde{\mathbf{x}}^l$ and fed into a PointNet [21] (without Spatial Transform) to produce the scene reference feature $\mathbf{R}^l[\mathbf{p}]$ that encodes the registration result. Please refer to the supplementary for detail configurations.

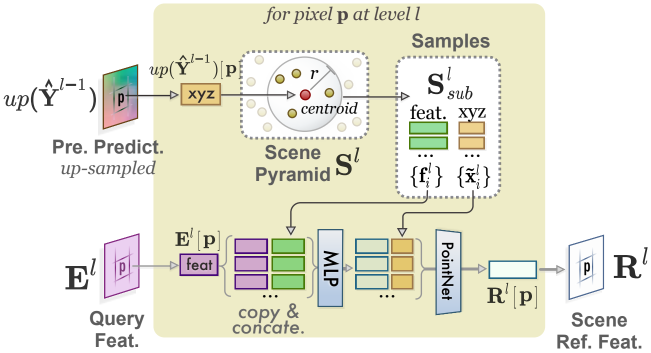

for pixel $\mathbf{p}$ at level l
Samples
$up(\hat{\mathbf{Y}}^{l-1})$
$up(\hat{\mathbf{Y}}^{l-1})[\mathbf{p}]$
p
xyz
r
centroid
$\mathbf{S}^l_{sub}$
feat.
xyz
Pre. Predict.
up-sampled
Scene Pyramid $\mathbf{S}^l$
$\{\mathbf{f}^l_i\}$
$\{\tilde{\mathbf{x}}^l_i\}$
$\mathbf{E}^l[\mathbf{p}]$
$\mathbf{E}^l$
p
feat
MLP
PointNet
p
$\mathbf{R}^l$
$\mathbf{R}^l[\mathbf{p}]$
Query Feat.
copy & concate.
Scene Ref. Feat.

Figure 3: Query-Scene Registration(QSR) at level l.

Training Loss: We train the network for scene coordinate prediction by supervised training with ground truth scene coordinate maps of query images. Here, we apply the L_2-norm loss [5] averaged over all pixel $\mathbf{p}$ and pyramid levels l as bellow,

$$L = \sum_l \sum_{\mathbf{p}} \mathbf{v}^l[\mathbf{p}] \left\| \mathbf{Y}^l[\mathbf{p}] - \hat{\mathbf{Y}}^l[\mathbf{p}] \right\|, \tag{1}$$

where $\mathbf{Y}^l, \hat{\mathbf{Y}}^l$ are the ground truth and predicted scene coordinate map respectively; $\mathbf{v}^l[\mathbf{p}] = 1$ if the ground-truth exists for pixel $\mathbf{p}$ at level l, otherwise $\mathbf{v}^l[\mathbf{p}] = 0$.

4.3. Query Pose Estimation

Given a predicted scene coordinate map $\hat{\mathbf{Y}}$ at the highest resolution, the 6D camera pose Θ_q of the query image can be estimated by the PnP algorithm [8, 13]. Since the scene coordinate map contains outliers, a collection of k 4-point tuples are randomly sampled from predicted coordinates. The hypothesis pose set $\mathbf{H} = \{h_j | j = 1, ..., k\}$ can be estimated by solving the PnP problem for each 4-point tuple. We then find the best hypothesis h^* that is most coherent with the predicted scene coordinate map. The final camera pose Θ_q is obtained by further refining h^* using all inliers. Interested readers are referred to DSAC++ [5] for more details.

5. Experiments

Datasets: We train and evaluate our method on both indoor and outdoor scenes. For indoor scenes, we train and evaluate our pipeline on *SUN3D* [33] and *7Scenes* [27] datasets, both capturing raw indoor RGB-D video sequences with the ground-truth camera poses provided. *SUN3D* contains more than 300 scene sequences, while *7Scenes* consists of 7 different indoor scenes. For *7Scenes*, each scene is divided into multiple `train` and `test` sequences with a thousand of frames each. We train our model with *SUN3D* sequences, and evaluate on the *7Scenes* dataset. For each test scene, we build the scene pyramid with the `train` frames and use the frames in the `test` sequences as queries.

The *Cambridge Landmarks* [11] is used for evaluating our pipeline on outdoor scenes. The dataset contains 6 different outdoor scenes with only RGB video frames available. To obtain 3D point clouds, we use the dense MVS reconstruction provided by [5]. Before testing on a specific scene, we finetune our network, which is pre-trained on *SUN3D*, with the remaining 5 scenes. When testing, similar to *7Scenes*, the scene pyramid is constructed with `train` frames of the testing scene, and the `test` frames are used as queries for evaluating the trained model.

Comparison: We compare estimated camera poses, method efficiency, and predicted scene geometry against various scene-independent and scene-specific methods. Active Search [26] and InLoc [30] are scene independent, which takes a 3D point cloud and some reference images as input, and utilize traditional hand-crafted or pre-trained deep features for matching query patches to the scene points respectively. Random Forest based approaches [18, 4], DSAC [3], and DSAC++ [5] are scene specific, which train their model and predict scene coordinate maps on the same scene. The pose regression based method [11] that trained with 3D geometry loss is also compared.

5.1. Implementation Details

Training Samples: In order to train our network, we need to construct training samples beforehand. Each training sample consists of 5 scene frames for building scene pyramid, and 1 query image.

From the *SUN3D* indoor dataset, we randomly select 388 video sequences. For each sequence, we randomly choose 10% of all frames as anchors. For each anchor frame, we collect 4 other scene frames in the sequence, where each new frame either shares more than 20% pixels with the previous frame, or translates less than 2 meters from the previous frame. In the final step, 50 query frames are randomly collected between the first and the fifth frame. In the end, we have more than 200K training samples.

As for the *Cambridge Landmarks* outdoor dataset, we finetune the network using 5 out 6 scenes, and test on the remaining one. All frames in training scenes are used as query frames, and for each query we run NetVLAD [1] to retrieval 100 closest candidate frames. Among these 100 candidates, we randomly choose 5 scene frames with relative translation less than 2 meters and relative rotation less than 45°. This way, we obtain in total 15K-30K training samples for each test scene.

Training: Our network is implemented with PyTorch [20], and trained on 1080Ti with 11G memory. When training the network, we set the sampling radius $r = [1.5m, 0.75m, 0.5m, 0.25m]$ for sampling local scene points (Section 4.2) at level 2 to 5 respectively. For levels $l > 1$, the number of sampled points in QSR is set to $k^l = 64$. For each training sample, we normalize their scene coordinates by subtracting the scene center, which is computed by averaging all scene coordinates in 3D space. Due to different scales of outdoor scenes in the *Cambridge Landmark* dataset, we scale their scene coordinates into a $5m \times 5m \times 5m$ cube. To augment the data, we add random rotation around the *up* direction to scene coordinates, and brightness jittering to scene RGB images. The optimizer Adam [12] with initial learning rate of 2.0×10^{-4} is applied, and the batch size is set to 6.

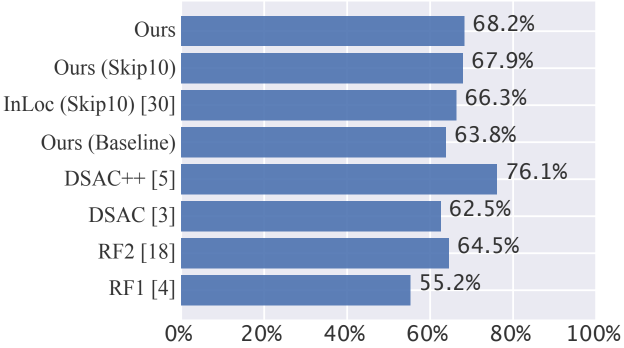


Figure 4: Percentage of predicted camera poses falling within the threshold of $(5°, 5cm)$ on *7Scenes* indoor dataset by `RF1` [4], `RF2` [18], `DSAC++` [5], `DSAC` [3], `InLoc(Skip10)` [30], and our approaches.

Inference: Limited by the size of GPU memory, for each query image, we retrieve 10 nearest scene frames by NetVLAD [1] to build the scene pyramid. The retrieval step is accelerated with `knncuda` library. We set the sampling radius with $r = [0.75m, 0.5m, 0.25m, 0.125m]$ for both indoor and outdoor scenes[1] and set the number of sampled points $k^l = 64$ for levels $l > 1$. Note that we uniformly scale all scene coordinates into a $5m \times 5m \times 5m$ cube for outdoor scenes.

5.2. Localization Accuracy

We first measure localization accuracy in terms of percentage of predicted poses falling within the threshold of $(5°, 5cm)$. Figure 4 shows comparison results with other methods on *7Scenes*. Our method outperforms the Random Forest based methods [18, 4], and DSAC [3]. Since the inference of InLoc [30] takes a considerable amount of time for all query frames, we only run InLoc algorithm on 1 out of every 10 testing frames, whose performance is denoted as *InLoc (Skip10)*. Under the same setting, our method *Our (Skip10)* outperforms *InLoc (Skip10)* by 1.6%. Note that

[1] Except for `Street` and `Great Court`, for which we use $r = [1.5m, 0.75m, 0.5m, 0.25m]$.

Table 1: Indoor and Outdoor localization accuracy on *7Scenes* and *Cambridge*.

	Scene Specific						Scene Independent					
	PoseNet (Geo) [10]		DSAC [3]		DSAC++ [5]		Active Search [26]		InLoc [30]		Ours	
7Scenes												
Chess	4.5°	0.13m	0.7°	**0.02m**	**0.5°**	**0.02m**	1.96°	0.04m	1.05°	**0.03m**	**0.88°**	**0.03m**
Fire	11.3°	0.27m	1.0°	0.03m	**0.9°**	**0.02m**	1.53°	**0.03m**	**1.07°**	**0.03m**	1.08°	**0.03m**
Heads	13.0°	0.17m	1.3°	0.02m	**0.8°**	**0.01m**	1.45°	**0.02m**	**1.16°**	**0.02m**	1.48°	**0.02m**
Office	5.6°	0.19m	1.0°	**0.03m**	**0.7°**	**0.03m**	3.61°	0.09m	1.05°	**0.03m**	**1.00°**	**0.03m**
Pumpkin	4.8°	0.26m	1.3°	0.05m	**1.1°**	**0.04m**	3.10°	0.08m	1.55°	**0.05m**	**1.32°**	**0.05m**
Kitchen	5.4°	0.23m	1.5°	0.05m	**1.1°**	**0.04m**	3.37°	0.07m	**1.31°**	**0.04m**	1.40°	**0.04m**
Stairs	12.4°	0.35m	49.4°	1.9m	**2.6°**	**0.09m**	**2.22°**	**0.03m**	2.47°	0.09m	4.59°	0.16m
Cambridge												
Great Court	3.7°	7.0m	1.5°	2.80m	**0.2°**	**0.40m**	-	-	**0.62°**	**1.20m**	1.95°	3.28m
King's College	1.1°	0.99m	0.5°	0.30m	**0.3°**	**0.18m**	0.6°	0.42m	0.82°	0.46m	**0.54°**	**0.32m**
Old Hospital	2.9°	2.17m	0.6°	0.33m	**0.3°**	**0.20m**	1.0°	0.44m	0.96°	0.48m	**0.53°**	**0.32m**
Shop Facade	4.0°	1.05m	0.4°	0.09m	**0.3°**	**0.06m**	**0.4°**	0.12m	0.50°	0.11m	0.47°	**0.10m**
St. Mary's Church	3.4°	1.49m	1.6°	0.55m	**0.4°**	**0.13m**	**0.5°**	0.19m	0.63°	0.18m	0.57°	**0.16m**
Street	**25.7°**	**20.7m**	-	-	-	-	**0.8°**	0.85m	2.16°	**0.75m**	12.64°	8.74m

DSAC++ [5] still performs the best as it is trained for each scene specifically. Ours(Baseline) is a conventional feature matching approach utilizing features from our network, which will be discussed in Section 5.5.

Secondly, localization accuracy is measured in terms of median translation and rotation error. Comparison results on the *7Scenes* indoor dataset are listed in the top half of Table 1, where reported numbers are from original papers. Comparing with scene specific methods, our approach outperforms PoseNet [10] on most scenes, while is slightly inferior to the state-of-the-art method DSAC/DSAC++ [3, 5]. Comparing with online methods, we outperform Active Search [26] on most scene except for Stairs, as our network is trained with the *SUN3D* indoor dataset that is lack of such extremely repetitive pattern. Our method yields comparable results to the InLoc [30] algorithm, especially slightly better on Chess, Office, and Pumpkin.

Table 1 also shows the localization accuracy on outdoor scenes. Comparing with scene specific model, our performance is inferior to DSAC [3] and DSAC++ [5] while still better than PoseNet [11]. Comparing with scene independent methods, our method yields comparable results with Active Search [26], and slightly better results than InLoc [30] on 4 scene, i.e., King's College, Old Hospital, Shop Facade and St. Mary's Church. We produce inferior results on Great Court and Street, due to the ambiguous patterns in large-scale scenes and sharp illumination changes in query images. Note that we resize the input scene frame resolution to 480×270 to speed up the InLoc algorithm.

5.3. Scene Coordinate Accuracy

We compare the accuracy of scene coordinate maps against InLoc [30] and DSAC/DSAC++ [3, 5] on the *7Scenes* dataset. Figure 6 shows the cumulative distribution function (CDF) of errors in the estimated scene coordinate maps. For a given error threshold, this CDF gives the percentage of pixels whose estimated scene coordinates are within the error threshold (i.e., the recall rate). We outperform the DSAC [3] after the 4cm threshold, even though DSAC [3] is specifically trained for each scene. Note that while DSAC++[5] demonstrates the best localization accuracy, its scene coordinate is comparable to DSAC[3], as the DSAC++[5] is trained with the less accurate rendered depth from a fused mesh. In addition, it indicates that the scene coordinate precision may not reflect the final pose accuracy, since the outliers are filtered by RANSAC. InLoc [30] removes invalid matchings by consistency check and produces highest precision over remaining pixels InLoc (pred. pixels), but those predicted pixels are limited to semi-dense level, as the CDF of errors over all pixels InLoc (all pixels) quickly saturates at around 38%.

Figure 5 visualizes several predicted scene maps as color coded coordinates as well as triangle meshes. In constrast to InLoc [30], our method produces dense predictions and is even robust at featureless areas, because our network exploit global context in the query image when decoding the scene coordinate map. For example, InLoc [30] cannot produce reasonable results on the wall and the ceiling, while our method can. The dense prediction enables applications other than localization, such as robotic obstacle avoidance.

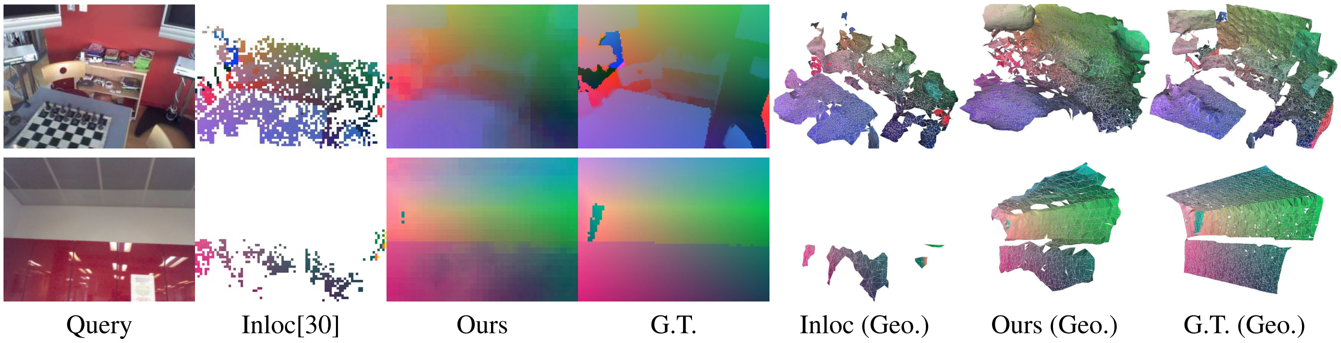


Figure 5: Scene coordinate map comparison with InLoc [30] and the ground truth (G.T.) on *7Scenes*, the scene coordinate positions `xyz` are encoded in `rgb` channels for visualization. The last three columns show the geometry (Geo.) comparison by reconstructing the mesh from the scene coordinate map. More examples are found in supplementary document.

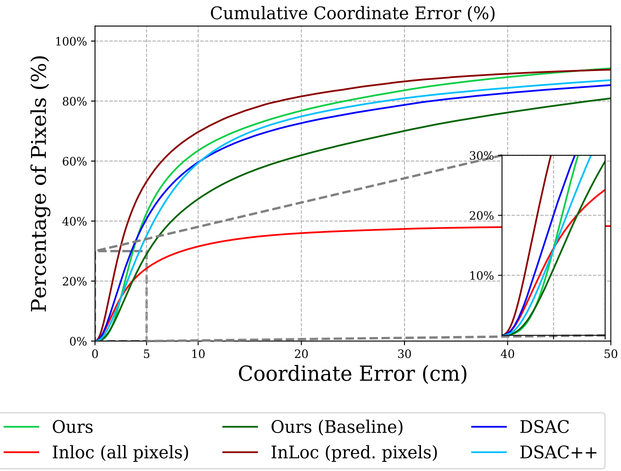


Figure 6: Cumulative Distribution Function of scene coordinate error compared with InLoc [30], DSAC [3] and DSAC++ [5] on *7Scenes*.

5.4. Efficiency

Our system is efficient enough for real-time scene update and query pose estimation, enabling SLAM applications.

Time Costs: Table 2 lists the time of each step w.r.t 7000 scene images in *7Scenes*, with comparisons to DSAC++ [5] and InLoc [30]. Taking the ORB-SLAM[19] as an example, it creates a new keyframe for approximately every 0.7s for TUM RGB-D SLAM indoor sequences. Our method and InLoc [30] can index an incoming keyframe on-the-fly by a NetVLAD [1] forward pass (avg. 0.06s), while DSAC++ [5] runs multiple epochs to train the model, taking several days as reported in their paper. When the loop detection or the re-localization are activated, assuming 7000 keyframes are indexed, our method takes only 0.54s to localize a query frame (faster than the keyframe creation), which meets the real-time requirement of SLAM, while InLoc [30] takes several seconds based on its public implementation (CPU version, using 6 threads in parallel).

Memory Costs: For each scene image, our method requires an RGBD image (256×192), its pose, and its VLAD feature, whose memory cost is 248kB.[2] With 7000 scene images, total 2.66GB (including 1GB for the SANet forward pass) is used. InLoc [30] has a similar memory consumption with ours as both methods rely on NetVLAD [1]. DSAC++ [5] only needs a small amount of constant memory since the whole scene is compressed into network weights, but cannot handle novel scenes without retraining.

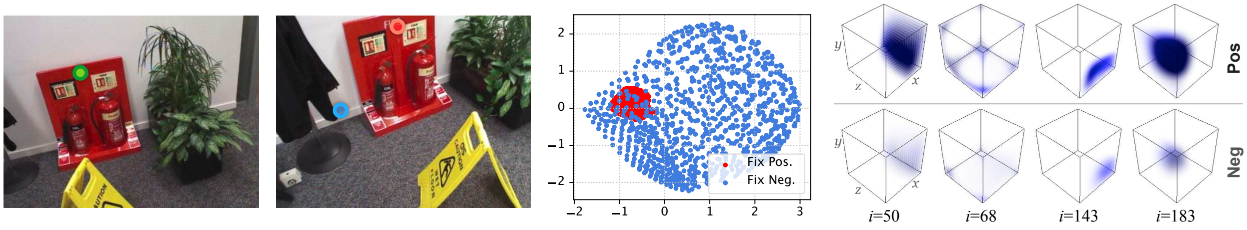
Table 2: Time for each step w.r.t 7000 scene images.

Steps	**Ours**	**DSAC++** [5]	**InLoc** [30]
Training CNN or Index. VLAD feat (all scene imgs.)	427s	> 1 day	427s
Retrieval (per query)	171ms	-	171ms
Estimate Pose (per query)	0.37s	0.2s	9.38s
Total (per query)	0.54s	0.2s	9.55s

5.5. Detailed Analysis

Compare with Explicit Matching: We evaluate the effectiveness of our network against the conventional feature matching pipeline. Specifically, we design a *baseline* approach, where features $\mathbf{E}^l[\mathbf{p}]$ from the query image are directly matched to sampled features $\mathbf{f}_i^l \in \mathbf{S}_{sub}^l$ in the scene pyramid by their angular similarity. Such baseline approach has two drawbacks comparing with our network: firstly, it processes pixels independently thus cannot take advantages of the global image context during the scene coordinate prediction; secondly, the distance metric of angular similarity might not be optimal. We report the accuracy of the estimated camera pose in Figure 4 and that of the scene coordinate map in Figure 6, both are denoted as `Ours (Baseline)`. It is clear that our proposed method

[2] 120kB for PNG compressed RGBD frame, 128kB for VLAD feat.



(a) (b) (c) (d)

Figure 7: (a) A pixel `Que` in the query image is marked in Green. (b) The ground truth corresponding pixel `Pos` in a scene reference image is marked in Red, while a randomly selected pixel `Neg` is marked in Blue. (c) Two sets of scene reference features from *Experiment-1* are projected in 2D space by PCA. (d) Four channels (i.e., the i-th chanels) of scene reference features from *Experiment-2* are plotted w.r.t. a $1m \times 1m \times 1m$ cube, where drak blue stands for high channel activation. Please refer to the main text for details.

surpasses this baseline approach with unnegligible margins on both evaluation metrics.

Scene Reference Feature: We further design two experiments to explore physical meanings of the scene reference feature $\mathbf{R}^l$.

Experiment-1: $\mathbf{R}^l$ *encodes the scene coordinates of corresponding pixels.* We randomly select a pixel `Que` from the query image feature map, whose location is marked by the green dot in Figure 7 (a). Its ground truth corresponding pixel `Pos` in the scene image feature map is marked by a red dot in Figure 7 (b), while `Neg` is a random irrelevant pixel marked by the blue dot. Now, we encode only two points, `Pos` and `Neg`, in our scene pyramid, and pass the query pixel `Que` through the network to generate its scene reference feature $\mathbf{R}^3[\texttt{Que}]$. Here, we choose $l = 3$ with resolution of 16×12 for experiment, and each feature pixel has 256 channels.

We add noise to the `xyz` coordinates of the `Pos` or `Neg` pixel while fixing the other to see how the scene reference feature $\mathbf{R}^3[\texttt{Que}]$ will be affected. Specifically, we first fix `Pos` while varying the position of `Neg` within an unit cube, which is centered at its original location with an edge length of $1m$. This way, we obtain a set of scene reference features $\{\mathbf{R}^3[\texttt{Que}]\}_{\text{fix-pos}}$. Accordingly, fixing `Neg` and varying the position of `Pos` generates another set of scene reference features $\{\mathbf{R}^3[\texttt{Que}]\}_{\text{fix-neg}}$.

Figure 7 shows both sets of scene reference features by projecting them in 2D space via PCA, where $\{\mathbf{R}^3[\texttt{Que}]\}_{\text{fix-pos}}$ and $\{\mathbf{R}^3[\texttt{Que}]\}_{\text{fix-neg}}$ are visualized as red and blue dots respectively. It is clear that the red dots vary much less than the blue dots. In other words, changing the position of the ground truth scene point leads to much larger variation of the scene reference feature. This is a strong evidence that our scene reference feature encodes scene coordinates of the corresponding pixels.

Experiment-2: *Each channel of* $\mathbf{R}^l$ *is responsible for a particular region in 3D space.* In the second experiment, we encode one single point `Pos` in the scene pyramid. We uniformly sample a set of positions $\mathbf{X}$ within the $1m \times 1m \times 1m$ cube centered at `Pos`. Given the pixel `Pos` at position $\mathbf{x}$ and the query pixel `Que` as input, the QSR module generates a scene reference feature $\mathbf{R}^3[\texttt{Que}]_{\text{pos}}(\mathbf{x})$. This way, we can collect a set of scene reference features $\{\mathbf{R}^3[\texttt{Que}]_{\text{pos}}(\mathbf{x}) | \mathbf{x} \in \mathbf{X}\}$. In the top row of Figure 7(d), we plot 4 channels of these features w.r.t. the set of positions $\mathbf{X}$ in a cube. The values are normalized for visualization, where the darker blue color stands for higher channel value. As we can see, each channel is correlated to a particular spatial region, for example, the 68th channel has high response on eight corners of the cube. We run the same experiment for the `Neg` sample, and plot the same four channels in the bottom row of Figure 7 (d). Comparing with the positive sample `Pos`, it shows similar activation patterns but with different magnitude. This further verifies that our scene reference feature encodes spatial information.

6. Conclusion

This paper presents a scene agnostic network architecture that predicts the dense scene coordinate map of a query RGB image on-the-fly in an arbitrary environment. The coordinates prediction is then used for estimating the camera pose. Our network learns to not only encode the scene environment into a hierarchical representation, but also predict the scene coordinate map with query-scene registration. In particular, we design a learnable module that iteratively registers a query image to the scene at different levels, and yields the dense coordinate map regularized by contextual image information. Our network is validated on both indoor and outdoor dataset, and achieves state-of-the-art performance.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly

supervised place recognition. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 5297–5307, 2016.

[2] Vassileios Balntas, Shuda Li, and Victor Adrian Prisacariu. Relocnet: Continuous metric learning relocalisation using neural nets. In *Proc. of European Conference on Computer Vision (ECCV)*, September 2018.

[3] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 6684–6692, 2017.

[4] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016.

[5] Eric Brachmann and Carsten Rother. Learning less is more-6d camera localization via 3d surface regression. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 4654–4662, 2018.

[6] Tommaso Cavallari, Stuart Golodetz, Nicholas A Lord, Julien Valentin, Luigi Di Stefano, and Philip HS Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 4457–4466, 2017.

[7] Michael Donoser and Dieter Schmalstieg. Discriminative feature-to-point matching in image-based localization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 516–523, 2014.

[8] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 25(8):930–943, 2003.

[9] Abner Guzman-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew Fitzgibbon, and Shahram Izadi. Multi-output learning for camera relocalization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 1114–1121, 2014.

[10] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 5974–5983, 2017.

[11] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 2938–2946, 2015.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International Journal of Computer Vision (IJCV)*, 81(2):155, 2009.

[14] Xiaotian Li, Juha Ylioinas, and Juho Kannala. Full-frame scene coordinate regression for image-based localization. In *Robotics: Science and Systems (RSS)*, 2018.

[15] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 15–29. Springer, 2012.

[16] Yunpeng Li, Noah Snavely, and Daniel P Huttenlocher. Location recognition using prioritized feature matching. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 791–804. Springer, 2010.

[17] Hyon Lim, Sudipta N Sinha, Michael F Cohen, and Matthew Uyttendaele. Real-time image-based 6-dof localization in large-scale environments. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 1043–1050. IEEE, 2012.

[18] Daniela Massiceti, Alexander Krull, Eric Brachmann, Carsten Rother, and Philip HS Torr. Random forests versus neural networkswhat's best for camera localization? In *Proc. of International Conference on Robotics and Automation (ICRA)*, pages 5118–5125. IEEE, 2017.

[19] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[20] 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.

[21] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.

[22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5099–5108, 2017.

[23] Torsten Sattler, Michal Havlena, Filip Radenovic, Konrad Schindler, and Marc Pollefeys. Hyperpoints and fine vocabularies for large-scale location recognition. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 2102–2110, 2015.

[24] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale location recognition and the geometric burstiness problem. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 667–674. IEEE, 2011.

[26] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, (9):1744–1756, 2017.

[27] 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 *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2937, 2013.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] Linus Svarm, Olof Enqvist, Magnus Oskarsson, and Fredrik Kahl. Accurate localization and pose estimation for large 3d models. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 532–539, 2014.

[30] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2018.

[31] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 4400–4408, 2015.

[32] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using lstms for structured feature correlation. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 627–637, 2017.

[33] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 1625–1632, 2013.

[34] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, 2017.