SANet: Scene Agnostic Network for Camera Localization

by

Ziqian Bai

B.Sc., The Chinese University of Hong Kong, 2018

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2021

Declaration of Committee

Name:	**Ziqian Bai**
Degree:	**Master of Science**
Thesis title:	**SANet: Scene Agnostic Network for Camera Localization**
Committee:	**Chair:** Mo Chen Assistant Professor, Computing Science
	Ping Tan Supervisor Associate Professor, Computing Science
	Yasutaka Furukawa Committee Member Associate Professor, Computing Science
	Manolis Savva Examiner Assistant Professor, Computing Science

Abstract

This thesis 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 with scene coordinate regression, 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 6 DoF 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 among methods working for arbitrary scenes without retraining or adaptation.

Keywords: RGB Camera Localization; Scene Agnostic; Scene Coordinate Regression; Deep Learning; ICCV 2019

Dedication

Plead for mercy from God, my source of wisdom.

Acknowledgements

I would sincerely appreciate my supervisor Professor Ping Tan for his insightful thoughts, invaluable support, and patient guidance during the completion of this thesis. Thanks Ping for leading me into the academic field of 3D computer vision and continuing to shape my research taste. It is my great honor to work with Ping who always cares for his students.

I would also express my appreciation to Professor Yasutaka Furukawa, Luwei Yang, Chengzhou Tang, Honghua Li for their collaborations. Thanks Professor Yasutaka Furukawa for fruitful discussions through out the study. Also thanks Luwei Yang and Chengzhou Tang who mentored me during my first research project.

Deepest appreciation to my loving family. I wouldn't be able to achieve where I am without your unconditional love, encourage, and support through out my life.

I'm also grateful to all the people who helped me and shared this journey with me, including but not limited to, Feitong, Sicong, Zeshi, Jie, Fei-peng, Rakesh, Jamal, as well as all other lab members and my friends.

Thanks and wish you all the best!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The ability to perceive, understand, and analyze our 3D world has been one of the cornerstones for various intelligence systems such as Autonomous-driving, Augmented/Virtual Reality, Embodied AI and etc, which makes **3D Computer Vision** attract more and more research attentions. Among various 3D problems, one fundamental task is to answer "*Where am I?*" based on some visual inputs such as a query image, named as "*Camera Localization*". More specifically, camera localization amounts to determining the 6 degree-of-freedom orientation and position (6 DoF pose) of a camera from the image it captures in a reference scene. It is a critical component for many applications such as simultaneous localization and mapping (SLAM), location recognition, robot navigation, and augmented reality. Conventionally [40, 35, 34, 21, 20, 10], this problem is solved by first matching a set of feature correspondences between the query image and a reference scene (either represented by a 3D point cloud appended with feature descriptors or a set of reference images), and then estimating the 6 DoF camera pose by minimizing some energy function defined over these correspondences (e.g. re-projection error). This class of approaches are often brittle due to the hand-crafted feature descriptor matching pipeline.

Recently, learning based approaches [24, 4, 15, 3] have advanced the performance of camera localization with random forests (RFs) and convolutional neural networks (CNNs). Some works [15, 14, 44, 2] directly regress the camera poses using CNNs; while others [13, 42, 36, 3, 5] first estimate a scene coordinate map, defining the `xyz` coordinates of each pixel in the query image. The scene coordinate map constructs the 2D-to-3D correspondences, from which we can perform random sample consensus (RANSAC) to reject outliers and compute the camera pose with Perspective-n-Points (PnP), 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 and 3D data 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 only been demonstrated on RFs [8] with RGB-D query images. 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 when the scene is novel or progressively updated.

We follow these learning based approaches for camera localization and aim to design a scene agnostic neural model that works on unseen scenes without retraining or adaptation. This capability is important for online applications such as SLAM and robotic navigation where novel scenes or scene updates need to be handled in real-time, thus making expensive retraining 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 deep neural network named as SANet. Instead of learning to memorize specific scene information in network weights as previous methods, SANet learns to extract a scene representation, which is a hierarchical pyramid of features appended to 3D points, from some reference scene images and their scene coordinates (i.e. `xyz` positions of pixels in the world coordinate space). At querying time, this scene representation is combined with features from the query image to predict a dense scene coordinate map in a coarse to fine manner. In order to fuse the query image feature and the scene representation, we propose a PointNet [27, 28] inspired architecture, which can handle unordered 3D point clouds as well as leverage context information in the query image, to predict the scene coordinate map. Intuitively, SANet can be viewed as learning to fit the query image onto the 3D scene surfaces in a visually consistent manner. By explicitly extracting a scene representation, our SANet can be applied to different scenes without any retraining or adaptation.

To demonstrate the effectiveness of the proposed method, we evaluate our method on several benchmark datasets including indoor scenes (*7Scenes* [36]) and outdoor scenes (*Cambridge* [15]). We are able to achieve state-of-the-art performance among methods working for arbitrary scenes without retraining or adaptation. The code is available at
`https://github.com/sfu-gruvi-3dv/sanet_relocal_demo`.
This work was published as:

Luwei Yang*, **Ziqian Bai***, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, Ping Tan. "SANet: Scene Agnostic Network for Camera Localization". In *International Conference on Computer Vision* (ICCV), 2019. (* indicates equal contribution)

In the rest of this thesis, Chapter 2 introduces the related works of RGB camera localization where a 3D model of the reference scene is available. Chapter 3 illustrates the proposed SANet architecture for performing scene coordinate regression in a scene agnostic manner and how camera poses can be further computed. Chapter 4 shows various experimental results on camera pose accuracy, scene coordinate map quality, efficiency, and analysis of our method. Finally, Chapter 5 concludes our proposed SANet.

Chapter 2

Related Works

As a classic 3D computer vision task, camera localization has been heavily studied, deriving a vast literature. Thus it is impractical to give a comprehensive review in this thesis. Here we will briefly discuss related works of localizing an RGB query image in a reference scene whose images and 3D models (point clouds, meshes or depths + poses) are available.

2.1 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 [12, 17]. These works focus on making the hand-crafted descriptor either more efficient [22, 35], more robust [33, 39], or more scalable to large outdoor scenes [20, 32, 35]. However, the hand-crafted feature detectors and descriptors only work with well textured images. Recently, InLoc [40] pushed forward this direction by replacing the hand-crafted features by CNNs, i.e. NetVLAD [1] for image retrieve and VGG [37] 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.

2.2 Random Forests

Shotton *et al*. [36] proposed to regress the scene coordinates using a Random Forest, and this pipeline was extended in several following works. Guzman-Rivera *et al*. [13] trained a random forest to predict diverse scene coordinates to resolve scene ambiguities. Valentin *et al*. [42] 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, utiliz-

ing the increased predictive power of an auto-context random forest. None of these works are scene agnostic. A recent work [8] 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.

2.3 Convolutional Neural Networks

CNN based methods have brought major advancements in performance. PoseNet [15] 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 [14] or utilized the temporal dependency of videos to improve localization accuracy [44]. 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 [19, 3, 5] serving as 2D-to-3D correspondences, 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 representations from scenes rather than learns a set of scene-specific network weights. In this way, our method is scene agnostic and can be applied to unknown scenes without expensive retraining or adaptation.

2.4 Methods On or After the Publication of SANet

In the section, we briefly summary related works **on or after** the publication of SANet in ICCV 2019. There are mainly 2 directions: scene coordinate regression, and feature matching.

For scene coordinate regression, Brachmann *et al*. [6] presented Expert Sample Consensus (ESAC), an ensemble formulation of Differentiable Sample Consensus (DSAC), to achieve scene coordinate regression in large scale scenes. Li *et al*. [18] proposed a hierarchical classification + regression framework for scene coordinate prediction to improve the performance as well as deal with large scenes. Zhou *et al*. [47] designed a temporal localization method by extending scene coordinate regression to the time domain with kalman filtering. Despite the impressive results made on small scenes, these methods are still inferior to feature matching counterparts on large scale outdoor scenes, and also require scene specific training.

For feature matching, many works [31, 43, 11, 29, 23] were proposed to learn accurate and robust feature detectors and descriptors. Rocco *et al*. [30] also included the matching process into the learning framework. Cheng *et al*. [9] and Sarlin *et al*. [31] designed hand-crafted matching processes for better disambiguation and efficiency. Some works focused on other aspects of the problem, such as pose verification [41], privacy preserving [38], and scene compression [7]. In general, this category of methods achieve better performances than scene coordinate regression methods on large scale scenes, while a thorough comparison on small scenes has not been investigated.

Chapter 3

Method

The overview of our pipeline is illustrated in Figure 3.1. The inputs to our method are a set of scene images $\{\mathbf{I}_s\}_{all}$ with its associated 3D scene coordinates $\{\mathbf{X}_s\}_{all}$ 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 neural network to regress the scene coordinate map of the query image by interpolating the 3D coordinates associated with the retrieved scene images. 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 3.1). With two constructed pyramids, we design two modules: Query-Scene Registration (QSR) as well as Fusing, and apply them iteratively to regress the scene coordinate map in a coarse-to-fine manner (Section 3.2). 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 the query image can be estimated by RANSAC+PnP similar to [5] (Section 3.3). The training process of our model is introduced in Section 3.4.

3.1 Constructing Pyramids

3.1.1 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 points (i.e. scene 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 network. Specifically, we use the Dilated Residual Network (DRN38) [46] 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

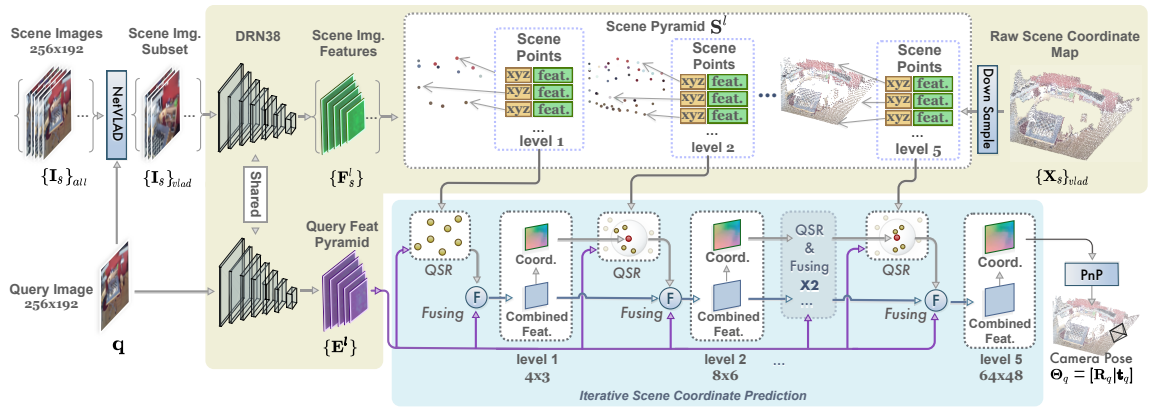

Figure 3.1: The overview of our pipeline. After the image retrieval (left most), we first construct the representations (i.e. 2 pyramids) of the scene and the query image (Yellow region. Section 3.1). Then, given the pyramids, we predict the scene coordinate map of the query image in a coarse-to-fine manner (Blue region. Section 3.2). Finally, the camera pose can be computed by RANSAC+PnP (Section 3.3).

feature maps $\{\mathbf{F}_s^l | l = 1, ..., 5\}$ from DRN38 with resolutions increasing from 4×3 to 64×48 by a factor of 2. The weights of the network are shared by all scene images.

Then, the scene coordinate map $\mathbf{X}_s$ is resized to match the resolution of feature maps at different levels. Here, we reduce the resolution of $\mathbf{X}_s$ by applying `Average Pooling` 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 $\{\mathbf{f}_i^l\}$. In 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 set of multi-scale point clouds equipped with image feature vectors, where m^l is the number of points at level l.

3.1.2 Query Feature Pyramid

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

3.2 Predicting Scene Coordinates

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 the computation, we take a coarse-to-fine strategy. The network first produces a coarse scene coordinate map at the resolution of 4×3 as a rough estimation, which is then upsampled and refined level-by-level to yield more detailed predictions.

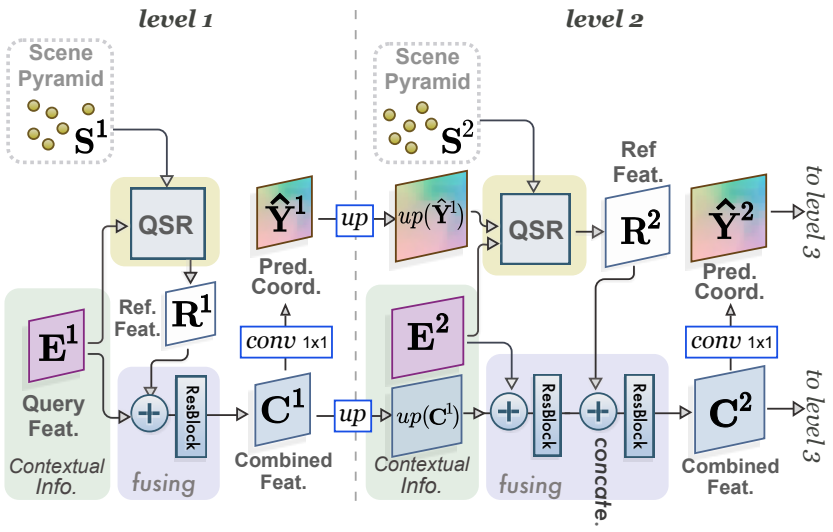

Figure 3.2: The coarse-to-fine scene coordinates prediction from level 1 to level 2. For level $l > 2$, the process is identical to level 2. We sequentially apply the pixel-wise operation, named as Query-Scene Registration (QSR), and the convolutional operation (i.e. cross-pixel), named as Fusing, to produce the predictions.

At each level, we sequentially apply two modules to predict the scene coordinate map: (1) The Query-Scene Registration (QSR) module, which is a pixel-wise operation that learns to register each pixel of the query image feature into the 3D scene space by interpolating the 3D coordinates of the scene pyramid based on visual similarity; (2) The Fusing module, which 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 coarse-to-fine scene coordinate map prediction. We will discuss the QSR module in detail later. This process is illustrated in Figure 3.2, where we only show the details of level 1 and 2 since the process of level $l > 2$ is identical to that of level 2.

At the initial level $l = 1$, we predict the coarsest scene coordinate map $\hat{\mathbf{Y}}^1$ from the first level pyramids $\mathbf{E}^1$ and $\mathbf{S}^1$. First, we register each pixel of the feature $\mathbf{E}^1$ to the 3D scene space by feeding $\mathbf{E}^1$ and $\mathbf{S}^1$ into the QSR module, which outputs the *scene reference feature* $\mathbf{R}^1$ that encodes the registration results. 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 it with the 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 the prediction $\hat{\mathbf{Y}}^{l-1}$ from the 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 the current level. The QSR module takes the query image feature $\mathbf{E}^l$, the l-th level scene pyramid $\mathbf{S}^l$ (i.e.

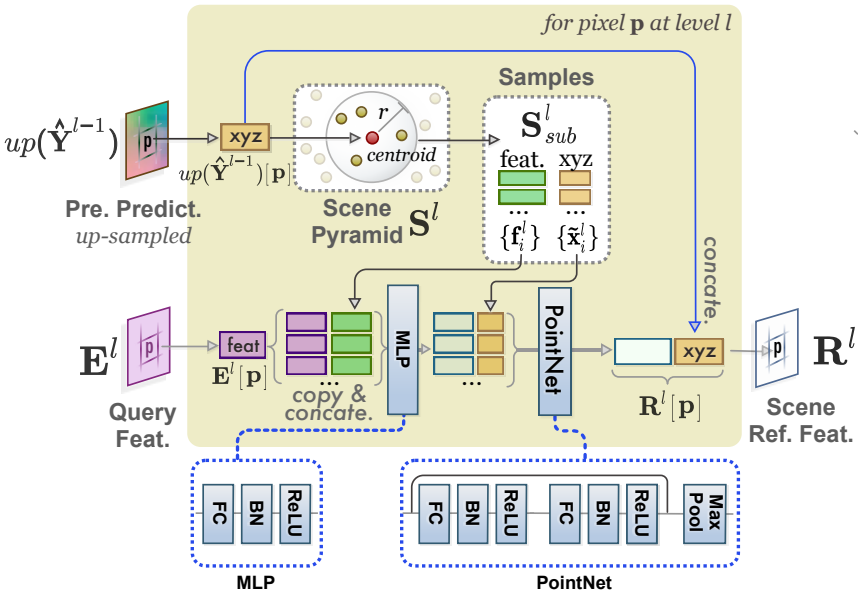

Figure 3.3: The Query-Scene Registration (QSR) at level l, which learns to register the feature vector $\mathbf{E}^l[\mathbf{p}]$ into the 3D scene space represented by $\mathbf{S}^l$ and produce the scene reference feature $\mathbf{R}^l[\mathbf{p}]$ that encodes the registration result. For the initial level $l = 1$, we have $\mathbf{S}^1_{sub} = \mathbf{S}^1$.

the 3D points with deep features), 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 the 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 combined feature $\mathbf{C}^l$ and the scene coordinate map prediction $\hat{\mathbf{Y}}^l$ of the current level. To reduce the memory cost and avoid over-smoothed results, we do not fuse the image context at the final level $l = 5$. Instead, we directly decode $\mathbf{R}^5$ by a 1×1 `Conv` to get the prediction.

3.2.1 Query-Scene Registration (QSR)

In QSR (see Figure 3.3), the query image feature $\mathbf{E}^l$ is processed pixel-wisely. 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 space of $\mathbf{S}^l$, and encode 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 3D points $\mathbf{S}^l_{sub}$ from the l-th level scene pyramid $\mathbf{S}^l$, denoted as $\mathbf{S}^l_{sub} = \{(\mathbf{f}^l_i, \tilde{\mathbf{x}}^l_i) | i = 1, ..., k^l\} \subseteq \mathbf{S}^l$, where k^l is the number of sampled points, $\mathbf{f}^l_i$ and $\tilde{\mathbf{x}}^l_i$ are feature vectors and scene coordinates as in Section 3.1.1; (2) Execute the registration through a PointNet [27]-inspired architecture with $\mathbf{E}^l[\mathbf{p}]$ and $\mathbf{S}^l_{sub}$ as inputs.

At the level l, to narrow down the solution space and let the network focus on local details, we sample a subset of 3D points $\mathbf{S}^l_{sub} \subseteq \mathbf{S}^l$ from the neighbourhood space of the previous level prediction $up(\hat{\mathbf{Y}}^{l-1})[\mathbf{p}]$, which is inspired by the `Sampling & Grouping` strategy in PointNet++ [28]. More specifically, we define a sphere region centered at $up(\hat{\mathbf{Y}}^{l-1})[\mathbf{p}]$ with radius r^l as the neighbourhood space, and randomly select 3D points of the l-th level scene pyramid $\mathbf{S}^l$ falling

in this sphere. When $l = 1$, we simply have $\mathbf{S}^1_{sub} = \mathbf{S}^1$, namely all points of $\mathbf{S}^1$ are used. During the sampling, we normalize the 3D scene space represented by $\mathbf{S}^l_{sub}$ through subtracting $up(\hat{\mathbf{Y}}^{l-1})[\mathbf{p}]$ from the 3D coordinates $\{\tilde{\mathbf{x}}^l_i\}$ for better generalization.

Given the sampled 3D points $\mathbf{S}^l_{sub}$ and the query image feature vector $\mathbf{E}^l[\mathbf{p}]$, we design a network based on PointNet [27] 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 feature vector $\mathbf{E}^l[\mathbf{p}]$ and the scene appearance feature $\mathbf{f}^l$ of each scene point in $\mathbf{S}^l_{sub}$. The resulting features are appended with the 3D coordinates $\tilde{\mathbf{x}}^l$ and fed into a PointNet [27] (without Spatial Transform) to produce the scene reference feature $\mathbf{R}^l[\mathbf{p}]$ that encodes the registration result. Note that we additionally append $up(\hat{\mathbf{Y}}^{l-1})[\mathbf{p}]$ to the feature vector output by the PointNet [27] to provide enough information for recovering the normalization of the 3D scene space represented by $\mathbf{S}^l_{sub}$.

3.3 Query Pose Estimation

Given the 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 [12, 17]. Since the scene coordinate map may contain outliers, we adopt a RANSAC+PnP pipeline similar to DSAC++ [5] for robust estimations. First, a collection of k 4-point tuples are randomly sampled from the predicted scene coordinates. Then, by solving the PnP problem for each 4-point tuple, we can obtain a set of hypothesis poses $\mathbf{H} = \{h_j | j = 1, ..., k\}$. We then find the best hypothesis h^* that is most coherent with the predicted scene coordinate map. In particular, each hypothesis is scored with counted inliers. Given a pose hypothesis h, the scoring function is defined as:

$$\xi(h) = \sum_{\mathbf{p}} sig(\beta(\gamma - \pi(h, \hat{\mathbf{Y}}[\mathbf{p}]))), \tag{3.1}$$

where the hyper-parameter β controls the softness of the sigmoid function $sig(\cdot)$, and the variable γ is a manually defined parameter indicating the inlier threshold. The function $\pi(h, \hat{\mathbf{Y}}[\mathbf{p}])$ defines the reprojection error at pixel $\mathbf{p}$ with 2D coordinates $\mathbf{x_p}$ under the hypothesis pose h as follows,

$$\pi(h, \hat{\mathbf{Y}}[\mathbf{p}]) = \left\| K h^{-1} \hat{\mathbf{Y}}[\mathbf{p}] - \mathbf{x_p} \right\|. \tag{3.2}$$

To obtain the final camera pose Θ_q , the hypothesis with the highest score is selected first,

$$h^* = \arg\max_h \xi(h).$$

Next, initialized by $\Theta_q = h^*$, the camera pose Θ_q is refined by iterating the following two steps until convergence or reaching the maximum number of iterations: (1) Selecting the inliers from all

scene coordinates of which the reprojection error is lower than the inlier threshold γ; (2) Optimizing the camera pose with PnP by involving all newly selected inliers.

3.4 Training

We train the neural networks for pyramid construction and scene coordinate prediction in a supervised manner with ground truth scene coordinate maps of query images. Here, we apply the L_2-norm loss [5] averaged over all pixels and pyramid levels as bellow,

$$L = \sum_{l} \sum_{\mathbf{p}} \mathbf{v}^l[\mathbf{p}] \left\| \mathbf{Y}^l[\mathbf{p}] - \hat{\mathbf{Y}}^l[\mathbf{p}] \right\|, \tag{3.3}$$

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 $\mathbf{v}^l[\mathbf{p}] = 0$. For a pixel $\mathbf{p}$ at level l, if its ground truth scene coordinate does not locate inside the sphere when sampling $\mathbf{S}^l_{sub}$ (Section 3.2.1), we discard the gradient of that pixel.

Chapter 4

Experiments

4.1 Experimental Setup

4.1.1 Datasets

We train and evaluate our method on both indoor and outdoor datasets, including *SUN3D* [45], *7Scenes* [36], and *Cambridge Landmarks* [15]. For indoor scenes, our method is trained and evaluated with *SUN3D* [45] and *7Scenes* [36] respectively, 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 further divided into multiple `train` and `test` sequences with a thousand of frames each. For outdoor scenes, the *Cambridge Landmarks* [15] is used for fine-tuning and evaluating our method in a cross-validation fashion. The dataset contains 6 different outdoor scenes with RGB frames, camera poses, and 3D models available. Each scene is also split into `train` and `test` sequences. To obtain ground truth scene coordinate maps, we use the rendered depths provided by [5], which are transformed with the camera poses given by the dataset.

4.1.2 Training Data

In order to train our model, we need to construct training samples beforehand. Each training sample consists of 5 scene images with their 3D scene coordinates for building the scene pyramid, and 1 query image with its ground truth scene coordinates for supervision.

For indoor scenes, our model is trained on *SUN3D*, from which we randomly select 388 video sequences for training. For each sequence, we randomly choose 10% of all frames as anchors. For each anchor frame, we sequentially collect 4 other frames in the sequence as scene images, 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, between the first and the last collected frames, 50 query images are randomly selected to form different training samples. In the end, we have more than 200K samples.

For outdoor scenes, *Cambridge Landmarks* is used for fine-tuning as well as testing. We fine-tune our model (pre-trained on *SUN3D*) using 5 out of 6 scenes from *Cambridge Landmarks*, and

test on the remaining one (i.e. 6 times of fine-tuning and testing). All frames in the training scenes are used as query images, and for each query we run NetVLAD [1] to retrieve 100 closest candidate frames. Among these 100 candidates, we randomly choose 5 frames as scene images, with relative translations less than 2 meters and relative rotations less than 45° to the query image. In this way, we obtain in total 15K-30K training samples for each time of fine-tuning.

4.1.3 Testing Data

Our model is evaluated on *7Scenes* and *Cambridge Landmarks* for indoor and outdoor scenarios. As mentioned in Section 4.1.1, the data of each scene in these 2 datasets is divided into `train` and `test` sequences. For each test scene, we use frames from the `train` sequences as scene images (i.e. To perform image retrieval and construct the scene pyramid as described in Chapter 3) and use the frames in the `test` sequences as queries to fairly compare with previous methods.

4.1.4 Implementation Details

Training: Our network is implemented with PyTorch [26], and trained on GTX1080Ti with 11G memory. When training our model, we set the sphere radius to $r = [1.5m, 0.75m, 0.5m, 0.25m]$ for sampling 3D points (Section 3.2.1) 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 (use y axis for convenience) to scene coordinates, and brightness jittering to scene RGB images. The optimizer Adam [16] with initial learning rate of 2.0×10^{-4} is applied, and the batch size is set to 6.

Testing: Limited by the 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 sphere radius to $r = [0.75m, 0.5m, 0.25m, 0.125m]$ 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. For pose estimation (Section 3.3), we sample 128 hypotheses from the predicted scene coordinate map. We set hyper-parameter $\beta = 4.0$, the inlier threshold $\gamma = 0.5$ for outdoor scenes and $\gamma = 0.75$ for indoor scenes respectively. In terms of pose refinement, the maximum number of iterations is set to 100.

4.1.5 Comparisons

We compare estimated camera poses, method efficiency, and predicted scene geometry against various scene-agnostic and scene-specific methods. Active Search [35] and InLoc [40] are scene ag-

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

Steps	**Ours**	**DSAC++** [5]	**InLoc** [40]
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

Table 4.1: Time of each step w.r.t 7000 scene images.

Steps	**500 imgs.**	**1000**	**2000**	**5000**	**7000**
Index. VLAD feat. (all scene imgs.)	23s	50s	128s	263s	427s
Retrieval (per query)	16ms	27ms	61.8ms	123ms	171ms
Estimate Pose (per query)	0.37s	0.37s	0.37s	0.37s	0.37s
Total (per query)	0.39s	0.40s	0.43s	0.49s	0.54s

Table 4.2: Time of each step w.r.t number of scene images.

nostic, which utilize traditional feature matching pipelines, with hand-crafted or pre-trained deep features respectively, for matching query image patches to the 3D points. Random Forest based approaches [24, 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 [15] that trained with 3D geometry loss is also compared.

4.2 Efficiency

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

Time Costs: Table 4.1 lists the time of each step w.r.t 7000 scene images in *7Scenes*, with comparisons to DSAC++ [5] and InLoc [40]. Taking the ORB-SLAM[25] as an example, it creates a new keyframe for approximately every 0.7s for TUM RGB-D SLAM indoor sequences. Our method and InLoc [40] 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 [40] takes several seconds based on its public implementation (CPU version, using 6 threads in parallel).

We also investigate how the running time of our pipeline scales w.r.t number of scene images. As shown in Table 4.2, the time of indexing all scene images and image retrieval increases linearly, while the pose estimation takes a constant time as we only retrieve a fixed number (i.e. 10) of scene images.

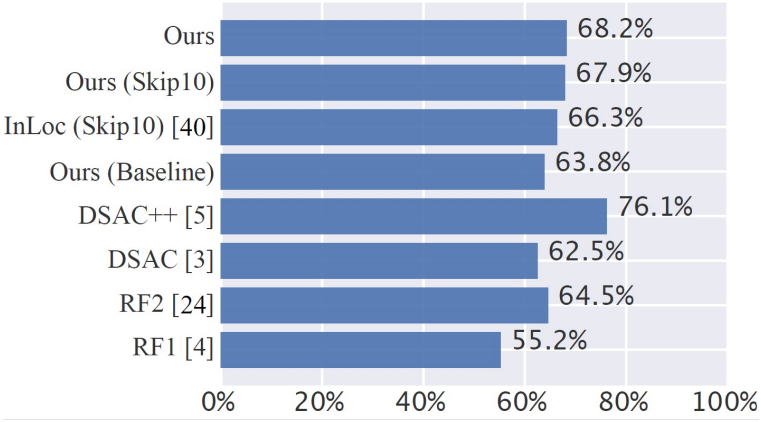

Figure 4.1: Percentage of predicted camera poses falling within the error threshold of $(5^\circ, 5cm)$ on *7Scenes* indoor dataset by `RF1` [4], `RF2` [24], `DSAC++` [5], `DSAC` [3], `InLoc(Skip10)` [40], and our approaches.

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 [40] 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.

4.3 Localization Accuracy

We first measure localization accuracy in terms of the percentage of predicted poses falling within the error threshold of $(5^\circ, 5cm)$. Figure 4.1 shows comparisons with other methods on *7Scenes*. Our method outperforms the Random Forest based methods [24, 4], and DSAC [3]. Since the inference of InLoc [40] 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 *Ours (Skip10)* outperforms *InLoc (Skip10)* by 1.6%. Note that 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 4.5.

Secondly, localization accuracy is measured in terms of the median translation and rotation errors. Comparisons on the *7Scenes* indoor dataset are listed in the top half of Table 4.3, where reported numbers of other methods are cited from the original papers. Comparing with scene specific methods, our approach outperforms PoseNet [14] on most scenes, while is slightly inferior to the state-of-the-art method DSAC/DSAC++ [3, 5]. Comparing with scene agnostic methods, we outper-

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

	Scene Specific						Scene Agnostic					
	PoseNet (Geo) [14]		DSAC [3]		DSAC++ [5]		Active Search [35]		InLoc [40]		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

Table 4.3: Indoor and Outdoor localization accuracy on *7Scenes* and *Cambridge*.

form Active Search [35] on most scene except for `Stairs`, as our model is trained with the *SUN3D* indoor dataset that is lack of such extreme repetitive patterns. Our method yields comparable overall results to the InLoc [40] algorithm, while slightly better on `Chess`, `Office`, and `Pumpkin`.

Table 4.3 also shows the localization accuracy on outdoor scenes. Comparing with scene specific models, our performance is inferior to DSAC [3] and DSAC++ [5] while still better than PoseNet [15]. Comparing with scene agnostic methods, our method yields comparable results with Active Search [35], and slightly better results than InLoc [40] on 4 scenes, 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.

4.4 Scene Coordinate Accuracy

We compare the accuracy of scene coordinate maps against InLoc [40] and DSAC/DSAC++ [3, 5] on the *7Scenes* dataset. Figure 4.2 shows the cumulative distribution function (CDF) of errors of the estimated scene coordinates. 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 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 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 [40] removes invalid matchings by consistency check and produces the highest precision over remaining pixels as shown by *InLoc*

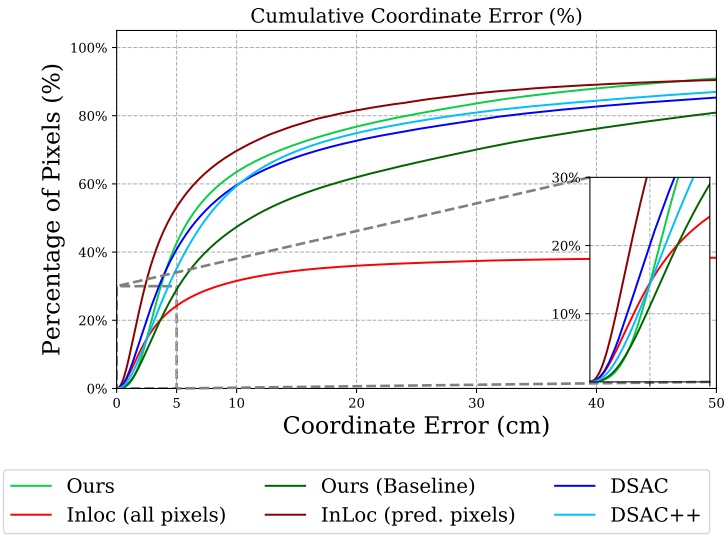

Figure 4.2: Cumulative Distribution Function of scene coordinate errors compared with InLoc [40], DSAC [3] and DSAC++ [5] on *7Scenes*.

(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 4.3 visualizes several predicted scene coordinate maps as color coded coordinates as well as triangle meshes. In constrast to InLoc [40], our method produces dense predictions and is robust at featureless areas, because our network exploit global context in the query image when predicting the scene coordinate maps. For example, InLoc [40] cannot produce reasonable results on the wall and the ceiling, while our method can. The dense prediction enables potential applications other than localization, such as robotic obstacle avoidance.

4.5 Detailed Analysis

4.5.1 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 4.1 and that of the scene coordinate map in Figure 4.2, both are denoted as *Ours (Baseline)*. It is clear that our proposed method surpasses this baseline approach with unnegligible margins on both evaluation metrics.

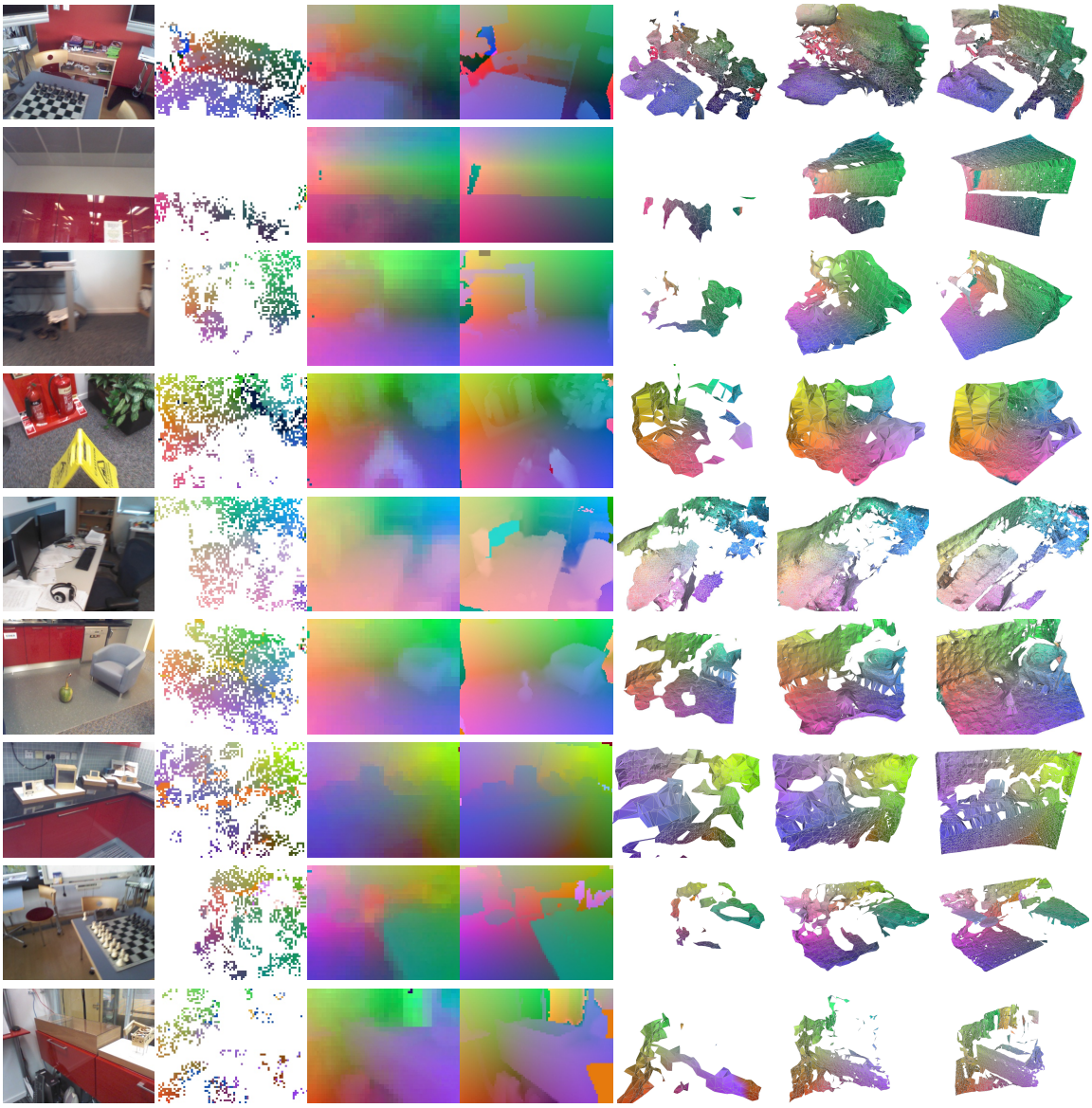

Figure 4.3: Scene coordinate map comparison with InLoc [40] and the ground truth (G.T.) on *7Scenes*, the scene coordinates `xyz` are encoded in `rgb` channels for visualization. The last three columns show the geometry (Geo.) comparison by reconstructing the mesh from the scene coordinate map.

4.5.2 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 correspondence points.* We randomly select a pixel `Que` from the query image feature map, whose location is marked by the green dot in Figure 4.4 (a). Its ground truth correspondence point `Pos` in the scene image feature map is marked by a red dot in Figure 4.4 (b), while `Neg` is a random irrelevant point marked by the blue dot. Now,

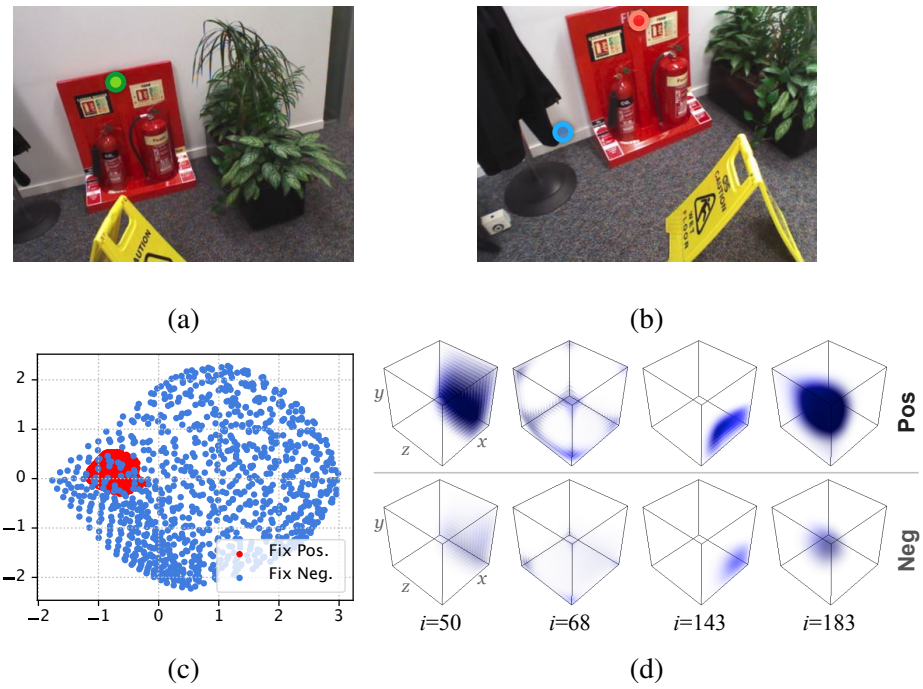

Figure 4.4: (a) A pixel Que in the query image is marked in Green. (b) The ground truth correspondence point Pos in a scene reference image is marked in Red, while a randomly selected point Neg is marked in Blue. (c) Two sets of scene reference features from *Experiment-1* are projected in 2D space by 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 dark blue stands for high channel activation. Please refer to the main text for details.

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 the experiment, and each feature pixel has 256 channels.

We add noise to the xyz coordinates of the Pos or Neg point 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 4.4 (c) 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 correspondence 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 correspondence points.

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

	K. College	O. Hospital	S. Facade	S.M Church
w/o F.T.	0.76° 0.39m	0.47° 0.34m	0.56° 0.15m	0.84° 0.23m
w/ F.T.	0.54° 0.32m	0.53° 0.32m	0.47° 0.10m	0.57° 0.16m

Table 4.4: Pose median errors w/ or w/o fine-tuning (F.T.) on outdoor scenes.

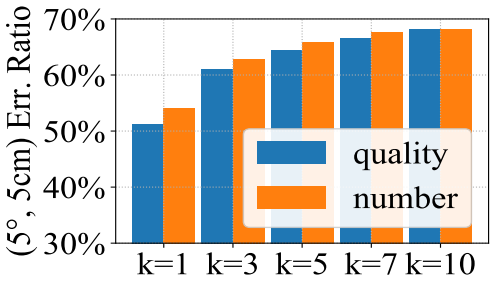

Figure 4.5: Pose accuracy w.r.t Retrieval **quality** and **Number** of images in the scene pyramid.

a set of positions $\mathbf{X}$ within the $1m \times 1m \times 1m$ cube centered at $\texttt{Pos}$. Given the point $\texttt{Pos}$ at position $\mathbf{x}$ and the query pixel $\texttt{Que}$ as inputs, 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 4.4 (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 68-th channel has high response on eight corners of the cube. We run the same experiment for the $\texttt{Neg}$ sample, and plot the same four channels in the bottom row of Figure 4.4 (d). Comparing with the positive sample $\texttt{Pos}$, it shows similar activation patterns but with different magnitude. This further verifies that our scene reference feature encodes spatial information.

4.5.3 Fine-tuning on Outdoor Dataset

We investigate the effect of fine-tuning on outdoor scenes in this section. The fine-tuning is necessary due to the distribution gap between indoor and outdoor images. As shown in Table 4.4, without fine-tuning (F.T.), the performance degrades gently on 4 outdoor scenes. In principle, if sufficient outdoor data is given, a common indoor/outdoor model could be trained.

4.5.4 Reliance on Retrieval Quality and Number of Images in the Scene Pyramid

To quantify the influence of retrieval **quality**, we simulate different levels of retrieval quality by keeping the top k retrieval results and replace the rest of the 10 retrieved images with least scored ones. To test the effect of the **number** of images in the scene pyramid, we use the top k retrieved images to construct the scene pyramid. The performances are measured by the ratio of pose errors that are less than ($5°$, 5cm) on *7Scenes*. As shown in Figure 4.5, even with the worst retrieval quality or only 1 image in the scene pyramid, our method could still localize more than 50% queries.

Chapter 5

Conclusion

This thesis presents a scene agnostic neural 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 model learns to encode the scene environment into a hierarchical representation, and predict the scene coordinate map of a query image based on the scene representation. 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 method is validated on both indoor and outdoor datasets, and achieves state-of-the-art performance among scene agnostic methods.

Bibliography

[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] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera relocalization. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 7525–7534, 2019.

[7] Federico Camposeco, Andrea Cohen, Marc Pollefeys, and Torsten Sattler. Hybrid scene compression for visual localization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 7653–7662, 2019.

[8] 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.

[9] Wentao Cheng, Weisi Lin, Kan Chen, and Xinfeng Zhang. Cascaded parallel filtering for memory-efficient image-based localization. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 1032–1041, 2019.

[10] 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.

[11] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 8092–8101, 2019.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 11983–11992, 2020.

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

[20] 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.

[21] 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.

[22] 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.

[23] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan. Aslfeat: Learning local features of accurate shape and localization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 6589–6598, 2020.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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 (NeurIPS)*, pages 5099–5108, 2017.

[29] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[30] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1651–1662, 2018.

[31] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 12716–12725, 2019.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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

[38] Pablo Speciale, Johannes L Schonberger, Sing Bing Kang, Sudipta N Sinha, and Marc Pollefeys. Privacy preserving image-based localization. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 5493–5503, 2019.

[39] 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.

[40] 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.

[41] Hajime Taira, Ignacio Rocco, Jiri Sedlar, Masatoshi Okutomi, Josef Sivic, Tomas Pajdla, Torsten Sattler, and Akihiko Torii. Is this the right place? geometric-semantic pose verification for indoor visual localization. In *Proc. of Internatoinal Conference on Computer Vision (ICCV)*, pages 4373–4383, 2019.

[42] 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.

[43] Lukas von Stumberg, Patrick Wenzel, Qadeer Khan, and Daniel Cremers. Gn-net: The gauss-newton loss for multi-weather relocalization. *IEEE Robotics and Automation Letters*, 5(2):890–897, 2020.

[44] 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.

[45] 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.

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

[47] Lei Zhou, Zixin Luo, Tianwei Shen, Jiahui Zhang, Mingmin Zhen, Yao Yao, Tian Fang, and Long Quan. Kfnet: Learning temporal camera relocalization using kalman filtering. In *Proc. of Computer Vision and Pattern Recognition (CVPR)*, pages 4919–4928, 2020.