ISPRS Journal of Photogrammetry and Remote Sensing 189 (2022) 50–62

Contents lists available at ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

FeatLoc: Absolute pose regressor for indoor 2D sparse features with simplistic view synthesizing

Thuan Bui Bach [a], Tuan Tran Dinh [b], Joo-Ho Lee [b,*]

[a] Graduate School of Information Science and Engineering, Ritsumeikan University, Japan
[b] College of Information Science and Engineering, Ritsumeikan University, Japan

ARTICLE INFO

Keywords:
Visual localization
Sparse features
Absolute pose regression

ABSTRACT

Precise localization using visual sensors is a fundamental requirement in many applications, including robotics, augmented reality, and autonomous systems. Traditionally, the localization problem has been tackled by leveraging 3D-geometry registering approaches. Recently, end-to-end regressor strategies using deep convolutional neural networks have achieved impressive performance, but they do not achieve the same performance as 3D structure-based methods. To some extent, this problem has been tackled by leveraging the beneficial properties of sequential images or geometric constraints. However, these approaches can only achieve a slight improvement. In this work, we address this problem for indoor scenarios, and we argue that regressing the camera pose using sparse feature descriptors could significantly improve the pose regressor performance compared with deep single-feature-vector representation. We propose a novel approach that can directly consume sparse feature descriptors to regress the camera pose effectively. More importantly, we propose a simplistic data augmentation procedure to exploit the sparse descriptors of unseen poses, leading to a remarkable enhancement in the generalization performance. Lastly, we present an extensive evaluation of our method on publicly available indoor datasets. Our FeatLoc achieves 22% and 40% improvements in translation errors on 7-Scenes and 12-Scenes relatively, compared with recent state-of-the-art absolute pose regression-based approaches. Our codes are released at https://github.com/ais-lab/FeatLoc.

1. Introduction

Precise localization techniques play an essential role in many real-world applications, such as intelligent systems, augmented reality applications (Häne et al., 2017; Lim et al., 2015; Castle, 2008), and autonomous systems. Their algorithms enable us to accurately compute the position and orientation of a given image in a known scene. Popular approaches use advanced hardware, such as LIDAR sensors, GPS, WIFI, or Bluetooth. However, such wireless hardware sensors suffer from GPS-denied environments, such as bad weather conditions, blocked areas, and especially indoor environments. Another common solution is the use of inexpensive visual sensors. However, robust visual localization based on a single image is still challenging, especially in end-to-end learning approaches.

Camera-based localization has been widely tackled by exploiting 2D-3D matching between a query 2D image and a given 3D map (Sattler, 2016; Brachmann, 2017; Brachmann and Rother, 2018; Meng et al., 2017). Traditionally, these approaches first establish 2D-3D matches based on matching descriptors associated with both test images and a 3D map. The algorithms then estimate the camera pose by applying an n-point-pose solver (Albl et al., 2015; Kneip et al., 2011) inside a RANSAC (Chum and Matas, 2008). Alternatively, the 3D point positions can be directly predicted by leveraging state-of-the-art machine learning regressors (Sattler, 2016; Brachmann, 2017; Meng et al., 2017; Meng et al., 2018).

In recent years, methods based on deep convolutional neural networks (DNNs) have become popular for absolute pose regression (APR) (Kendall, 2015; Walch et al., 2017; Brahmbhatt, 2018; Wang et al., 2020a,b; Purkait et al., 2018; Zhou et al., 2021). Instead of storing a large 3D map in memory or just using machine learning only for some parts of the localization pipeline (Sattler, 2016), these approaches aim to learn the whole pipeline of the localization task. Given a training image set and corresponding poses, APR approaches train DNNs to automatically extract features and directly recover camera poses from single images. The general advantages of APR approaches are the benefits of end-to-end training, where the camera pose can be computed directly

* Corresponding author.
E-mail address: leejooho@is.ritsumei.ac.jp (J.-H. Lee).

https://doi.org/10.1016/j.isprsjprs.2022.04.021
Received 13 December 2021; Received in revised form 21 April 2022; Accepted 22 April 2022
Available online 11 May 2022

from images in real-time (less than 50 ms for pose estimation per image). Another advantage is low memory requirement (≈ 100 MB) for the network weight instead of several GB for reference 3D cloud map and training samples. The final re-localization system, therefore, is very scalable, compared to pure geometric based method (Kendall, 2015). In addition, the performance speed and the usage of memory are independent of number of training samples, while metric localization scales O(n) with training data size (Wu, 2013). Although APR techniques are computationally efficient, they are still significantly less accurate than conventional geometry-based methods (Sattler et al., 2019; Purkait et al., 2018), and they are restricted from generalizing beyond the limitation of the training data.

In this work, we argue that regressing the camera pose using sparse feature descriptors can significantly improve the localization performance and allow generalization beyond the limitation of the training data compared with using dense feature representation approaches (Kendall, 2015; Wang et al., 2020a,b; Brahmbhatt, 2018). We propose **FeatLoc**, a direct end-to-end camera pose regressor from 2D sparse **feat**ure descriptors for indoor **loc**alization. Unlike previous methods, which rely on pre-trained deep feature extractors (Kendall, 2015; Walch et al., 2017; Wang et al., 2020a,b), our proposed FeatLoc consumes sparse descriptors to regress the global map, which is advantageous for several reasons.

1. Learning from sparse features could make it easier for the DNN model to understand the underlying geometric concepts of the environments, while previous direct image-based approaches have no built-in reasoning about geometry (Kendall, 2015; Wang et al., 2020a,b; Brahmbhatt, 2018) during the training phase. The results thus are poor of ability to explore unseen poses that are different from training set (Sattler et al., 2019).
2. The usage of sparse features allows the training data to be augmented with unseen poses, leading to substantial improvements in the generalization performance.
3. Previous RGB-based methods (Kendall, 2015; Brahmbhatt, 2018; Wang et al., 2020a,b) spend a lot of unnecessary computation (and trainable parameters) on dense feature-extraction, which seems not necessary for this task of re-localization. In light of empirical evidence, pure geometry methods which rely only on 2D sparse features (a gold-standard such as SIFT (Lowe, 2004)) in general have better accuracy in pose estimation than current CNN-based approaches (Sattler et al., 2019). Hence, the networks used in our approach is significantly small and be able to train from scratch without a need of leveraging pre-trained classification networks.
4. More importantly, in the later experiment results, we shown that our networks learned on 2D sparse feature are more robust to condition changing compared to recent CNN-based approaches (Kendall, 2017; Brahmbhatt, 2018).

In addition, our proposed architecture is developed mainly based on the PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) structures. Thus, it inherits the capability of learning the local context and multi-scale feature combination, which leads to an improvement in this task. Unlike SPPNet (Purkait et al., 2018), which must randomly regulate the sparse descriptors into a fixed 2D grid before applying the convolutional network, our proposed architecture can directly input the entire features descriptors to produce camera poses without losing valuable information. Fig. 1 illustrates the superior performance of our FeatLoc approach on an indoor chess scene from the 7-Scenes (Shotton et al., 2013) dataset.

2. Related work

2.1. Structure-based localization

Geometry-based localization of camera pose with respect to a 3D environment has been well studied in the last decade (Li et al., 2010; Sattler, 2016; Sattler, 2011; Li, 2012). Specifically, these approaches rely on 2D-3D matches between 2D pixel positions and a pre-defined 3D point map for pose estimation. The matches can be established by matching the descriptors (Donoser, 2014; Li, 2012; Sarlin, 2020; Sattler, 2016) before applying the n-point algorithm (Albl et al., 2015; Kneip et al., 2011) to estimate the camera poses. In the case that the underlying 3D map is enormous, the computational cost can be greatly reduced by adding bag-of-features-based steps to quickly identify relevant subsets of the cloud points (Zhang & Kosecka, 2006). Recent efforts have addressed this problem by regressing 3D coordinates from pixel patches (Brachmann, 2017; Brachmann et al., 2016; Brachmann and Rother, 2018) and have achieved impressive pose accuracy at a small scale. Nevertheless, they ultimately cover only a tiny fraction of the world.

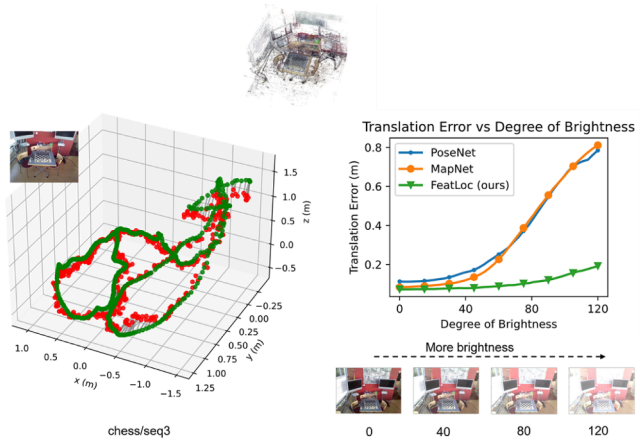

Fig. 1. Camera localization results for an indoor scene (left figure, green represents ground truth and red represents the prediction) from 7-Scenes (Shotton et al., 2013) dataset and comparison of localization errors under the effect of brightness. We directly feed sparse features from a single image to a neural network for predicting the 6-degrees-of-freedom (DoF) pose without any need for pre-processing. We leverage the 3D cloud map to generate unlimited synthetic sparse training data to boost the relocalization performance of FeatLoc. The estimation results of FeatLoc are more robust under condition changing such as brightness (figure on the right) when compared with recent state-of-the-art methods (such as PoseNet (Kendall, 2015, 2017) and MapNet (Brahmbhatt, 2018)). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.2. Deep neural networks for camera localization

Recent works have proposed learning-based methods to estimate absolute camera poses of an input image (Brahmbhatt, 2018; Kendall, 2016, 2017, 2015; Melekhov et al., 2017; Naseer and Burgard, 2017; Wang et al., 2020a,b; Zhou et al., 2021), thus implicitly representing the entire scene by the weights of the network. They all follow the same pipelines of a feature extractor comprising with a fully connected regressor (Huang et al., 2019; Kendall, 2015). The key to these methods lies in the image vector representation as the embedded vector, which is extracted using a base network, such as VGG (Simonyan & Zisserman, 2014) or ResNet (He et al., 2016). These methods mainly differ in the underlying base network, or the loss function used to train the network. PoseNet (Kendall, 2015) was probably the first to adopt DNNs to learn the absolute camera pose from an input image. This approach was then extended by leveraging long short-term memory (LSTM) (Walch et al., 2017) to structure embedded feature vectors, leading to a significant improvement. Later, localization performance was improved by using geometric reprojection errors (Kendall, 2017) or adding visual odometry constraints (Radwan et al., 2018; Valada et al., 2018). The authors of (Brahmbhatt, 2018; Radwan et al., 2018) proposed methods to localize camera poses based on sequential images, and they showed that the

relocalization performance can be significantly improved when learning on these approaches. Recent efforts additionally leverage the self-attention module (Wang et al., 2020a,b) to automatically guide the feature extractor to focus on the static region of the input image, yielding superior global camera pose regression performance, or introduce a novel framework to effectively fuse multiple modalities, such as image and depth (Zhou et al., 2021). However, the aforementioned approaches are still less accurate than structure-based methods (Sattler et al., 2019). This is probably because of the difficulty in geometric reasoning or generalizing beyond the limitation of the training data.

2.3. Feature extractors for deep camera pose regressor

Training a deep neural network from scratch for camera pose regression would be impractical due to the extensive training set requirements (Walch et al., 2017). In addition, it seems impossible to collect such a massive dataset because each camera pose label is covered by at least one training sample, where the output of the regression task is continuous and infinite. Therefore, most previous APR approaches (Kendall, 2015; Wang et al., 2020a,b; Brahmbhatt, 2018; Kendall, 2016, 2017; Sattler et al., 2019) utilize a pre-trained classification network (VGG (Simonyan & Zisserman, 2014) or ResNet (He et al., 2016)) as the feature extractor for this task. It will then output a high-dimensional feature vector, which can be seen as a feature that represents the image to be localized. However, this procedure of camera pose regression struggles to generalize beyond the limitation of the training data. In contrast, utilizing feature extractors under sparse descriptors can solve this problem by augmenting unlimited additional training data of the unseen poses by simply projecting 3D-2D of local features to image plane (Sattler et al., 2017; Hyeon et al., 2021). In this way, SPPnet (Purkait et al., 2018) can achieve a superior relocalization performance compared with previous APR methods.

3. Proposed approach

6-DoF camera pose estimation utilizing geometry-based methods can still achieve superior performance compared with recent efforts towards end-to-end learning strategies (Brachmann and Rother, 2018; Meng et al., 2017; Walch et al., 2017). Inspired by that, our proposed approach relies on sparse features, as in geometry-based approaches, to regress the global camera pose from an input image. Instead of matching sparse key points to compute the global camera poses, we propose a lightweight regressor architecture to learn the global map from sparse feature descriptors. Our FeatLoc architecture can directly input the sparse features to estimate the absolute camera pose efficiently. In addition, it can automatically focus on geometrically robust features and adapt efficiently to simplistic synthetic data. The proposed architecture is notably compact while achieving state-of-the-art performance, as described in detail in Section 3.2. In Section 3.3, we present a simplistic data augmentation procedure for exploiting the sparse descriptors of unseen poses to enrich the FeatLoc performance. Finally, in Section 3.4, we compare our approach with some related works to show the improvements of our framework that can lead to state-of-the-art performance on indoor scenes.

3.1. Problem statement

We design a lightweight deep neural architecture to regress the camera pose from spare feature sets. For each timestamp t, the agent receives a sparse feature set $\mathscr{P}_t = \{\mathbf{d}_i, \mathbf{k}_i | i = 1, .., N\}$ from image $\mathscr{I}_t$, where $\mathbf{d}_i$ is a description vector, and $\mathbf{k}_i$ is the vector of its coordinate (x, y) in the 2D pixel space. For simplicity, to design the network, we chose the descriptor dimension $\mathbf{d}_i \in \mathbb{R}^{256}$ for all experiments. However, our designed network can work well under an arbitrary number of input descriptors and automatically focus on robust features. In contrast, other approaches, such as SPPNet, must randomly regulate the set of features on a fixed 2D regular grid before feeding them to a deep convolutional architecture. For the output of the regression task, our model will output a camera pose, which is parameterized by a 6-DoF pose $[\mathbf{t}, \mathbf{r}]^T$, where $\mathbf{t}$ is a translation vector $\mathbf{t} \in \mathbb{R}^3$ and $\mathbf{r}$ is a quaternion-based orientation vector $\mathbf{r} \in \mathbb{R}^4$.

Given a feature set $\mathscr{P}_t$ extracted from image $\mathscr{I}_t$, our goal is not only to estimate the global camera pose $[\mathbf{r}, \mathbf{t}]$ but also to understand of the change of camera pose according to the change of feature set positions. To achieve this, we assume that the descriptors remain invariant to a slight change of viewpoint. Fig. 2 shows an example of sparse feature positions changed under a near viewpoint. Suppose that $\mathscr{P}_t^1, \mathscr{P}_t^2, ..., \mathscr{P}_t^M$ are the augmented feature sets generated based on reference feature set $\mathscr{P}_t^0$ and its visible 3D cloud points. To this end, with additional training data, our deep 6-DoF pose regressor $\widetilde{\mathfrak{F}}(\mathscr{P}) = (\mathbf{r}, \mathbf{t})^T$ can not only learn to avoid overfitting but can also generalize the environment well. The function $\widetilde{\mathfrak{F}}$ here is usually a DNN.

3.2. Base architecture of FeatLoc

This section introduces the proposed architecture of FeatLoc, a DNN architecture that directly consumes sparse feature key points to learn a global map through a monocular image set. Instead of learning directly from RGB images, our network learns from sparse features. Hence, the proposed architecture has the advantage of possible data augmentation. The overall architecture of FeatLoc is illustrated in Fig. 3. The FeatLoc model consists of an input layer, feature joining, and a regressor layer. In the input layer, the extracted sparse feature set $\mathscr{P}_t$ is down-sampled to a fixed N samples. The whole FeatLoc architecture is mainly based on PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b).

3.2.1. Feature Pre-processing: input layer

The task of training a DNN regressor for a 6-DoF camera pose from scratch is impractical because the output in this regression task is continuous and infinite. As a solution, previous works have leveraged a pre-trained classification network, such as GoogLeNet (Szegedy et al., 2015) or ResNet (Brahmbhatt, 2018), to extract the image representation feature vector before localizing the camera pose. While this procedure has shown some success in APR tasks, it is nevertheless limited in its achievable accuracy. Therefore, it is questionable whether these feature extractors are the best choice for the pose regression task. Human beings utilize key points to localize their position. In addition, classical geometric methods can precisely measure the query image's location by exploring the key points of the image. Inspired by this, our work uses the feature key points of images as representatives to be localized.

Here, we utilize the SuperPoint architecture (DeTone, 2018) as a base key point feature extractor. We select the SuperPoint model for the following reasons: (1) it is one of the most powerful deep feature

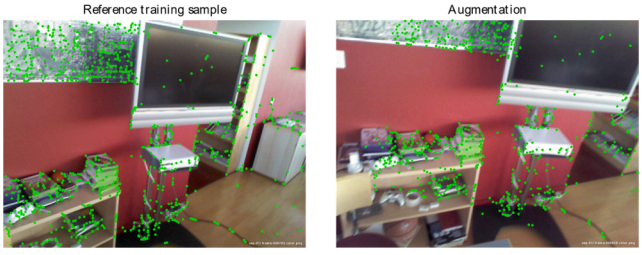

Fig. 2. An example of synthesized sparse descriptors (right) when changing the viewpoints for the challenging 7-Scenes dataset (Shotton et al., 2013). We leverage the 3D cloud map generated from training image samples to synthesize unseen viewpoints. The augmentation result on the right was generated by reprojecting the features from the reference image on the left.

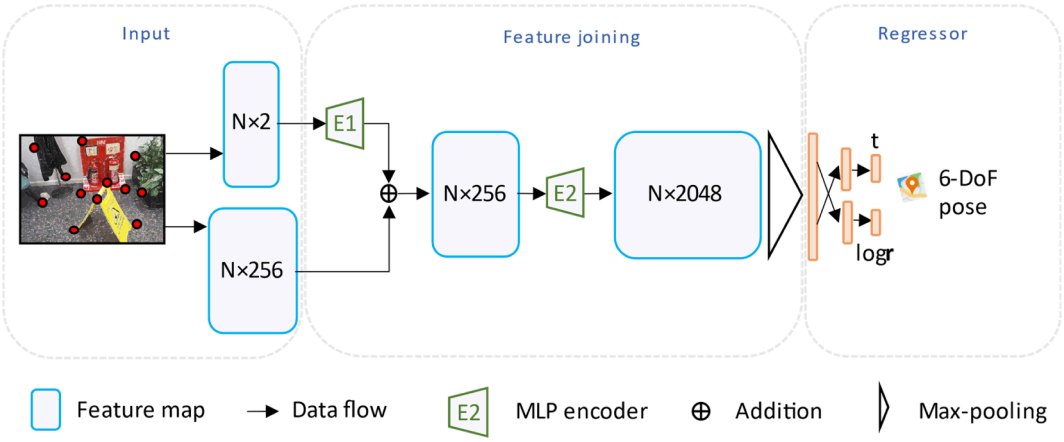

Fig. 3. Architecture of FeatLoc. It consists of an input layer, feature joining, and a pose regressor. The input layer is composed of a feature extractor (can be either a traditional or deep feature extractor, such as SIFT (Juan and Gwun, 2009) or SuperPoint (Sarlin, 2020). The extracted features are sent to the feature joining layer for registering feature positions and their descriptor into a united feature vector. Finally, the regressor layer regresses the 6-DoF camera poses using the united feature vector as the input.

extractors compared in terms of localization error and repeatability (Luo et al., 2020; Sarlin, 2020). (2) SuperPoint is available to be implemented for generating 3D cloud maps in hloc toolbox (Sarlin et al., 2019), which gives us a straightforward way to produce additional sparse descriptor data as the central part of this work.

3.2.2. Feature joining layer

Feature representation plays a crucial role in achieving accurate relocalization (Kendall, 2015; Walch et al., 2017). Here, each key point has two essential factors: the descriptor and key point position. This section discusses our feature joining layer, where the key point position and its descriptor will jointly learn to output a united independent vector. The obtained feature vectors are then extended to a higher dimensional space before applying the max-pooling layer to obtain the final feature to be localized.

Existing literature (Qi et al., 2017a; Zheng et al., 2019) has proved that the combination of feature extension and max-pooling layers, such as in PointNet, can identify critical points. Therefore, this is a desirable choice for our regression task. Initially, the 2D key point locations $\mathbf{k}$ will be encoded into 256 dimensions, the same as the descriptor dimension, using multi-layer perceptron (MLP) before combining with its descriptor. This encoder is illustrated as the E1 block in Fig. 3. The whole encoding process can be described as.

$$\mathbf{x}_i = \mathbf{d}_i + \mathbf{MPL}(\mathbf{k}_i) \tag{1}$$

The encoding of key point positions will later help the model gain more information about the camera pose. It also shows a distinct feature between two near-camera poses. The second part of the feature joining layer is developed mainly based on the PointNet structure.

In particular, PointNet exploits the MLP layer, max-pooling layer, and feature transformation to approximate a permutation-invariant function for point cloud data. Here, we only inherit the combination between the max-pooling and multi-perceptron function for our localization task. In general, it is a universal continuous set function approximator applied on orderless sets. The idea can be described as.

$$f(\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}) \approx g(h(\boldsymbol{x}_1), \cdots, h(\boldsymbol{x}_n)) \tag{2}$$

where $f : 2^{R^N} \rightarrow R$, $h : R^N \rightarrow R^K$, and $g : \underbrace{R^K \times \cdots R^K}_{n} \rightarrow R$ is a symmetric function.

This technique is a relatively simple but efficient way to capture the different properties of the input set. In particular, h is approximated by MLP, and g is a symmetric function, which is the max-pooling function in this case. h is the encoder block E2 of our proposed architecture in Fig. 3.

In addition, our later experiments show that replacing the feature encoder with a multi-scale grouping set abstraction layer in the PointNet++ architecture (Qi et al., 2017b) can lead to a substantial improvement in terms of learning with additional synthetic data Fig. 4 shows the proposed FeatLoc++ architecture, where the MSG-SA layer replaces the combination of encoder E1 and descriptor. Here, the SA layer is composed of a sampling layer, grouping layer, and PointNet layer (Qi et al., 2017a). The SA layer takes a feature matrix $\mathbf{F} \in \mathbb{R}^{N \times C}$ as input and outputs a new feature matrix $\mathbf{F}^{'} \in \mathbb{R}^{N^{'} \times C^{'}}$, where N and $N^{'}$ are the numbers of input and the output sparse features, and C and $C^{'}$ are the input and output feature dimensions, respectively. The multi-scale grouping strategy adopts farthest point sampling to sample $N^{'}$ regions with each region center as x_j; then, for each region (defined by a neighborhood of radius r), it utilizes the following symmetric function to extract its local feature (Liu et al., 2019):

$$\mathbf{F}_j^{'} = \mathbf{MAX}_{\{i \mid \|x_i - x_j \le r\|\}} \{h(\mathbf{F_i}, \mathbf{x_i} - \mathbf{x_j})\} \tag{3}$$

where $\mathbf{F_i}$ is the i^{th} row of $\mathbf{F}$, $\mathbf{F_j}$ is the j^{th} row of $\mathbf{F}^{'}$, $h : \mathbb{R}^C \rightarrow \mathbb{R}^{C^{'}}$ is the MLP, and **MAX** is the max pooling layer (Wang et al., 2020a,b).

In this work, we experiment with three variants of FeatLoc: FeatLoc, FeatLoc+, and FeatLoc++. FeatLoc and FeatLoc+ have the same structural components as illustrated in Fig. 3, where FeatLoc+ is modified by adding additional hidden layers in the encoder E1, E2, and the regressor layer. In FeatLoc++, the encoder block E1 of the feature joining layer is replaced with a multi-scale grouping set abstraction, as seen in Fig. 4.

3.2.3. Regressor block and loss function

This section describes the regressor block and loss function we used to train the architecture. The regressor layer consists of a sequence of fully connected layers, which consume the obtained feature vector from the former feature joining block to estimate the 6-DoF pose $[\hat{\mathbf{t}}, \hat{\mathbf{r}}]$.

Given a feature set $\mathscr{P}$ extracted from image $\mathscr{I}$ and its corresponding pose label $[\hat{\mathbf{t}}, \hat{\mathbf{r}}]$, our network predicts the 6-DoF camera pose as two disjoint vectors. The output vector consists of a camera position $\mathbf{t} \in \mathbb{R}^3$ and camera orientation $\mathbf{r} \in \mathbb{R}^4$ in quaternion form.

We follow (Kendall, 2016) in the choice of objective function. The parameters inside the neural network are optimized using L1 loss according to the following function:

$$loss(\theta) = \|\mathbf{t} - \hat{\mathbf{t}}\|_1 e^{-\beta} + \beta + \|\log\mathbf{r} - \log\hat{\mathbf{r}}\|_1 e^{-\gamma} + \gamma \tag{4}$$

where β and γ are the weights used to balance the learning losses be-

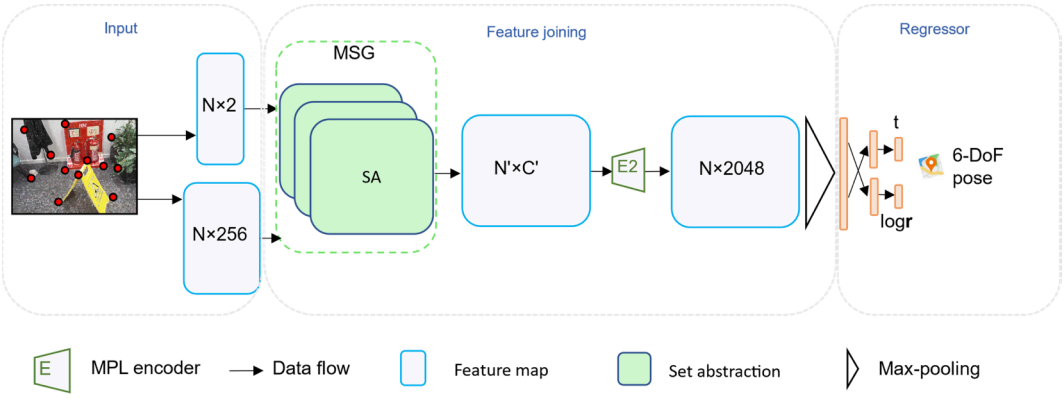

Fig. 4. Architecture of FeatLoc++. The main components are the same as those of FeatLoc, which includes an input layer, feature joining, and a pose regressor. The new structure's only change is in the feature joining layer, where the encoder E1 is replaced with a multi-scale grouping (MSG) set abstraction (SA) module. FeatLoc++ can adaptively learn to combine features from multiple scales and capture the local context of additional synthetic training samples with this replacement.

tween position and orientation; for all scenes, β and γ are simultaneously learned during the training period from their initial values of β_0 and γ_0, respectively. log$\mathbf{r}$ is the logarithmic form of quaternion $\mathbf{r}$, which is defined as.

$$\log\mathbf{r} = \begin{cases} \dfrac{\mathbf{v}}{\|\mathbf{v}\|}\cos^{-1}u, & \text{if } \|\mathbf{v}\| \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

where $\mathbf{r} = (u, \mathbf{v})$, u is a real scalar part, and $\mathbf{v}$ is a 3D vector of the imaginary part. This parameterization of camera orientation as the logarithm of a quaternion unit can map any rotations in 3D space uniquely because the quaternion itself is not unique. In practice, both $-\mathbf{r}$ and $\mathbf{r}$ are represented as the same rotation because two hemispheres can be used to represent a single rotation. In addition, the logarithmic parameterization of the rotation parameter showed a better performance than learning from the original 4D quaternion (Wang et al., 2020a,b). Thus, all experiments used the logarithm of quaternions to ensure all rotations are restricted to the same hemisphere.

3.3. Data augmentation

Training a deep neural network requires a large number of training samples (Krizhevsky et al., 2012), especially in the task of camera pose regression (Sattler et al., 2019), where the camera pose labels are in a continuous domain. To solve this problem, we propose a simple augmentation approach to synthesize more training data. Our FeatLoc model can thus learn to generalize beyond the limitation of the training data.

The general approaches for augmenting data for image classification include image distortions, cropping, rotation, and color changing. Because applying these methods to the training data does not affect the class label, it thus enriches the DNN model with new unseen data. In our case, however, these traditional distortion approaches cannot be applied as they would affect the camera pose. Therefore, we leverage the 3D cloud map generated based on SfM toolboxes using the training samples. Our augmented method is inspired by previous work (Purkait et al., 2018; Irschara et al., 2009); thus, the following strategy is generally similar to that of (Purkait et al., 2018). However, the synthesis strategy of this work is much simpler compared with (Purkait et al., 2018).

Given a 3D cloud map $\mathscr{D}$, a training dataset $\mathscr{T}$, and the intrinsic camera parameter $\mathbf{c}$ of the dataset, where $\mathscr{D}$ consists of k different cloud point sets $\mathbf{D}_i$, $\mathscr{T}$ consists of k different feature sets $\mathscr{P}_i$ and 6-DoF camera pose $\mathbf{T}_i$. k is the number of image samples in the training dataset. The camera model in this work is assumed to be a pinhole camera; thus, the relationship between a 3D cloud point $\mathbf{p} = (x, y, z)^T \in \mathbb{R}^3$, $\mathbf{p} \in \mathbf{D}$ and a 2D pixel position $x = (i, j)^T \in \mathbb{R}^2$ is as follows:

$$\pi(x, y, z)^T = \left(\frac{f_x x}{z} + c_x, \frac{f_y y}{z} + c_y\right) = (i, j)^T \tag{6}$$

where f_x, f_y and c_x, c_y refer to the focal length and optical center of the camera, respectively. Here, we represent the 6-DoF camera pose using transformation matrix $\mathbf{T}$. The transformation matrix set is also known as the special Euclidean group.

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \middle| \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{7}$$

For each training sample $\{\mathscr{P}_i, \mathbf{T}_i\}$, we randomly synthesize 50 relative poses $\mathbf{T}_{ij}$ based on the original camera pose $\mathbf{T}_i$, where the translations and orientations of $\mathbf{T}_{ij}$ are chosen uniformly within the ranges of $[-\alpha, +\alpha]$ and $[-d, +d]$, respectively.

Algorithm 1: Data augmentation based on 3D cloud map

Input: α, d, $\mathcal{T}$, $\mathcal{D}$, $\mathbf{c}$, $threshold$
- α is the random shift interval of all orientations,
- d is the random shift interval of all directions,
- $\mathcal{T}$ training data, where $\mathcal{T} = \{\mathcal{P}_i, \mathbf{T}_i | i = 1, ..., k\}$. $\mathcal{P}$ is the feature set and $\mathbf{T}$ is the camera pose,
- $\mathcal{D}$ is the 3D cloud map, where $\mathcal{D} = \{\mathbf{D}_i | t = 1, ..., k\}$,
- $\mathbf{c}$ represents the intrinsic camera parameters, where $\mathbf{c} = [f_x, f_y, c_x, c_y]$
- $threshold$ is the minimum number of valid reprojected key points.

Output: $\mathcal{A}$
- $\mathcal{A}$ represents the augmented camera poses.

```
i := 0; j := 0
while i < k, i++ do
    while j < 50, j++ do
        T_ij ← do_rand(α, d)
        T_wi ← T_i
        T_wj = T_wi * T_ij
        (P*, T*) ← reproject(T_wj, D_i, c)
        if length(P*) > threshold then
            A.append((P*, T*))
```

Assume that α_x, α_y, and α_z are roll, pitch, and yaw counterclockwise

rotations, respectively, which are chosen randomly within the range of $[-\alpha, +\alpha]$, and t_x, t_y, and t_z are the shifts along all directions, which are also chosen randomly from the interval $[-d, +d]$. This random process is illustrated in line 4 of Algorithm 1, where the relative camera pose $\mathbf{T}_{ij}$ will be calculated based on translation vector $\mathbf{t}_{ij} = (t_x, t_y, t_x)^T$ and the random shifts of $\alpha_x, \alpha_y, \alpha_z$. The new synthesized camera pose can thus be described as follows:

$$\mathbf{T}_{wj} = \mathbf{T}_{wi}\mathbf{T}_{ij} \tag{8}$$

where the $\mathbf{T}_{wi}$ and $\mathbf{T}_{wj}$ are the reference and synthesized camera coordinates in the world coordinate system, respectively. $\mathbf{T}_{ij}$ is the relative transformation matrix between reference and synthesized camera pose, computed as follows:

$$\mathbf{T}_{ij} = \begin{bmatrix} \mathbf{R}_{ij} & \mathbf{t}_{ij} \\ 0^T & 1 \end{bmatrix} \tag{9}$$

$$\mathbf{R}_{ij} = \mathbf{R}_x(\alpha_x)\mathbf{R}_y(\alpha_y)\mathbf{R}_z(\alpha_z) \tag{10}$$

$$\mathbf{R}_x(\alpha_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_x & -\sin\alpha_x \\ 0 & \sin\alpha_x & \cos\alpha_x \end{bmatrix} \tag{11}$$

$$\mathbf{R}_y(\alpha_y) = \begin{bmatrix} \cos\alpha_y & 0 & \sin\alpha_y \\ 0 & 1 & 0 \\ -\sin\alpha_y & 0 & \cos\alpha_y \end{bmatrix} \tag{12}$$

$$\mathbf{R}_z(\alpha_z) = \begin{bmatrix} \cos\alpha_z & -\sin\alpha_z & 0 \\ \sin\alpha_z & \cos\alpha_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13}$$

This gives us the new pose labels for the synthetic viewpoints. To generate the synthetic key points of the new pose labels, we use Eq. (6) to reproject the 3D cloud points $\mathbf{D}_i$ to the new synthesized camera plane based on its coordinate system. Note that before reprojecting, we convert all related point cloud coordinates to the new coordinate systems of the synthetic poses. Once the reprojection process is done, we remove all invalid reprojected key points, which are those not in front of the camera, using the *threshold* parameter, which defines the minimum number of valid reprojected key points. We then discard entire synthetic ones if the number of valid reprojected points is less than the *threshold*. Finally, the valid feature descriptors are copied corresponding to its original feature set $\mathscr{P}_i$. Note that none of the prior knowledge of testing data is exploited during augmentation. The algorithm 1 summarizes the whole above process.

3.4. Comparisons to related work

The FeatLoc architecture is developed mainly based on the PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) structures. FeatLoc inherits the critical-points theory of the PointNet structure; thus, FeatLoc can learn to use the collection of critical key points. Unlike other deep pose regression approaches, which cannot synthesize additional data from existing training samples, our approach regresses the camera poses from sparse features descriptors. The performance can thus be significantly enhanced due to the addition of synthetic training samples. In addition, in the test time, the trained network can work well with the independent number of input descriptors, while other sparse feature regressor approaches (Purkait et al., 2018) require some pre-processing steps before localizing the image location.

FeatLoc vs. SPPNet (Purkait et al., 2018): FeatLoc can jointly learn the position and appearance of every sparse descriptor. This makes the FeatLoc descriptor independent of all extracted key points, making it feasible to apply critical-points theory (Qi et al., 2017a). The model is thus flexible to consume a different number of sparse descriptors and automatically capture the valuable descriptors. Moreover, SPPnet must randomly regulate the sparse descriptors into a fixed 2D grid before applying the convolutional layer. However, this random process renders data unnecessarily grid-like, which could cause issues or discard important information. Regarding the augmentation method, our approach is simpler to implement. We only use the information from reference images to generate the synthetic data, while in SPPNet, each synthetic image must use all 3D key points, leading to a time-consuming augmentation process.

FeatLoc vs. MapNet (Brahmbhatt, 2018): MapNet leverages the valuable properties of image sequences during the learning period to enhance the performance of camera relocalization. However, it still struggles to generalize beyond the training data or might not generalize at all (Sattler et al., 2019). We believe that with sufficient training samples from the environment, end-to-end pose regressor approaches can close this gap and will become practically relevant. Here, our FeatLoc can provide additional training data by learning from sparse features. Further, training FeatLoc does not require sequential images, leading to the more straightforward data collection.

4. Experimental evaluation

In this section, we discuss the different evaluation aspects of our approach on a small-scale indoor dataset to show the robustness of our approach in indoor scenarios and compare the obtained results to those of state-of-the-art methods.

4.1. Implementation details

We implemented our algorithm with Pytorch (Ketkar, 2017) using the ADAM optimizer (Kingma & Ba, 2014). We set the initial learning rate to 6e-4 with a weight decay of 5e-4. The mini-batch size was 86, and the weight initialization was $\beta_0 = -3.0$ and $\gamma_0 = -3.0$. All experiments were conducted on two NVIDIA GTX 1080ti GPUs, where FeatLoc++ requires approximately 5 ms to run on time. We used the same parameter settings of FeatLoc when evaluating with a different number of sparse features. Following the convention of previous works (Kendall, 2015, 2016; Walch et al., 2017; Wang et al., 2020a,b; Purkait et al., 2018), we computed the median error for both translation and orientation. We report the orientation errors in degrees and positional errors in meters. Each data setting was trained for 200 epochs.

The parameter settings of FeatLoc and FeatLoc++ are listed in Table 1 and Table 2. The initial FeatLoc architecture is rather simple compared to that of FeatLoc++. A comparison among different numbers of parameters can also be found in Table 3.

4.2. Dataset description and pre-processing

We evaluated our proposed approach on the 7-Scenes (Shotton et al., 2013) and the 12-Scenes (Valentin, 2016) public indoor benchmark datasets to verify the robustness of our approach in indoor scenarios. The 7-Scenes dataset consists of multi-sequence RGB-D images, where the spatial extent is less than 4 m. The Kinect sensor was used to capture the image sequences, and the ground truths of the camera poses were recorded using KinectFusion. The 12-Scenes dataset is very similar to 7-

Table 1
Parameter settings for FeatLoc architecture.

Feature Joining	
Layer Name	Feature dimension
MPL encoder 1 (E1)	[2, 32, 64, 128, 256]
MPL encoder 2 (E2)	[256, 2048]
Pose Regressor	
FC1 + LeakyReLu	[2048, 40]
FC2	[40, 3]
FC3	[40, 3]

Table 2
Parameter settings for FeatLoc++ architecture.

Feature Joining	
MSG SA setting name	Setting
Number output points	1024
Radius	[0.025, 0.05, 0.1]
Number samples	[32, 16, 8]
MPLs	[[256, 64, 64, 128], [256, 128, 128,256], [256, 128, 128,256]]
Layer Name	Feature dimension
MPL encoder 2 (E2)	[740, 2048]
Pose Regressor	
FC1 + LeakyReLu	[2048, 512]
FC2 + LeakyReLu	[512, 256]
FC3 + LeakyReLu	[256, 40]
FC2	[40, 3]
FC3	[40, 3]

Table 3
Comparison of the numbers of parameters (Note that SuperPoint (DeTone, 2018) costs about 1.3 M parameters).

Method	#param
SuperPoint (DeTone, 2018)& FeatLoc	2.0 M
SuperPoint (DeTone, 2018)& FeatLoc+	3.8 M
SuperPoint (DeTone, 2018)& FeatLoc++	4.1 M
Original PoseNet (GoogleNet) (Kendall, 2015)	8.9 M
Baseline (ResNet50) (Laskar, Melekhov, Kalia, & Kannala, 2017)	26.5 M

Scenes but has much larger indoor spaces.

To train our proposed network on different scenes of the datasets, we leveraged the Hloc (Sarlin et al., 2019) toolbox to collect the training data under sparse feature format. Hloc is a well-known modular toolbox for state-of-the-art 6-DoF visual localization based on sparse features. It implements hierarchical localization (Sarlin et al., 2019), leveraging image retrieval and feature matching to localize accurate monocular query images. We used Hloc for both extracting the feature descriptors and constructing the 3D cloud map from image training samples.

To create the 3D cloud map and sparse features, SuperPoint (DeTone, 2018) and SuperGlue (Sarlin, 2020) were set as the default feature extractor and matcher, relatively. The maximum number of features extracted using SuperPoint was set as 2048. This process extracted the sparse descriptors of both training and testing images. Note that Hloc only uses the training images to generate the 3D cloud map while extracting the features of test images for later evaluation.

4.3. Baselines

To validate the performance of our proposed FeatLoc, we compared it with several state-of-the-art learning-based visual localization methods. Because the 7-Scenes dataset has recently been evaluated extensively as a benchmark (Kendall, 2015, 2016; Wang et al., 2020a,b; Kendall, 2017; Brahmbhatt, 2018), comparing our method with those prior state-of-the-art methods using the 7-Scenes dataset was an ideal choice. We chose the following mainstream single-image-based methods as our baselines: PoseNet Spatial-LSTM (Walch et al., 2017), AtLoc (Wang et al., 2020a,b), MapNet (Brahmbhatt, 2018). It is worth mentioning that SPPNet (Purkait et al., 2018) is the only DNN-based approach, which learns from sparse features, as our method does. We also report the performance of MapNet+ (Brahmbhatt, 2018), the state-of-the-art method on this dataset using sequential image-based relocalization. Note that the sequential image-based approaches generally perform better than single-image-based method. We nevertheless still compare our approach with MapNet+ to examine how accurate our FeatLoc learning is on single-image-based and augmented data. Lastly, we report the performance of traditional metric localization on this dataset compared with that of regressor-based approaches. We used two state-of-the-art methods of feature extractor and feature matching, which are SuperPoint (DeTone, 2018) and SuperGlue (Sarlin, 2020), to produce the comparing results.

4.4. FeatLoc verification

This section verifies the FeatLoc architecture on learning with different numbers of sparse descriptors to show its effects on performance. The sparse features obtained from the Hloc toolbox are then separated into six different train-test datasets. Each dataset will have different numbers of features as follows: 68, 126, 256, 512, 1024, and 2048. To refine a lower number of features in each dataset, we leveraged the score values, which are the output of the SuperPoint feature extractor. This score represents the evaluation of each detected key point, ranging between 0 and 1, where a higher score represents a better key point. Fig. 5 illustrates different number of key points after filtering following the score values. The image example is taken from the third sequence of the Chess dataset.

Table 4 shows the change of FeatLoc performance on those datasets. It is clear that the performance of the regressor notably increases when the number of input features increases. Interestingly, the dataset with fewer features filtered with better scores has lower performance than others. As seen from Table 4, the best relocalization results were obtained with the highest number of features, which is 2048. In Table 5, we compare results of our FeatLoc with SPPNet (Purkait et al., 2018) when learning on only training data. SPPNet also consumes sparse features for its pose regression task. Our base architecture significantly outperformed SPPNet in this scenario. This reveals that the proposed FeatLoc approach can effectively learn meaningful features from a set of sparse descriptors for relocalization.

4.5. Learning with synthetic data

In this section, we present our experiment results on learning with additional synthetic data. For the scenes with lower volumes, such as Fire and Heads ($2.5m^3$ and $1m^3$ respectively) we used the following parameters to synthesize new viewpoints: $d = 0.3m$, $\alpha = 10$, and $threshold = 1200$. With the remaining scenes, we simply increased d to $d = 0.5m$. Because our augmentation approach is much simpler compared with that of SPPNet (Purkait et al., 2018) approach. It took only approximately 8 min for a sequence of 1000 training samples, while SPPNet required approximately two hours. Separately, generating a 3D cloud map costs about 18 min for a sequence of 1000 images of 480x640 size. Note that all remaining experiments used the number key points set as 2048.

Initially, we evaluated our base lightweight FeatLoc architecture with only 0.7 million parameters on learning with both training and synthetic data. However, the base FeatLoc faces overfitting when learning with extensive additional data. This could be due to the simplicity of this architecture. We thus increased the architecture weights to 2.5 million parameters, as shown in Table 6 in detail. We call this architecture FeatLoc+. We illustrate the results of FeatLoc (learning with only training data) and FeatLoc+ (learning with both types of data), which were evaluated on the most difficult scene, Stairs, in Fig. 6. It can be seen that the improvement of FeatLoc+ (+augmentation) compared with the base FeatLoc is not significant. The reason for this is likely that FeatLoc's base appears to not be very powerful in capturing the local context of the synthetic data. To resolve this problem, we replaced the encoder block E1 of the feature joining layer with a multi-scale grouping set abstraction layer, which is the original in PointNet++ (Qi et al., 2017b). We call this improved architecture FeatLoc++.Fig. 6 also illustrates the results of FeatLoc++ when learning with both training and synthetic data on the Stairs dataset. The results show that FeatLoc++ adaptively learned to combine features from multi-scale local contexts, thus leading to a significant improvement. Table 7 reports entire results of three FeatLoc variants when learning with and

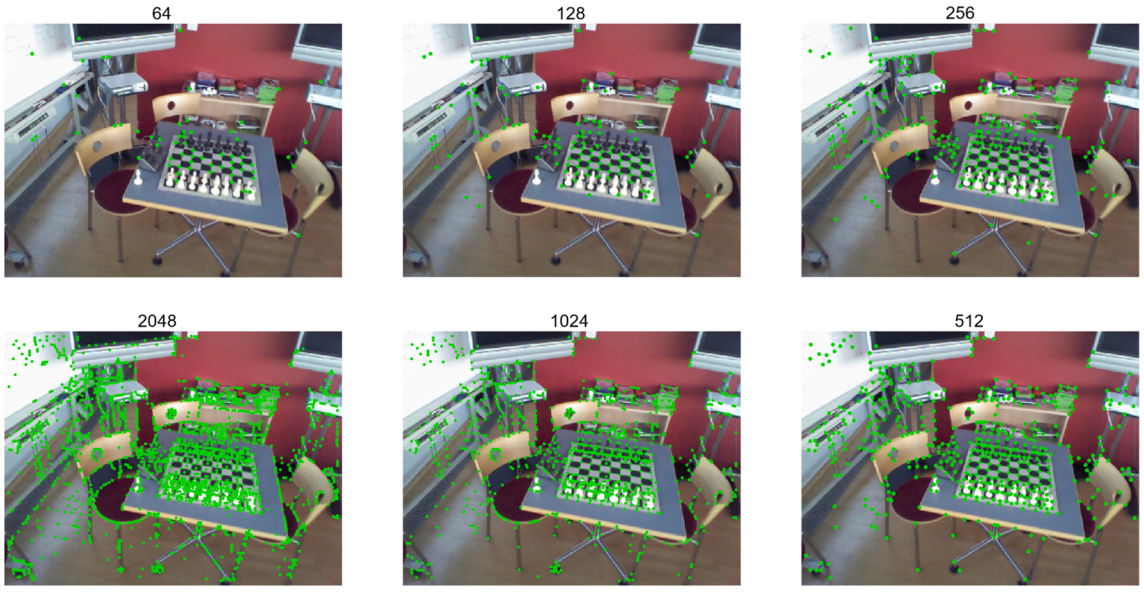

Fig. 5. Illustration on different number of feature key points on the Chess scene dataset, where the fewer features are filtered with better scores.

Table 4
Comparison of the effect of number of feature points on the errors when learning on the 7-Scenes dataset.

Number of features	68	126	256	512	1024	2048
Chess	0.29 m, 11.6°	0.36 m, 10.1°	0.20 m, 8.21°	0.18 m, 7.64°	0.17 m, 7.45°	**0.16 m, 6.45°**
Fire	0.45 m, 16.5°	0.40 m, 16.4°	0.39 m, 14.4°	0.37 m, **14.3°**	0.36 m, 14.7°	**0.36 m,** 15.6°
Heads	0.23 m, 16.4°	0.19 m, 16.0°	0.20 m, 15.1°	0.19 m, 15.1°	0.17 m, 14.6°	**0.17 m, 14.4°**
Office	0.42 m, 17.2°	0.35 m, 14.5°	0.31 m, 12.9°	0.29 m, 11.3°	0.26 m, 11.1°	**0.25 m, 10.3°**
Pumpkin	0.41 m, 12.5°	0.35 m, 9.95°	0.32 m, 8.44°	0.28 m, 8.29°	0.26 m, 8.21°	**0.24 m, 7.64°**
RedKitchen	0.53 m, 16.2°	0.47 m, 13.7°	0.42 m, 12.3°	0.38 m, 10.8°	0.34 m, 9.96°	**0.31 m, 8.91°**
Stairs	0.46 m, 14.7°	0.44 m, 11.8°	0.40 m, **10.6°**	0.38 m, 12.2°	0.36 m, 12.6°	**0.33 m,** 11.7°
Average error	0.40 m, 15.0°	0.37 m, 13.1°	0.32 m, 11.7°	0.30 m, 11.4°	0.27 m, 11.2°	**0.26 m, 10.7°**

Table 5
Comparison with SPP-Net in terms of median localization error when learning with no augmented data on 7-Scenes dataset.

Scene	SPP-Net	FeatLoc (ours)
Chess	0.22*m*, 7.61°	**0.16 m, 6.45°**
Fire	0.37 m, **14.1°**	**0.36 m,** 15.6°
Heads	0.22*m*, 14.6°	**0.17 m, 14.4°**
Office	0.32 m, **10.0°**	**0.25 m,** 10.3°
Pumpkin	0.47*m*, 10.2°	**0.24 m, 7.64°**
RedKitchen	0.34*m*, 11.3°	**0.31 m, 8.91°**
Stairs	0.40*m*, 13.2°	**0.33 m, 11.7°**
Average	0.33*m*, 11.6°	**0.26 m, 10.7°**

without augmented data.

Finally, we compare our results of the FeatLoc++ architecture on all seven scenes when learning with additional synthetic data with recent state-of-the-art APR-based methods. The comparison results are listed in Table 8, where FeatLoc++ significantly outperformed these APR-based baselines.

Table 6
Parameter settings for FeatLoc + architecture.

Layer Name	Feature dimension
MPL encoder 1 (E1)	[2, 32, 64, 128, 256]
MPL encoder 2 (E2)	[256, 512, 2048]
FC1 + LeakyReLu	[2048, 512]
FC2 + LeakyReLu	[512, 256]
FC2 + LeakyReLu	[256, 40]
FC2	[40, 3]
FC3	[40, 3]

4.6. Evaluation on 12-Scenes Dataset

In this section, we focus on localization on a larger indoor space, for which we use a publicly available 12-Scenes dataset (Valentin, 2016). This dataset was only used by DSAC++ (Brachmann, 2021) leading to a lack of APR baselines for comparison. To solve this issue, we evaluate this dataset with two recent state-of-the-art APR approaches PoseNet17 (Kendall, 2017) and MapNet (Brahmbhatt, 2018) as the baselines for comparison with our FeatLoc. When training PoseNet17 and MapNet, we set up the same configuration as illustrated in (Brahmbhatt, 2018). Here, we only evaluate our best variant of FeatLoc, which is FeatLoc++. The configuration for training FeatLoc++ was set as the same as above, while the augmentation parameters are set as $d = 0.5$, $\alpha = 10$, and $threshold = 1200$. We also provide localization results of state-of-the-art metric-based method (SuperPoint + SuperGlue) on this dataset. The results in Table 9 show that we outperform previous APR-based approaches by 40% in positional error and 14% in orientation error in general. Fig. 7 shows the camera trajectories for several testing sequences from the 12-Scenes dataset for MapNet and FeatLoc++Au (Au means learning with augmented data, No means learning with only training data). These experiment results validate the state-of-the-art of our proposed FeatLoc as an effort to avoid overfitting for indoor APR strategies.

5. Ablation study and efficiency evaluation

We conduct an ablation study on the FeatLoc results to examine how robust they are when operating under changing conditions. We also report the efficiency of the proposed method in two factors of storage requirements and running time compared with that of related works.

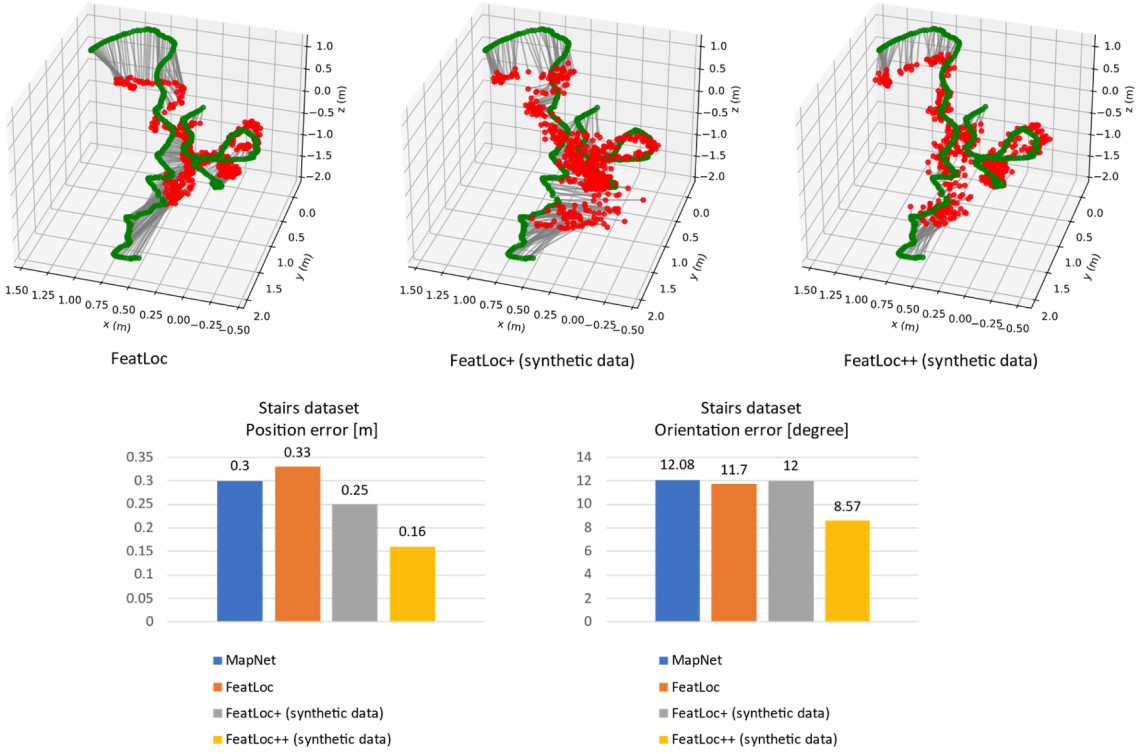

Fig. 6. Camera localization results on the Stairs scene in the 7-Scenes dataset (Shotton et al., 2013). For each subfigure, the top 3D plot shows the camera trajectory (green represents ground truth and red represents prediction), and the bottom bar charts show the comparison of location and orientation errors among MatNet (Brahmbhatt, 2018) and our FeatLoc, FeatLoc+, and FeatLoc++. All results were obtained from single images, where FeatLoc results were obtained with only training samples, while FeatLoc + and FeatLoc++ were trained on both training and augmented data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Comparison results of three FeatLoc versions when learning with and without augmented data.

	FeatLoc	FeatLoc+	FeatLoc++
Chess	0.16 m, **6.45°**	0.15 m, 6.62°	**0.11 m,** 7.56°
Fire	0.36 m, 15.6°	0.38 m, **14.4°**	**0.32 m,** 16.2°
Heads	0.17 m, 14.4°	0.17 m, **15.0°**	**0.16 m,** 15.7°
Office	0.25 m, 10.3°	0.25 m, **9.19°**	**0.23 m,** 10.6°
Pumpkin	**0.24 m, 7.64°**	0.25 m, 8.53°	0.30 m, 8.90°
RedKitchen	0.31 m, 8.91°	0.29 m, **9.05°**	**0.27 m,** 11.2°
Stairs	0.33 m, **11.7°**	**0.30 m,** 12.7°	0.33 m, 12.1°
Average error	0.26 m, 10.7°	0.26 m, **10.8°**	**24.6 m,** 11.7°
	FeatLoc Au	FeatLoc + Au	FeatLoc++Au
Chess	0.21 m, 8.03°	0.21 m, 8.03°	**0.07 m, 3.66°**
Fire	0.48 m, 11.9°	0.24 m, 8.22°	**0.17 m, 5.95°**
Heads	0.20 m, 11.6°	0.14 m, 11.1°	**0.10 m, 7.57°**
Office	0.29 m, 9.59°	0.26 m, 6.77°	**0.16 m, 5.20°**
Pumpkin	0.14 m, 11.1°	0.22 m, 5.11°	**0.11 m, 3.86°**
RedKitchen	0.36 m, 9.07°	0.31 m, 7.02°	**0.20 m, 6.43°**
Stairs	0.37 m, 13.0°	0.25 m, 12.0°	**0.16 m, 8.57°**
Average error	0.32 m, 9.90°	0.22 m, 7.85°	**0.14 m, 5.89°**

Finally, we provide an additional analysis of our method on an outdoor dataset to understand how outdoor properties affect the proposed method.

5.1. Changing testing condition

We evaluate two models of FeatLoc: FeatLoc++No and FeatLoc++Au. Following the localization results in Table 8, we selected the chess scene for this evaluation, since it has the localization error that is nearly same as that of MapNet results (Brahmbhatt, 2018).

Effect of Brightness. Here we study the effect of brightness more carefully by varying its magnitude. We expected that our models which were learned on sparse features should be more robust compared to that of direct-image-based approaches. To test this hypothesis, we linearly interpolate between clean images (degree of brightness = 0) and more or less brightness images (degree of brightness = 120 or −120). Note that this change of image was made before extracting 2D sparse features. Fig. 8 illustrates the comparison results of our approach with recent state-of-the-art methods on increasing degree of brightness. Fig. 9 also shows that of results but decreasing brightness degree. Results shown in these figures illustrate that our models are more robust to brightness change compared with that of direct-RGB learning methods, even with FeatLoc++No (model of learning without augmentation data).

Effect of Shadow Noise. To solid the results above, we study one more noise type “shadow noise”. We randomly create shadow noise on some parts of testing images. We then linearly interpolate between clean images and completely shadow noise in testing images (transparency of shadow ranges from 0.0 to 0.8 as shown in Fig. 10). Fig. 10 shows results under changing shadow noise transparency. FeatLoc++ models achieve more stable results than that of PoseNet (Kendall, 2017) and MapNet (Brahmbhatt, 2018) under this type of noise.

5.2. System efficiency

To evaluate the efficiency of our proposed FeatLoc we analyze two factors of storage requirement and running time. We compared FeatLoc with traditional metric localization method (SuperPoint (DeTone, 2018) Structure from Motion - SfM) and regressor-based method (MapNet (Brahmbhatt, 2018)).

Storage Requirements. We report the storage requirements for each method with respect to 7-Scenes dataset in the Table 10. The results show that our method is more efficient in terms of memory requirements. It is very scalable and independent with both number of

Table 8
Comparison of median localization errors with existing APR-based and metric localization approach on 7-Scenes dataset.

Scene	Area or Volume	Metric-based SuperPoint (DeTone, 2018) +SuperGlue (Sarlin, 2020)	APR-based MapNet (Brahmbhatt, 2018)	MapNet+ (Brahmbhatt, 2018)	AtLoc (Wang et al., 2020a,b)	SPP-Net (Purkait et al., 2018)	FeatLoc++Au (ours)
Chess	6 m^3	0.02 m, 0.84°	0.08 m, 3.25°	0.10 m, 3.17°	0.10 m, 4.07°	0.12 m, 4.42°	**0.07 m, 3.66°**
Fire	2.5 m^3	0.02 m, 0.93°	0.27 m, 11.7°	0.20 m, 9.04°	0.25 m, 11.4°	0.22 m, 8.84°	**0.17 m, 5.95°**
Heads	1 m^3	0.01 m, 0.74°	0.18 m, 13.2°	0.13 m, 11.1°	0.16 m, 11.8°	0.11 m, 8.33°	**0.10 m, 7.57°**
Office	7.5 m^3	0.03 m, 0.92°	0.17 m, 5.15°	0.18 m, 5.38°	0.17 m, 5.34°	0.16 m, **4.99°**	**0.16 m**, 5.20°
Pumpkin	5 m^3	0.05 m, 1.27°	0.22 m, 4.02°	0.19 m, 3.92°	0.21 m, 4.37°	0.21 m, 4.89°	**0.11 m, 3.86°**
RedKitchen	18 m^3	0.05 m, 1.40°	0.23 m, 4.93°	0.20 m, 5.01°	0.23 m, 5.42°	0.21 m, **4.76°**	**0.20 m**, 6.43°
Stairs	7.5 m^3	0.05 m, 1.57°	0.30 m, 12.1°	0.30 m, 13.4°	0.26 m, 10.5°	0.22 m, **7.17°**	**0.16 m**, 8.57°
Average	6.8 m^3	0.03 m, 1.09°	0.21 m, 7.77°	0.19 m, 7.29°	0.20 m, 7.56°	0.18 m, 6.20°	**0.14 m, 5.89°**

Table 9
Comparison of median localization errors with existing APR-based and metric localization approach on 12-Scenes dataset.

Scene	Area or Volume	Metric-based SuperPoint (DeTone, 2018) +SuperGlue (Sarlin, 2020)	APR-based PoseNet (Kendall, 2017)	MapNet (Brahmbhatt, 2018)	FeatLoc++No (ours)	FeatLoc++Au (ours)
apt1_kitchen	33 m°	0.02 m, 0.20°	0.62 m, 6.75°	0.48 m, 5.28°	0.53 m, 14.0°	**0.32 m, 5.19°**
apt1_living	30 m^3	0.02 m, 0.18°	0.61 m, 6.03°	0.50 m, 4.84°	0.55 m, 9.75°	**0.26 m, 3.89°**
apt2_bed	14 m^3	0.02 m, 0.29°	0.65 m, 5.66°	0.58 m, 6.02°	0.60 m, 10.2°	**0.37 m, 5.39°**
apt2_kitchen	21 m^3	0.05 m, 0.20°	1.24 m, 6.84°	1.18 m, **6.18°**	1.21 m, 29.1°	**0.73 m**, 6.37°
apt2_living	42 m^3	0.03 m, 0.21°	0.78 m, 7.61°	0.65 m, 7.20°	0.73 m, 11.4°	**0.40 m, 5.71°**
apt2_luke	53 m^3	0.02 m, 0.28°	0.66 m, 7.10°	0.49 m, 6.66°	0.59 m, 11.4°	**0.33 m, 4.85°**
office1_gates362	29 m^3	0.04 m, 0.24°	1.05 m, 5.67°	0.91 m, 5.50°	1.03 m, 8.95°	**0.52 m, 5.22°**
office1_gates381	44 m^3	0.02 m, 0.23°	0.70 m, 8.23°	0.62 m, 7.96°	0.73 m, 12.9°	**0.42 m, 6.23°**
office1_lounge	38 m^3	0.03 m, 0.22°	0.77 m, 7.35°	0.56 m, 6.0°	0.97 m, 8.30°	**0.39 m, 4.50°**
office1_manolis	50 m^3	0.02 m, 0.27°	0.64 m, 6.56°	0.54 m, 5.25°	0.66 m, 11.9°	**0.30 m, 4.67°**
office2_5a	38 m^3	0.02 m, 0.23°	0.59 m, 5.65°	0.54 m, 5.48°	0.61 m, 8.05°	**0.31 m, 4.32°**
office2_5b	79 m^3	0.02 m, 0.17°	0.52 m, 4.32°	0.47 m, **3.80°**	0.51 m, 7.29°	**0.23 m**, 4.14°
Average	39 m^3	0.03 m, 0.23°	0.74 m, 6.48°	0.63 m, 5.85°	0.73 m, 11.9°	**0.38 m, 5.04°**

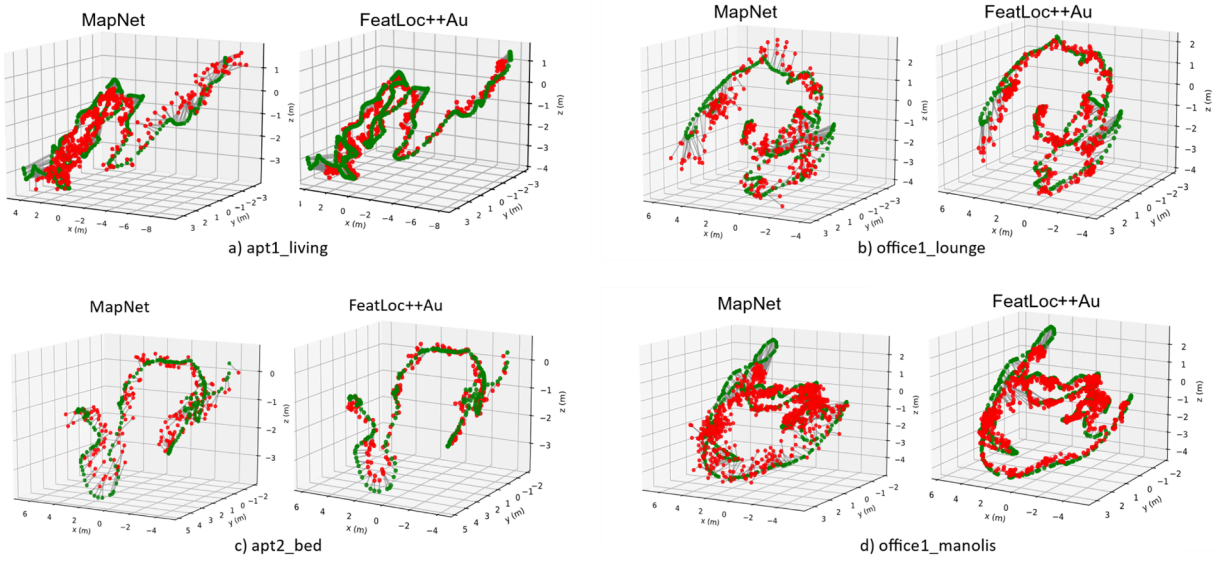

Fig. 7. Camera localization results on 12-Scenes dataset. For each subfigure, the 3D plot shows the camera trajectory (green for the ground truth and red for the prediction). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

training samples and area of environments. In contrast, the metric localization method requires a dramatic increase of memory alongside with the increase of data size.

Running time. To evaluate this factor of efficiency, we analyze three methods, our FeatLoc, MapNet (Brahmbhatt, 2018) and metric localization method. Along with these methods, MapNet requires about 9.4 ms to compute camera pose for a single image. Our FeatLoc++ consumes approximately 5 ms per frame from the 2D sparse feature (in total it consumes around 14 ms, since extracting feature using SuperPoint already takes 9 ms for the image size of 480x640). In contrast to regressor-based methods, the metric localization approach requires minutes to compute each camera pose and the running time scales $\mathcal{O}(n)$ with training data size (Wu, 2013; Kendall, 2015), as it needs to process many additional steps before the camera poses being computed, such as

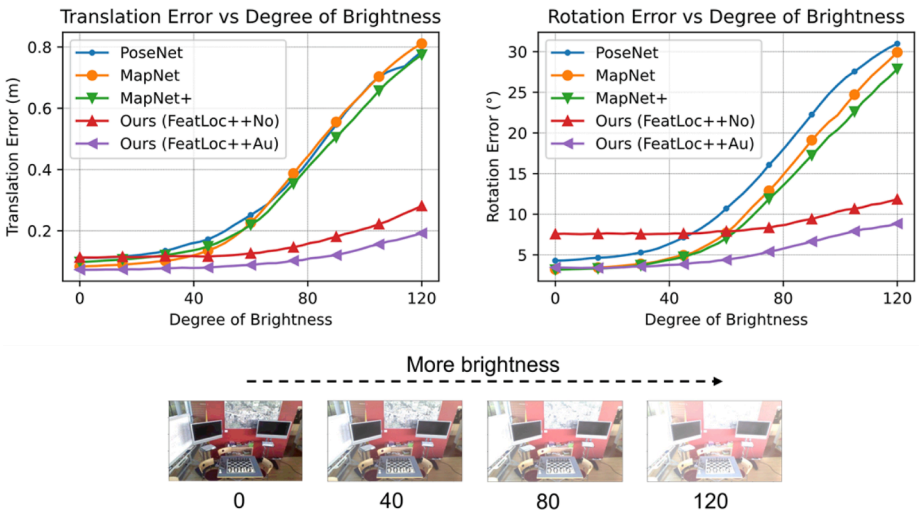

Fig. 8. Effect of Increasing Brightness. Two versions of FeatLoc++ (FeatLoc++No means learning without addition of augmented data and FeatLoc++Au means learning with addition of augmented data) compared to three recent state-of-the-arts localization models PoseNet17 (Kendall, 2017), MapNet, MapNet+ (Brahmbhatt, 2018). The FeatLoc models outperform the direct-image-based models in terms of presence of brightness.

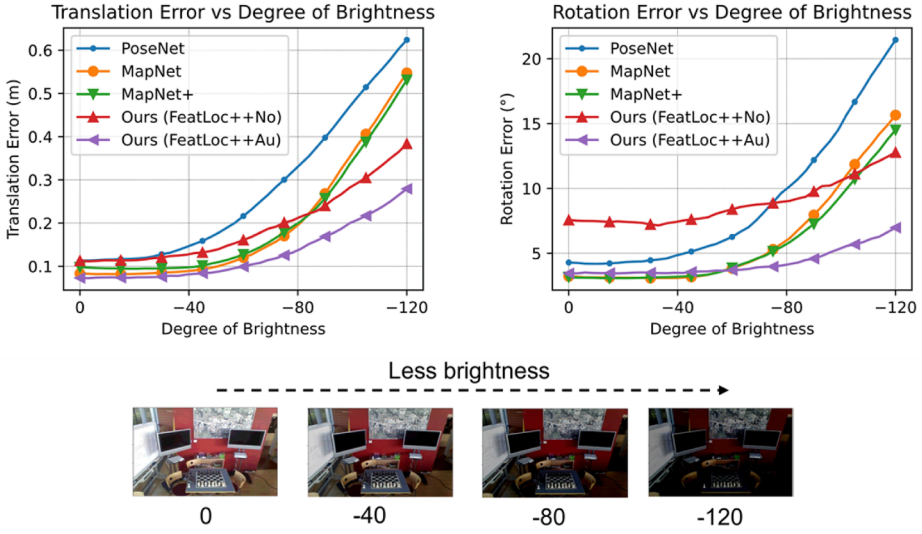

Fig. 9. Effect of Decreasing Brightness.

matching 2D-3D features, filtering good ones, etc.

5.3. FeatLoc for outdoor

In this section, we provide an additional evaluation of FeatLoc approach on an outdoor dataset Cambridge (Kendall, 2015) to analyze how accurate it is when applying to outdoor scenarios. For experiments on this dataset, we use the same configuration for training the network while the augmentation parameters are set as $d = 2.0$, $\alpha = 15$, and $threshold = 1200$. Note that for the outdoor scenes, we only synthesize pose along the detected horizontal plane. As we can see in Table 11, our method outperforms PoseNet (Kendall, 2015) in terms of translation error. However, they are just comparable with that of SPPNet (Purkait et al., 2018). The reasons come from our simplistic data augmentation technique. It is only robust for indoor environments that consist of more trustable features while the outdoor scenes contain various kinds of noise and less trustable ones. In order to continuously synthesize outdoor data, it is necessary to re-project the entire 3D cloud map onto a single augmented image plane with some additional preprocessing steps before obtaining the final reliable ones, as is done in (Purkait et al., 2018). In this work, however, we have proved that this labor-intensive procedure is not necessary for indoor scenarios.

6. Discussion

This section explains why FeatLoc++ is better than FeatLoc when learning with additional synthetic data. The initial FeatLoc version only inherits the symmetric function theory of PointNet (Qi et al. (2017a)), while FeatLoc++ additionally leverages the hierarchical point set feature learning of PointNet++ (Qi et al., 2017b). It has been proved that the original PointNet architecture does not capture the local structure induced in metric spaces of cloud point data (Qi et al., 2017b). The initial FeatLoc is thus limited in its ability to generalize to synthetic training data. In addition, due to the simplicity of our data augmentation approach, the synthesized samples only remain as separate features of the original images. In the testing phase, however, the new extracted features of test images will have many additional different sparse descriptors, which makes it difficult for FeatLoc to learn and capture the local context of the new synthesized data. Note that the augmented data retain most of the information from the original images while changing

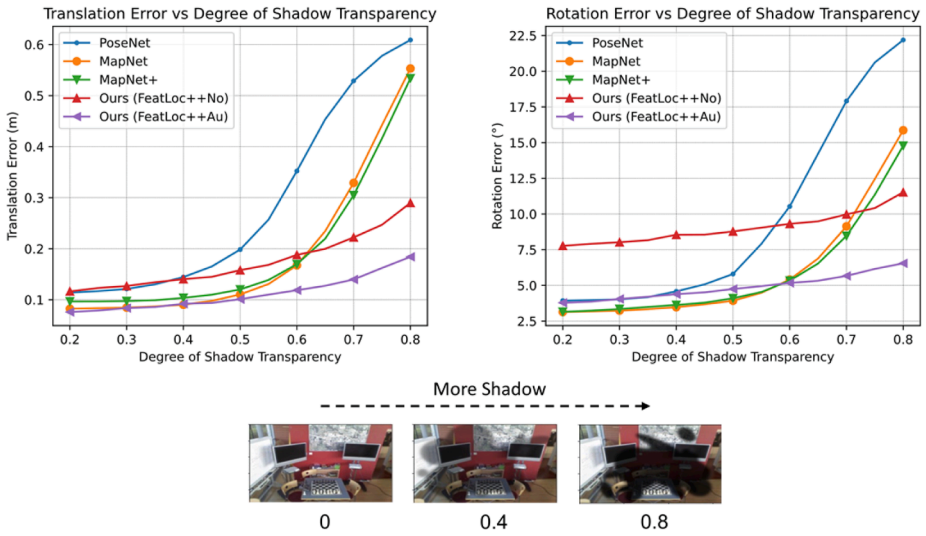

Fig. 10. Effect of Shadow Noise.

Table 10
Comparison of storage requirements on 7-Scenes (Shotton et al., 2013) dataset. We compared three methods of SuperPoint Structure from Motion (SfM), MapNet (Brahmbhatt, 2018), and our proposed FeatLoc++ (included SuperPoint's weight of 5.2 MB).

	Area or	#training	SuperPoint (DeTone, 2018)	MapNet (Brahmbhatt, 2018)	SuperPoint (DeTone, 2018) &
Scene	Volume	samples	SfM		FeatLoc++(ours)
Heads	1 m^3	1000	1.12 GB	268.3 MB	38.8 MB
Fire	2.5 m^3	2000	2.30 GB	268.3 MB	38.8 MB
Stairs	7.5 m^3	2000	2.27 GB	268.3 MB	38.8 MB
Pumpkin	5 m^3	4000	4.55 GB	268.3 MB	38.8 MB
Chess	6 m^3	4000	4.55 GB	268.3 MB	38.8 MB
Office	7.5 m^3	6000	6.85 GB	268.3 MB	38.8 MB
RedKitchen	18 m^3	7000	7.98 GB	268.3 MB	38.8 MB

Table 11
Comparison of median localization errors on outdoor Cambridge dataset.

Scene	Area or Volume	PoseNet (Kendall, 2015)	FeatLoc++Au (Ours)	SPPNet (with synthesis data) (Purkait et al., 2018)
King's College	5600 m	1.66 m, 4.86°	1.30 m, 3.84°	**0.74 m, 0.96°**
Old Hospital	2000 m^2	2.62 m, 4.90°	**2.05 m**, 6.06°	2.18 m, **3.92°**
Shop Facade	875 m^2	1.41 m, 7.18°	0.91 m, 7.50°	**0.59 m, 2.53°**
StMary's Church	4800 m^2	2.45 m, 7.96°	2.99 m, 10.4°	**1.83 m, 3.35°°**

only the position and distribution of sparse descriptors. Therefore, FeatLoc performs well on the training data while facing overfitting when learning with both training and synthesized data.

7. Conclusion

This paper proposed a novel approach for regressing the global 6-DoF camera pose from sparse feature descriptors for indoor environments. Our contributions can be summarized as follows:

- We addressed the domain adaption problem in absolute pose regression by augmenting additional training. The augmented data may cover the poses in the regions not available in the training data.
- We proposed an architecture that can directly consume sparse feature descriptors and showed its effectiveness compared to state-of-the-art methods.
- The proposed data augmentation method is a relatively simple but efficient technique for generating an unlimited amount of synthetic training data based on a 3D cloud map.
- We performed extensive experiments to validate our architecture and the proposed augmentation method. The results demonstrate that our method outperforms state-of-the-art APR-based approaches in terms of indoor scenarios.
- We have verified that learning from sparse feature can encourage the framework mitigating the impacts from changing illumination or gradual change of environments.

However, this study still consists of several limitations:

- Proposed method relies on sparse features, it might be less robust when working under less contextual environments.
- Although the proposed augmentation strategy is rather simple, it might be limited to environments where having less trustable features such as outdoor scenarios.

In future work, we plan to investigate the feasibility of improving the network architecture to understand the invariance in descriptors when changing the viewpoint. If the network can reveal the properties of matched descriptors, the performance could be remarkably improved and comparable with that of geometry-based approaches. In addition, we plan to enhance the method's ability to work under dynamic and less contextual environments.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Albl, C., Kukelova, Z., Pajdla, T., 2015. R6p-rolling shutter absolute camera pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2292–2300.

Brachmann, E.K., 2017. Dsac-differentiable ransac for camera localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6684–6692.

Brachmann, E., Rother, C., 2018. Learning less is more-6d camera localization via 3d surface regression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4654–4662.

Brachmann, E., Michel, F., Krull, A., Yang, M.Y., Gumhold, S., et al., 2016. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3364–3372.

Brahmbhatt, S.G., 2018. Geometry-aware learning of maps for camera localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2616–2625.

Castle, R.K., 2008. Video-rate localization in multiple maps for wearable augmented reality. In: 2008 12th IEEE International Symposium on Wearable Computers, pp. 15–22.

Chum, O., Matas, J., 2008. Optimal randomized RANSAC. IEEE Trans. Pattern Anal. Mach. Intell. 30 (8), 1472–1482.

DeTone, D.T., 2018. Superpoint: Self-supervised interest point detection and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp. 224–236.

Donoser, M.S., 2014. Discriminative Featureto-Point Matching in Image-Based Locallization. CVPR.

Häne, C., Heng, L., Lee, G.H., Fraundorfer, F., Furgale, P., Sattler, T., Pollefeys, M., 2017. 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. Image Vis. Comput. 68, 14–27.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

Huang, Z., Xu, Y., Shi, J., Zhou, X., Bao, H., Zhang, G., 2019. Prior guided dropout for robust visual localization in dynamic environments. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2791–2800.

Hyeon, J., Kim, J., Doh, N., 2021. Pose Correction for Highly Accurate Visual Localization in Large-scale Indoor Spaces. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 15974–15983.

Irschara, A., Zach, C., Frahm, J.-M., Bischof, H., 2009. From structure-from-motion point clouds to fast location recognition. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 2599–2606.

Juan, L., Gwun, O., 2009. A comparison of sift, pca-sift and surf. Int. J. Image Process. (IJIP) 3, 143–152.

Kendall, A., 2016. Modelling uncertainty in deep learning for camera relocalization. In: 2016 IEEE international conference on Robotics and Automation (ICRA), pp. 4762–4769.

Kendall, A., 2017. Geometric loss functions for camera pose regression with deep learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5974–5983.

Kendall, A.G., 2015. Posenet: A convolutional network for real-time 6-dof camera relocalization. In: Proceedings of the IEEE international conference on computer vision, pp. 2938–2946.

Ketkar, N., 2017. Introduction to pytorch. In: Ketkar, N. (Ed.), Deep Learning with Python. Apress, Berkeley, CA, pp. 195–208.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kneip, L., Scaramuzza, D., Siegwart, R., 2011. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In: CVPR 2011, pp. 2969–2976.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inform. Process. Syst. 25, 1097–1105.

Laskar, Z., Melekhov, I., Kalia, S., Kannala, J., 2017. Camera relocalization by computing pairwise relative poses using convolutional neural network. In: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 929–938.

Li, Y.S., 2012. Worldwide pose estimation using 3d point clouds. In: European conference on computer vision, pp. 15–29.

Li, Y., Snavely, N., Huttenlocher, D.P., 2010. Location recognition using prioritized feature matching. In: European conference on computer vision, pp. 791–804.

Lim, H., Sinha, S.N., Cohen, M.F., Uyttendaele, M., Kim, H.J., 2015. Real-time monocular image-based 6-DoF localization. Int. J. Robot. Res. 34 (4-5), 476–492.

Liu, X., Qi, C.R., Guibas, L.J., 2019. Flownet3d: Learning scene flow in 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 529–537.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision 60 (2), 91–110.

Luo, Z., Zhou, L., Bai, X., Chen, H., Zhang, J., Yao, Y., Quan, L., 2020. Aslfeat: Learning local features of accurate shape and localization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 6589–6598.

Melekhov, I., Ylioinas, J., Kannala, J., Rahtu, E., 2017. Image-based localization using hourglass networks. In: Proceedings of the IEEE international conference on computer vision workshops, pp. 879–886.

Meng, L., Chen, J., Tung, F., Little, J.J., Valentin, J., de Silva, C.W., 2017. Backtracking regression forests for accurate camera relocalization. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6886–6893.

Meng, L., Tung, F., Little, J.J., Valentin, J., de Silva, C.W., 2018. Exploiting points and lines in regression forests for RGB-D camera relocalization. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6827–6834.

Naseer, T., Burgard, W., 2017. Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1525–1530.

Purkait, P., Zhao, C., Zach, C., 2018. Synthetic View Generation for Absolute Pose Regression and Image. Synthesis BMVC, (p. 69).

Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 652–660.

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413.

Radwan, N., Valada, A., Burgard, W., 2018. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. IEEE Rob. Autom. Lett. 3 (4), 4407–4414.

Sarlin, P.E., 2020. Superglue: Learning feature matching with graph neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 4938–4947.

Sarlin, P.-E., Cadena, C., Siegwart, R., Dymczyk, M., 2019. From Coarse to Fine: Robust Hierarchical Localization at Large Scale. CVPR.

Sattler, T.L., 2011. Fast image-based localization using direct 2d-to-3d matching. In: 2011 International Conference on Computer Vision, pp. 667–674.

Sattler, T.L., 2016. Efficient & effective prioritized matching for large-scale image-based localization. IEEE Trans. Pattern Anal. Mach. Intell. 39, 1744–1756.

Sattler, T., Torii, A., Sivic, J., Pollefeys, M., Taira, H., Okutomi, M., Pajdla, T., 2017. Are large-scale 3d models really necessary for accurate visual localization?. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1637–1646.

Sattler, T., Zhou, Q., Pollefeys, M., Leal-Taixe, L., 2019. Understanding the limitations of cnn-based absolute camera pose regression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 3302–3312.

Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A., 2013. Scene coordinate regression forests for camera relocalization in RGB-D images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2930–2937.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.

Valada, A., Radwan, N., Burgard, W., 2018. Deep auxiliary learning for visual localization and odometry. In: 2018 IEEE international conference on robotics and automation (ICRA), pp. 6939–6946.

Valentin, J.D., 2016. Learning to navigate the energy landscape. In: In 2016 Fourth International Conference on 3D Vision (3DV), pp. 323–332.

Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., Cremers, D., 2017. Image-based localization using lstms for structured feature correlation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 627–637.

Wang, B., Chen, C., Lu, C.X., Zhao, P., Trigoni, N., Markham, A., 2020a. Atloc: Attention guided camera localization. Proceedings of the AAAI Conference on Artificial Intelligence vol. 34, 10393–10401.

Wang, W., Wang, B., Zhao, P., Chen, C., Clark, R., Yang, B., Trigoni, N., et al., 2020. Pointloc: Deep pose regressor for lidar point cloud localization. arXiv preprint arXiv: 2003.02392.

Wu, C., 2013. Towards linear-time incremental structure from motion. In: 2013 International Conference on 3D Vision-3DV 2013, pp. 127–134.

Zhang, W., Kosecka, J., 2006. Image based localization in urban environments. In: Third international symposium on 3D data processing, visualization, and transmission (3DPVT'06), pp. 33–40.

Zheng, T., Chen, C., Yuan, J., Li, B., Ren, K., 2019. Pointcloud saliency maps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1598–1606.

Zhou, K., Chen, C., Wang, B., Saputra, M.R., Trigoni, N., Markham, A., 2021. VMLoc: Variational Fusion For Learning-Based Multimodal Camera Localization. Proceedings of the AAAI Conference on Artificial Intelligence vol. 35, 6165–6173.