*Journal of Computational Design and Engineering*, 2022, **9**, 1482–1497

DOI: 10.1093/jcde/qwac066
Advance access publication date: 26 July 2022
**Research Article**

# Camera localization with Siamese neural networks using iterative relative pose estimation

Daewoon Kim and Kwanghee Ko*

The School of Mechanical Engineering, Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Republic of Korea.
*Corresponding author. E-mail: khko@gist.ac.kr

## Abstract

This paper presents a novel deep learning-based camera localization method using iterative relative pose estimation to improve the accuracy of pose estimation from a single RGB image. Although most existing deep learning-based camera localization methods are more robust for textureless cases, illumination changes, and occlusions, they are less accurate than other non-deep learning-based methods. The proposed method improved the localization accuracy by using the relative poses between the input image and the training dataset images. It simultaneously trained the network for the absolute poses of the input images and their relative poses using Siamese networks. In the inference stage, it estimated the absolute pose of a query image and iteratively updated the pose using relative pose information. Real world examples with widely used camera localization datasets and our dataset were utilized to validate the performance of the proposed method, which exhibited higher localization accuracy than the state-of-the-art deep learning-based camera localization methods. In the end, the application of the proposed method to augmented reality was presented.

**Keywords:** camera localization, camera pose estimation, motion estimation, deep learning, neural network, augmented reality

## 1 Introduction

Camera localization, also known as camera pose estimation, estimates the position and orientation of a camera based on various types of data, such as RGB images, inertial sensor data, and 2.5-dimensional (2.5D) or 3D scanned values. It computes the position and orientation of camera coordinates relative to the reference (world) coordinates.

Parallel to the rapid development of smart wearable and personal handheld devices, camera localization is adopted in several applications, including autonomous navigation, robotics, games, augmented reality (AR), and mixed reality. AR is regarded as the most promising technology for industrial applications. It can assist in equipment repair tasks or aid workers in verifying discrepancies between actual products and their design by intuitively providing on-site fabrication information. However, AR cannot be successfully used in practice unless the physical geometric environment is adequately recognized, to determine an appropriate coordinate system to augment 2D or 3D content to the user on the mobile device's screen, which requires the exact location and orientation of the camera.

The pose of the camera corresponds to the position and viewing direction of the user in the reference coordinate system. Therefore, additional information or intuitive visual expressions for the location can be provided. A typical example is AR navigation, which can visually provide direction to the destination to the user based on the current position and heading orientation on the screen. In addition, position-based services or desired information associated with the user's position can be provided.

Camera localization is the problem of estimating camera pose from an input image. The pose is a vector of six degrees of freedom (DoFs), where the three DoFs correspond to the location and the others correspond to the orientation of the input image. The approaches to solve this problem can be categorized into three types. One is the feature-based method or handcrafted feature-based approach. Lowe (2004) proposed a scale-invariant feature transform (SIFT) method that extracts and matches the scale and, rotation-invariant features of images. It is widely employed to compare two images in terms of their similarity. The relative location and orientation between the images can be estimated using epipolar geometry constraints from 2D point correspondences. Bay *et al.* (2008), Leutenegger *et al.* (2011), and Rublee *et al.* (2011) proposed several variants of SIFT, that are currently widely used. Simultaneous localization and mapping (SLAM) is the most prevalent localization approach that adopts handcrafted features (Mur-Artal *et al.*, 2015, 2017). It computes the relative positions and orientations between inter-frame images, and generates a 3D map via feature matching. Structure from motion (SfM) is another method that uses handcrafted features for camera localization. It extracts image features and matches them between multiple images using the perspective-n-point (PnP) solver in the random sample consensus (RANSAC) loop (Wu, 2011). Unlike SLAM, which compares inter-frame image sequences, SfM uses all images to estimate the poses. Furthermore, it can yield accurate poses; however, this requires extensive time. SfM is often used for scanning 3D models or constructing a training dataset for deep learning-based camera localization when other sensory inputs are unavailable.

**Received:** March 21, 2022. **Revised:** July 12, 2022. **Accepted:** July 12, 2022

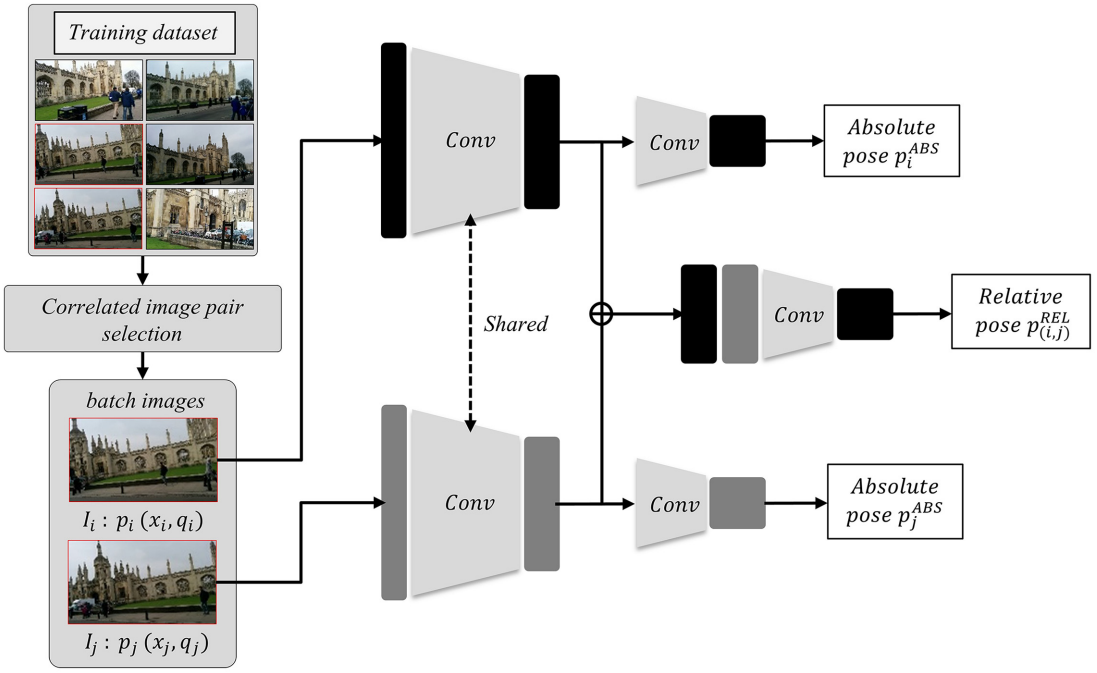
The other class adopts 3D models for pose estimation. It compares the geometry of a 3D model with the geometric features of an image, such as lines or edges, and determines the location and orientation of a scene using the PnP approach. Furthermore, it performs 2D–3D matching during pose computation, unlike the handcrafted feature-based method that performs 2D–2D matching. Drummond and Cipolla (1999) proposed an edge-based inter-frame camera-tracking method. In this method, sample points were placed on the extracted edges and the edges of a 3D model. The relative pose of the camera was estimated by minimizing the average distance between the features of the 2D edges and projected 3D model edges. Kim *et al.* (2018) used line segments and a model for camera pose estimation in an AR environment, and applied it to the shipbuilding fabrication process. Typically, a model-based method requires a good initial pose to obtain the correspondence between a 2D image and a 3D model, as described by Liu *et al.* (2017) and Yu *et al.* (2020).

Deep learning is another class of methods to solve camera localization problems. It has been widely used to solve several computer vision problems. It has demonstrated an overwhelming performance for image classification (Krizhevsky *et al.*, 2012), object detection (Girshick, 2015), and semantic segmentation (Long *et al.*, 2015). In addition, there has been a growing number of applications in diverse areas, such as small object detection in a remote sensing environment (Yuan *et al.*, 2021), analysis and forecasting of slope stability for geotechnical engineers (Mahmoodzadeh *et al.*, 2021), prediction of water inflow into tunnels (Mahmoodzadeh *et al.*, 2022), pneumonia detection on chest X-rays (Varshni *et al.*, 2019), and medical face mask detection in real-life images (Loey *et al.*, 2021).

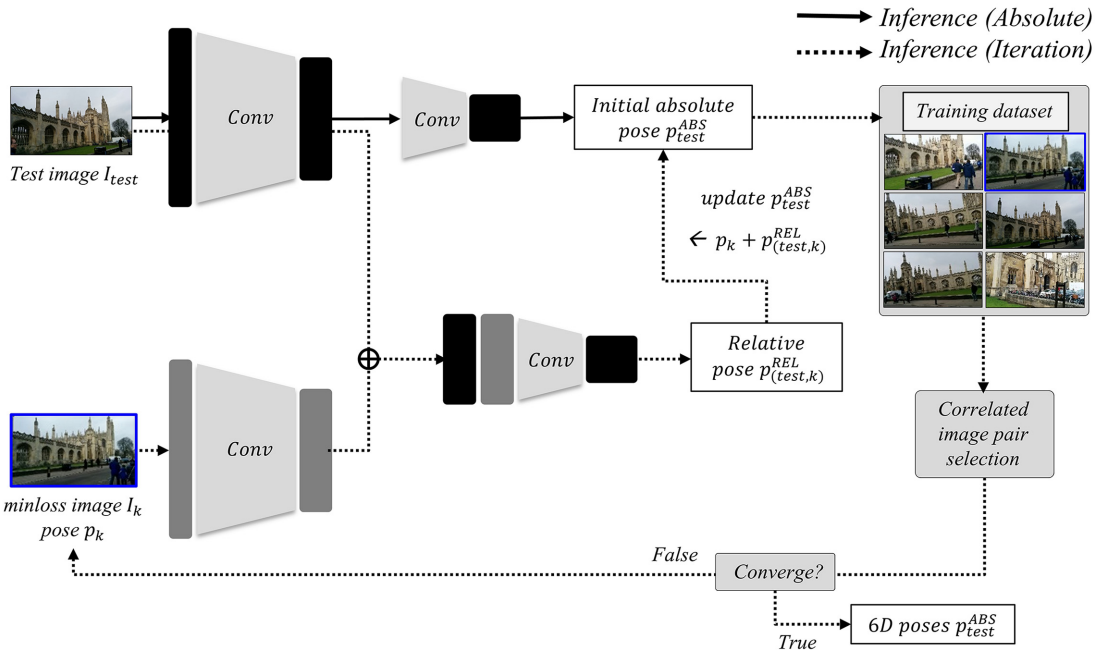
Recently, it has been used for camera pose estimation. Kendall *et al.* (2015) proposed the PoseNet. They applied a convolutional neural network (CNN) to the camera localization problem, which modified the object classification architecture, GoogLeNet (Szegedy *et al.*, 2015), to estimate the camera's location and orientation. Additional fully connected layers (FCLs) replacing the softmax classifier of GoogLeNet were have being introduced to regress the location and orientation. SfM was used to generate a labelled training dataset. PoseNet is an absolute pose regressor, which is also known as learned absolute pose (LAP) estimation. It estimates the location and orientation of a single input image. Usually, a feature extractor or an encoder is used to extract high-level features of an image. The extracted features are fed to the regressor to estimate the pose. Here, the estimated pose is the absolute pose of an image because its reference coordinates are based on world coordinates. PoseNet used a novel loss function that combines location and orientation, and transfer learning was applied in this approach to reduce the training time and achieve high localization accuracy. It yields robust results for textureless cases, significant illumination changes, and occlusions compared with feature-based methods. However, it may yield larger errors than feature-based and model-based methods. Hence, extensive studies have been conducted to improve the localization accuracy of deep learning-based camera localization approaches. Melekhov *et al.* (2017a) proposed Hourglass-Pose, which is an encoder–decoder used to regress the absolute pose of an input image. Naseer *et al.* (2017) proposed SVS-Pose, which utilizes the VGG-16 (Simonyan & Zisserman, 2014) architecture to replace GoogLeNet in PoseNet. In addition, studies have been conducted to improve the regressor, which is also known as the localizer in deep pose regression. For example, Walch *et al.* (2017) used long short-term memory (Hochreiter & Schmidhuber, 1997) and Zhang *et al.* (2018) used the Georgia Tech smoothing and mapping library (Carlone *et al.*, 2014) for PoseNet++. In addition, a novel type of loss function has being adopted to regress the absolute pose. Kendall and Cipolla (2017) proposed a geometric re-projection error that adopts the 3D points of an input scene. Brahmbhatt *et al.* (2018) proposed MapNet, which uses sensory inputs, such as GPS and visual odometry, and fused them for camera localization. Geometric constraints were formulated using bundle adjustment (Lourakis *et al.*, 2009) and pose-graph optimization (Grisetti *et al.*, 2010). An additional self-supervised video update was applied to improve the localization accuracy. VLocNet (Valada *et al.*, 2018) used learning with visual odometry, and VLocNet++ (Radwan *et al.*, 2018) applied an absolute pose regressor with semantic segmentation. Shavit *et al.* (2021) proposed attention-based camera localization, called TransPoseNet. It localizes the best position for corner-like image features and the best orientation for edge-like image features. IPR-Net (Shavit & Ferens, 2021) used a pre-trained encoder that was trained for visual similarity instead of a pose encoder trained for pose regression. It only trains the regressor part of the network, leading to a shorter training time and lighter storage space. Li *et al.* (2021) proposed VNLSTM-PoseNet, which directly adopts an input image without cropping to increase the receptive field of the training image, whereas other methods used the cropped image for training and test.

The learned relative pose (LRP) is another type of deep learning-based camera localization. It estimates the relative pose of the two images using two pose regressors. It uses two encoders to extract the high-level features of two images, and combines the feature maps generated by the two encoders into a single regressor. Finally, it estimates a six DoFs relative pose between the two images. Melekhov *et al.* (2017b) proposed relative camera localization using a CNN and obtained robust relative poses between the two images, the poses of which the existing methods failed to estimate. LRP has exhibited robust results for textureless surfaces, significant illumination changes, and occlusions compared with feature-based methods, which is similar to LAP. In addition, Siamese network architecture using RelocNet (Balntas *et al.*, 2018) and PairwiseNet (Laskar *et al.*, 2017) were used to estimate the relative pose of an image. Deep learning-based camera localization methods work robustly and exhibit faster inference speeds than other feature-based and model-based methods. However, they remain limited by their low localization accuracy, which is a significant drawback to these methods.

In this paper, a novel deep learning-based camera localization method is proposed. It uses a Siamese neural network architecture with iterative relative pose estimation. It comprises two stages: training and inference. During training, correlated image pairs in the training dataset are identified, and the pairs are used to train the relative poses. Next, a novel Siamese network architecture is utilized to train the two absolute poses of the two input images and the relative pose between them simultaneously. In addition, a novel loss function is introduced to train the network for absolute and relative poses. After the training is completed, the camera pose of an input image is estimated using the proposed method as follows. First, the absolute location and orientation of an input RGB image are estimated using the trained absolute pose regressor. Next, an image with minimum loss is selected as a correlated image from the training dataset. Subsequently, the relative pose between the input and selected images is estimated using the relative pose regressor. The absolute pose of the input image is then updated using the estimated relative pose. This update sequence is repeated until the sequence converges. The proposed correlated image pair selection (CIPS) step provides efficient relative pose training between the images in the training stage.




(a) Training stage.




(b) Inference stage.

**Figure 1:** Illustration of the proposed method.

The proposed method exhibited higher localization accuracy than other methods that estimate an absolute pose directly, using relative pose estimation during the inference stage. The experimental results of the proposed method are presented and compared with the results of other state-of-the-art camera localization studies involving deep learning. The experiments are conducted using the Cambridge Landmarks dataset, which is widely used for evaluating camera localization tasks. Furthermore, the proposed method adopted in AR applications was demonstrated using a thermal observation device (TOD) dataset.

## 2 Methodology

The objective of the proposed method is to estimate the location and orientation of the camera from a monocular RGB image. The overall structure of the proposed method is illustrated in Fig. 1.

## 2.1 Training stage

The training comprised two stages. In the first stage, a correlated image pair with an overlapping region was selected, where the images share the same pixels. In the second stage, a network was used to train the two absolute poses of the two correlated images and one relative pose between them using their ground-truth poses. Siamese networks were utilized to extract the features of two images with shared weights.

### i Correlated image pair selection

In this stage, several correlated image pairs, called batch images, were selected during training. Two images were considered correlated if they had the same world coordinate points or shared a region between them. The correlated image pairs with the absolute and relative poses of each pair were used to train the network.

Training the relative pose between uncorrelated images is almost meaningless, and adversely affects the training process. The proposed network required two correlated images for training. The network extracted high-dimensional features and matched them using CNNs. It always attempts to extract correspondences between two images based on quantitative measures and estimates the relative pose between them. However, this computation was performed even though the features do not match spatially. In this case, the relative pose between such images was incorrect, which adversely affected the network training. Therefore, providing correlated images as input to the training step is critical for the prediction performance of the network. The importance of correlated image selection is discussed in the experimental results and in Section 3.2.

The correlation between two images can be verified using one of two simple methods. The first method uses sequential coherence of the dataset. Most datasets used for camera localization are created from videos. When selecting an image with a random index $k$ from the training image dataset and the training image sampling gap threshold $d$, the images with indices of $(k-d)$ to $(k+d)$ in a video sequence are highly likely to be correlated because two consecutive inter-frame images must overlap unless a significant abrupt change occurs in the camera motion between the frames. Threshold $d$ was empirically selected as 30, which yielded good experimental results.

In the second method, two images are regarded as uncorrelated if the orientation difference between them is greater than the field of view (FoV) of the camera. This difference can be computed using Equation (1), in which the orientation information of each image is defined as a quaternion.

$$\cos\left(\frac{\theta}{2}\right) = q_1 \cdot q_2^{-1}, \tag{1}$$

where $q$ and $\theta$ denote the unit quaternion and angle between two quaternions, respectively. If the angular difference between two images is greater than the FoV, then the two images do not share the same scene or world coordinate points. If the images share at least one pixel, the two view-frustums intersect. The maximum angle difference between the two images is the angle of the orientations of the two images in which the view-frustums in each orientation intersect only at the corners. Half of the maximum angular difference was set as the threshold, to determine whether the two images are correlated. The correlated images were used for training because several matching image features can be extracted from the images to relate them to the relative poses when training the network using relative pose information.

However, it cannot be verified that the two images share the same scene, even though the view-frustums intersect. In this case, the positional difference between the two images could be problematic. If obstacles exist in front of the camera or if occlusion exists that obstructs the view of the camera in one orientation, then the images may not overlap, and pose computation fails. Hence, the overlap must contain the correspondence information. If a dataset with 3D point clouds or a depth map is available, it is easy to confirm whether the two images share the same scene. A point cloud is projected onto each image using the ground-truth poses with the camera intrinsic matrix $K$. By comparing the number of 2D points projected inside the two images, it can be determined whether the images share the same 3D points. If the number of projected 2D points is sufficiently large, it is assumed that the images are correlated. In the case of RGB-D images, one depth point can be projected onto another image. If the projected point is inside the image, then the two images share the same pixel. However, when the dataset had no point cloud or depth, the correlation was indirectly estimated. If only RGB images were provided, the correlation between the two images cannot be determined. Hence, another method must be used to determine whether two images share the same view. In this case, handcrafted feature matching was used to verify image correspondence. The ORB (Oriented FAST and Rotated BRIEF) feature (Rublee *et al.* 2011) was adopted to identify overlap. As mentioned in Section 1, handcrafted features were used to detect and match the rotation and scale-invariant features in the images. The percentage of matched features with a small distance over the number of detected features was used to determine whether two images were correlated. Algorithm 1 presents a pseudocode for determining whether two images are correlated and for generating a correlated image-pair matrix $C$. After the correlated image-pair matrix $C$ was constructed, the network was trained.

Algorithm 1. Pre-processing: CIPS.

*Number of input training images* : $n$
*Input RGB image sequence* : $I = \{I_1, I_2 \ldots, I_n\}$
*Ground truth poses of image sequence* : $P = \{P_1, P_2 \ldots, P_n\} = \{R_1|t_1, R_2|t_2 \ldots, R_n|t_n\}$
*Inter-frame index threshold number:* $d$
*(optional) Input depth image sequence* : $D = \{D_1, D_2 \ldots, D_n\}$
*(optional) 3D model points or point cloud* : $X$
*Correlated image pair matrix* $C(n \times n)$ *in Boolean with false initialization*
**for** $i = 1 : n$ **do**
**for** $j = i-d : i+d$ **do** // *automatically exclude index exceeding* $(1 \sim n)$
  $\theta$ = *angle between* $(P_i, P_j)$
  **if** $\theta < \theta_{thres}$ :
    $C_{ij} = true,\ C_{ji} = true$
**for** $k$, $\forall C_{ij}$ ($C_{ij}$ *is true*)
  *image* $I_i$, *image* $I_j$ = $k$
  **if** *3D model or point cloud is provided*
    $C_{ij}$ = *Point cloud projection test* $(X, I_i, I_j, P_1, P_2)$
  **if** *depth map is provided*
    $C_{ij}$ = *Depth map projection test* $(D_i, D_j, P_1, P_2)$
  **else**
    $C_{ij}$ = *Hand crafted overlap test* $(I_i, I_j)$
**for** $k$, $\forall C_{ij}$ ($C_{ij}$ *is true*)
  *generate batch images of index* $i,\ j = k$

### ii Network architecture

The proposed network architecture, illustrated in Fig. 2a, comprises two Siamese absolute pose regressors and one relative pose

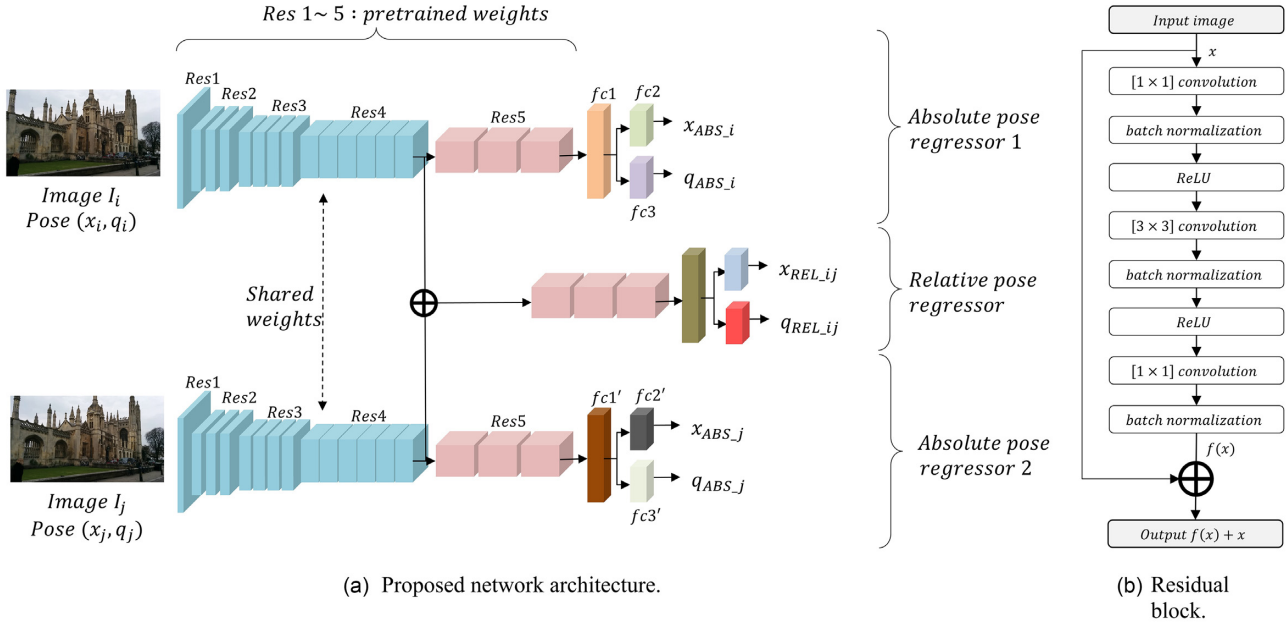



**Figure 2:** Illustration of the proposed network architecture.

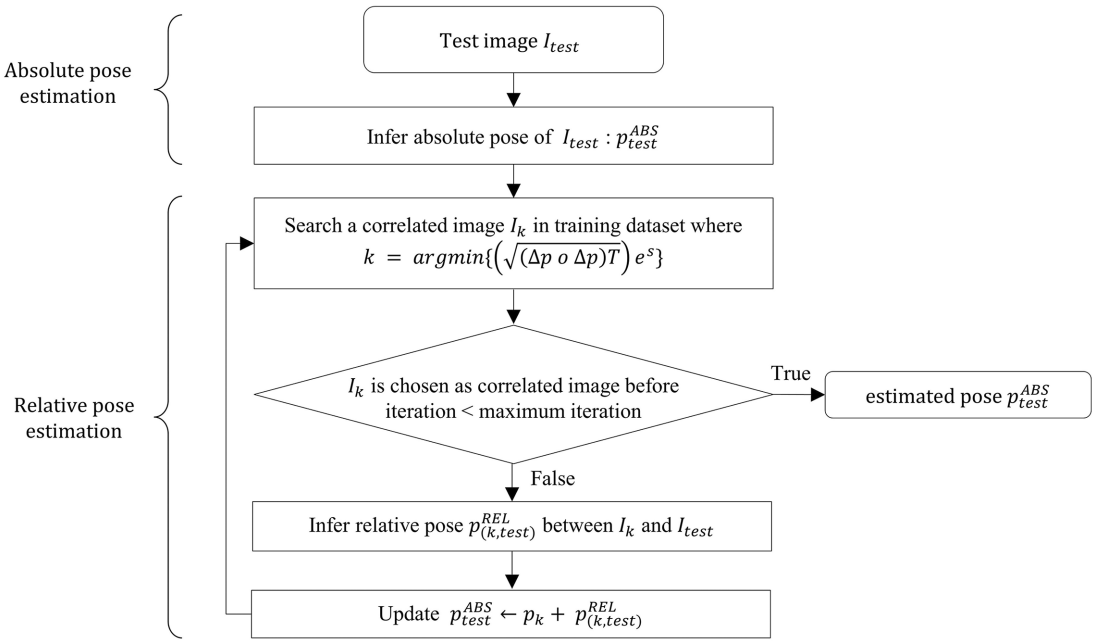
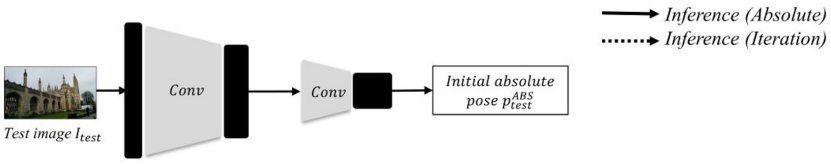


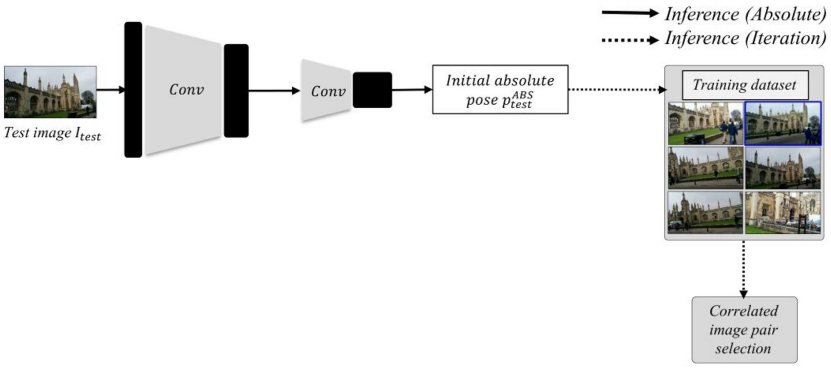

**Figure 3:** Flowchart of the proposed network: Inference (Equations 9–13).

regressor. The two absolute pose regressors were trained with two input images, $I_i$ and $I_j$, and the absolute ground-truth poses, ($x_i$, $q_i$) and ($x_j$, $q_j$). Simultaneously, the relative pose regressor was trained with the input image features and the relative pose between them.

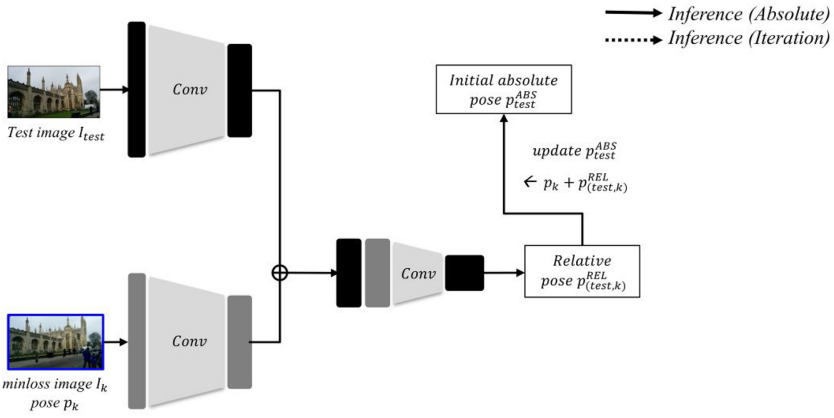
One of the absolute pose regressors (absolute pose regressor 1) uses the following structure: ResNet-50 (He *et al.*, 2016) was selected as the backbone network that extracts the features of an image using a CNN. It comprises five blocks with multiple residual blocks and includes [1 × 1], [3 × 3], and [1 × 1] convolutions, as illustrated in Fig. 2b. Each convolution was followed by batch normalization and a rectified linear unit (ReLU). ResNet was designed to solve one of the bottlenecks of VGGNet, convolutional layer stacking. Stacking convolutional layers increases the number of training parameters in the network and causes a gradient vanishing problem. ResNet addressed this problem by introducing a residual block to directly bypass the input tensor to the output tensor. It exhibited a good performance in object classification, which means that it worked as a good feature extractor. Three FCLs were added to the end of the final residual block (Res 5 in Fig. 2a) to regress the absolute pose of the image. Here, $fc1$ is a regressor of feature size 2048, $fc2$ is a regressor of feature size 3 that regresses the absolute location of an image, and $fc3$ is a regressor of feature size 3 that regresses the absolute orientation of an




(a) Inference of the absolute pose of $I_{test}$ : $p_{test}^{ABS}$.




(b) Search for correlated image $I_k$ in the training dataset.




(c) Infer the relative pose $p_{(test,k)}^{REL}$ between $I_k$ and $I_{test}$; update $p_{test}^{ABS} \leftarrow p_k + p_{(test,k)}^{REL}$.

**Figure 4:** Flowchart of the inference procedure.

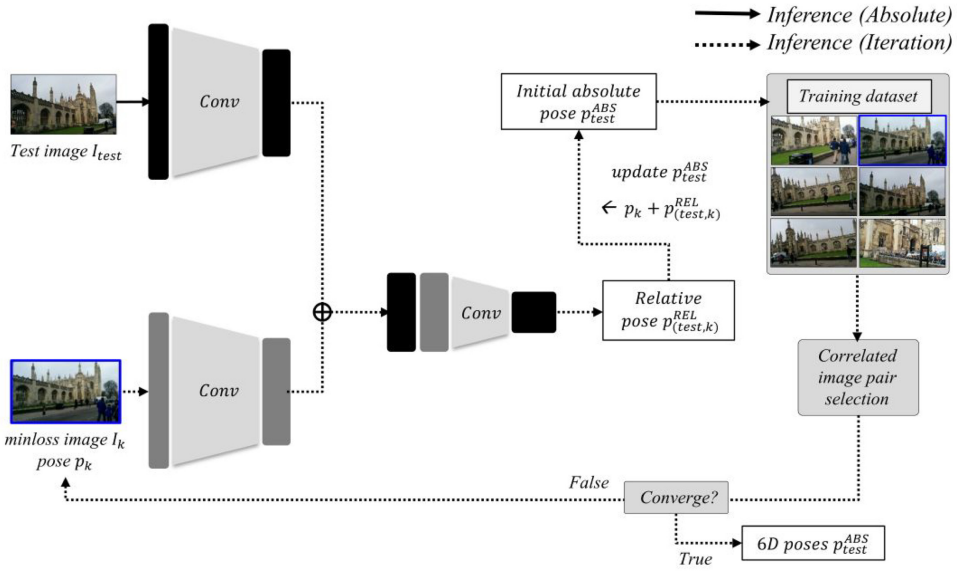
image. The pre-trained weights of ResNet-50 were used for feature extraction.

In this paper, the Siamese network concept for absolute pose estimation is proposed. The same architecture as that of absolute pose regressor 1 was used for the second absolute pose regressor, as shown in Fig. 2a. In addition, the same weights were used for the feature extractor of the second regressor because the estimation results were not affected by the feature extractor. Conceptually, it is possible to employ different kinds of network architecture for the absolute pose regressors. However, using the same architecture is recommended to make the relative pose regressor work consistently by minimizing any unanticipated effects that may be caused by the use of different network structures.

Using the Siamese network architecture, the absolute poses of the two input images were simultaneously trained (absolute pose regressors 1 and 2). The regressor of the second network has the same structure as that of the first network. However, this results in a different set of weights, because the input images and their poses assigned to the first and second regressors are different.

The Siamese network architecture was proposed to train the absolute poses of two images and the relative pose between them during the training stage. The relative pose regressor was designed as follows. The features extracted by each absolute pose regressor were concatenated along a specified dimension (in our case, the depth of the feature maps). The two extracted feature maps exhibited the same dimensions, because the same network structures were used in the absolute pose regressors. Three FCLs were added to the relative pose regressor to train the relative location and orientation between the two input images. All FCLs were followed by the ReLU, except for the final FCLs that were connected directly to the location and orientation.

Novel loss functions were designed for the proposed network. Equation (2) that PoseNet proposed expresses the ordinary loss function $L_{\beta,\mathrm{ABS}}(\mathrm{I_i})$ and was used to regress the single absolute




(d) Repeat relative pose estimation during iteration.

**Figure 4:** Continued.

pose of an input image, where $\gamma$ refers to $L_{\gamma-\text{norm}}$. $\gamma = 2$ was applied, which is the Euclidean norm between the two vectors. It has a user-selected hyperparameter $\beta$ that balances its location and orientation. In this paper, two absolute pose regressors and one relative pose regressor were used. Therefore, more hyperparameters are required to operate them appropriately; however, using many hyperparameters is inefficient. Hence, a loss function with trainable parameters was selected for the entire network. Kendall and Cipolla (2017) proposed a loss function for an absolute pose regressor with trainable parameters that balance location and orientation, as expressed in Equation (3). For a single absolute pose regression, Equation (3) can be used instead of Equation (2), where $s_x$ and $s_q$ are learnable/trainable parameters, beginning from the initial values selected by the user. The learnable parameters were optimized during the training stage to balance location and orientation.

$$L_{\beta,\text{ABS}}(I_i) = \|x - x'\|_\gamma + \beta\left\|q - \frac{q'}{\|q'\|}\right\|_\gamma = L_x(I_i) + \beta L_q(I_i) \tag{2}$$

$$\begin{aligned} L_{s,\text{ABS}}(I_i) &= e^{-s_x}\|x - x'\|_\gamma + s_x + e^{-s_q}\left\|q - \frac{q'}{\|q'\|}\right\|_\gamma + s_q \\ &= e^{-s_x}L_x(I_i) + s_x + e^{-s_q}L_q(I_i) + s_q \end{aligned} \tag{3}$$

$$\begin{aligned} L_{s,\text{REL}}(I_i, I_j) &= e^{-s_x}\|\Delta x - \Delta x'\|_\gamma + s_x + e^{-s_q}\|\Delta q - \Delta q'\|_\gamma + s_q \\ &= e^{-s_x}L_x(I_i, I_j) + s_x + e^{-s_q}L_q(I_i, I_j) + s_q \end{aligned} \tag{4}$$

$$\text{where } \Delta x = x_i - x_j,\ \Delta q = \frac{q_i}{\|q_i\|} - \frac{q_j}{\|q_j\|}$$

$$L = L_{s,\text{ABS}}(I_i) + L_{s,\text{ABS}}(I_j) + L_{s,\text{REL}}(I_i, I_j). \tag{5}$$

Equation (4) expresses the loss function used to train the relative pose between $I_i$ and $I_j$. $L_{\text{ABS}}$ and $L_{\text{REL}}$ were integrated to train two absolute poses and one relative pose between two images, as shown in Equation (5), which is the final loss function used during the training. The orientation is represented by the logarithm of a unit quaternion, which is a 3D vector. Because unit quaternion expresses orientation in four DoFs, it is overparametrized compared to Euler angles that express orientation in three DoFs. The logarithm of a unit quaternion is applied to the loss function of Equations (2)–(5) (Brahmbhatt *et al.*, 2018). A quaternion is defined as a tuple of scalar component $u$ and 3D vector $\mathbf{v}$ as shown in Equation (6). The unit quaternion $q$ was converted into the logarithm of $q$, as shown in Equation (7). Conversely, the exponential of $w$ becomes $q$ as expressed in Equation (8). Using Equation (7), the rotation can be regressed using a 3D vector that is parametrized well to represent the orientation in three DoFs, such as Euler angles. In this paper, for all expressions associated with orientation in implementation, the logarithm of a unit quaternion $w$ was used instead of unit quaternion $q$.

$$q(u, x, y, z) = (u, \mathbf{v}) \tag{6}$$

$$w = \log q = \begin{cases} \frac{\mathbf{v}}{\|\mathbf{v}\|}\cos^{-1}u, & if\ \|\mathbf{v}\| \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$$\exp(w) = \left(\cos\|w\|, \frac{w}{\|w\|}\sin\|w\|\right) = q. \tag{8}$$

## 2.2 Inference stage

In the proposed method, the absolute pose (initial guess) of an input image was first estimated, followed by iterative compensation of the estimated pose using a relative pose regressor, as illustrated in Fig. 3. The inference process is illustrated in Figs 3 and 4. From a single-input query test image $I_{\text{test}}$, the absolute pose of an image $p^{\text{ABS}}_{\text{test}}$ was estimated using the absolute pose regressor, which is based on ResNet-50 as a backbone attached to the FCLs to regress the location and orientation (absolute pose regressor 1), as illustrated in Fig. 4a. Subsequently, one image was selected as the network input (Fig. 4b) to estimate the relative pose. The images were obtained from the training dataset. Here, the loss function for relative pose estimation was used to obtain a correlated image from the training dataset, as expressed in Equation (4). After training was performed, the trainable parameters $s_x$ and $s_q$ were set to specific numerical values, to minimize the loss function.

**Figure 5:** Sample images from Cambridge Landmarks (King's College) and TOD datasets.

$$\hat{p}_{\text{train}} = \begin{bmatrix} x_1 & y_1 & z_1 & w_{x1} & w_{y1} & w_{z1} \\ & \vdots & & & \vdots & \\ x_k & y_k & z_k & w_{xk} & w_{yk} & w_{zk} \\ & \vdots & & & \vdots & \\ x_n & y_n & z_n & w_{xn} & w_{yn} & z_{zn} \end{bmatrix} \tag{9}$$

$$\hat{p}^{\text{ABS}}_{\text{test}} = \begin{bmatrix} x^{\text{ABS}}_{\text{test}} & y^{\text{ABS}}_{\text{test}} & z^{\text{ABS}}_{\text{test}} & w^{\text{ABS}}_{\text{x,test}} & w^{\text{ABS}}_{\text{y,test}} & w^{\text{ABS}}_{\text{z,test}} \\ & \vdots & & & \vdots & \\ x^{\text{ABS}}_{\text{test}} & y^{\text{ABS}}_{\text{test}} & z^{\text{ABS}}_{\text{test}} & w^{\text{ABS}}_{\text{x,test}} & w^{\text{ABS}}_{\text{y,test}} & w^{\text{ABS}}_{\text{z,test}} \end{bmatrix} \tag{10}$$

$$\Delta p = \hat{p}_{\text{train}} - \hat{p}^{\text{ABS}}_{\text{test}} \tag{11}$$

$$k = argmin \left\{ \left( \sqrt{(\Delta p \, o \, \Delta p) \, T} \right) e^{s} \right\} \tag{12}$$

$$\text{where } (A \, o \, B)_{ij} = (A)_{ij} (B)_{ij}$$

$$T = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^{T}$$

$$e^{s} = \begin{bmatrix} e^{s_x} & e^{s_q} \end{bmatrix}^{T}.$$

The differences between the test image and the ground-truth poses of all the training images were calculated using Equations (9)–(12). In addition, using the trainable parameters $s_x$ and $s_q$ and Equation (12), an image in the training dataset that had the minimum loss between the test images was selected as the input to the relative pose estimation. Equation (12) determined the index of the matrix that represents the relative losses between the images using Equation (4), where each row corresponds to $L_{s,\text{REL}}(I_{\text{Test}}, I_i) - s_x - s_q, i \in (1, 2, \ldots, n)$. The addition of $s_x$ and $s_q$ was not calculated from $L_{s,\text{REL}}(I_{\text{Test}}, I_i)$ in Equation (12) because $s_x$ and $s_q$ are constant after the training was completed. Hence, it is not necessary to add the same values to each row.

A total of $n$ relative losses were calculated, and the image $I_k$ that had the minimum loss value was selected. Subsequently, $I_k$ and $I_{\text{test}}$ were set as inputs to the entire network, and the relative pose $p^{\text{REL}}_{(\text{test},k)}$ between them was obtained. Thus far, only absolute pose regressor 1, front feature extractor of absolute pose regressor 2, and relative pose regressor have been used. The pose of image $I_k$ was not estimated iteratively because its ground-truth pose $p_k$ was already known. After estimating $p^{\text{REL}}_{(\text{test},k)}$, the absolute pose of the test image was replaced and updated as follows:

$$p^{\text{ABS}}_{\text{test}} \leftarrow p_k + p^{\text{REL}}_{(\text{test},k)}. \tag{13}$$

This is illustrated in Fig. 4c. Here, not all training dataset images were used to infer the relative pose of all images as input to the CNN. Considering all training images is time consuming. Therefore, one relative pose was estimated between the query input image and the image with minimum loss value. Searching for an image with minimum loss requires a simple matrix calculation, and such an image can be identified easily.

Finally, the relative pose estimation was repeated to compensate for the inference result, as illustrated in Fig. 4d, until the estimation reached the maximum number of iterations or converged. Using a query input test image, absolute pose regressor 1 was used to estimate the initial absolute pose of the input query image. After estimating the absolute pose of the test image $p^{\text{ABS}}_{\text{test}}$, the minimum loss image $I_k$ was searched using the final fixed trainable parameters $s_x$ and $s_q$ from the pose list of the training dataset $\hat{p}_{\text{train}}$. In addition, the relative pose $p^{\text{REL}}_{(\text{test},k)}$ was estimated using images $I_{\text{test}}$ and $I_k$. Subsequently, $p^{\text{ABS}}_{\text{test}}$ was updated to $p_k + p^{\text{REL}}_{(\text{test},k)}$. This process was repeated, except for the absolute pose estimation. The absolute pose estimation was not performed twice because it always exhibits the same absolute pose if the input image $I_{\text{test}}$ remains not changed. Using the updated $p^{\text{ABS}}_{\text{test}}$, a new minimum-loss image and the relative pose between the two images can be obtained. This iterative relative pose estimation process can be repeated until the correlated image was selected in advance or the number of iterations exceeded the maximum threshold. It is unnecessary to estimate the relative pose between an already selected training image with minimum loss and an input query test image, because the relative pose between them is already known. The pose was updated after several iterations, and it was assumed that the final compensated absolute pose of the input query image was $p^{\text{ABS}}_{\text{test}}$, which was the pose obtained from the final iteration.

## 3 Experimental Results and Discussion

### 3.1 Dataset

The Cambridge Landmarks dataset that Kendall *et al.* (2015) used to validate the localization accuracy of PoseNet contains outdoor images with ground-truth locations and orientations. The images were captured using a smartphone, and SfM was applied to generate the ground-truth labels (location and orientation) of the images. Urban clutter, such as pedestrians and vehicles, were present; in addition, occlusions, rapid changes in motion, and changes in illumination and weather were included. They were classified into the training and test set. Five datasets were obtained from different locations. The number of training dataset images varied between [231–1487] and [103–2923]. The images exhibited a full HD resolution [1920 px × 1080 px].

The second dataset, called the TOD dataset, contains indoor images of a camera device, as illustrated in Fig. 5. RGB images

**Table 1:** Details of Cambridge Landmarks and TOD datasets.

| Dataset | Scene | # of frames | | Spatial extent (m) |
|---|---|---|---|---|
| | | Train | Test | |
| Cambridge Landmark | King's College | 1220 | 343 | 140 × 40 |
| | Old Hospital | 895 | 182 | 50 × 40 |
| | Shop Facade | 231 | 103 | 35 × 25 |
| | St Mary's Church | 1487 | 530 | 80 × 60 |
| TOD | Scene1 (3:1) | 1677 | 559 | 5 × 5 |
| | Scene2 (2:1) | 1490 | 746 | |
| | Scene3 (1:1) | 1118 | 1118 | |
| | Scene4 (1:2) | 746 | 1490 | |
| | Scene5 (1:3) | 559 | 1677 | |

**Table 2:** Localization error comparison between Cambridge Landmarks dataset to state-of-the-arts.

| Models | King's College | Shop Facade | Old Hospital | St Mary's Church | Average |
|---|---|---|---|---|---|
| PoseNet (Kendall *et al.*, 2015) | 1.92 m, 5.40° | 1.46 m, 8.08° | 2.31 m, 5.38° | 2.65 m, 8.48° | 2.08 m, 6.83° |
| PoseNet++ (Zhang *et al.*, 2018) | 1.58 m, 2.38° | 1.10 m, 5.61° | | | 1.34 m, 3.99° |
| Bayesian PoseNet (Kendall & Cipolla, 2016) | 1.74 m, 4.06° | 1.25 m, 7.54° | 2.57 m, 5.14° | 2.11 m, 8.38° | 1.91 m, 6.28° |
| PoseNet Spatial LSTM (Walch *et al.*, 2017) | 0.99 m, 3.65° | 1.18 m, 7.44° | 1.51 m, 4.29° | 1.52 m, 6.68° | 1.30 m, 5.51° |
| GeoPoseNet (Kendall & Cipolla, 2017) | 0.99 m, **1.06**° | 1.05 m, 3.97° | 2.17 m, **2.94**° | 1.49 m, **3.43**° | 1.42 m, **2.85**° |
| MapNet (Brahmbhatt *et al.*, 2018) | 1.07 m, 1.89° | 1.49 m, 4.22° | 1.94 m, 3.91° | 2.00 m, 4.53° | 1.62 m, 3.63° |
| TransPoseNet (Shavit *et al.*, 2021) | 0.60 m, 2.43° | 0.55 m, 3.49° | 1.45 m, 3.08° | 1.09 m, 4.99° | 0.92 m, 3.49° |
| VNLSTM PoseNet (Li *et al.*, 2021) | 1.71 m, 4.77° | | **0.89 m**, 3.83° | | 1.30 m, 4.30° |
| IPRNet (Shavit & Ferens, 2021) | 1.18 m, 2.19° | 0.72 m, **3.47**° | 1.87 m, 3.38° | 1.87 m, 4.94° | 1.42 m, 3.45° |
| **Proposed** | **0.45 m**, 2.22° | **0.37 m**, 3.53° | 1.12 m, 3.38° | **0.58 m**, 5.17° | **0.63 m**, 3.58° |

were captured using a Galaxy S20+ smartphone in video form, and the images were extracted from each frame. SfM was used to generate ground-truth labels, similar to the Cambridge Landmarks dataset. The entire images were classified into the training and test datasets. The training dataset was used to train the proposed architecture, and the test dataset was used for validation. A summary of the datasets is presented in Table 1. Each dataset was classified into several scenes. The scenes had completely different training and test images. The spatial extent indicates the ranges of the *x* and *y* directions of the datasets. Here, the range of the *z* values of the datasets is negligible compared to those of the *x* and *y* directions.

## 3.2 Implementation

The input images were scaled such that the shorter side of each image had 256 pixels, while the width and height ratios of the original input images were maintained. Images of [224 px × 224 px] pixels cropped randomly from the scaled input images were provided as input to the network for training. Images cropped at the centre of the scaled images were utilized for inference. The mean image of the cropped images was computed with the standard deviation. The mean image was subtracted from each cropped image to obtain intermediate images, which were then divided by standard deviation. The final images were used for training and inference. This process is known as regularization or normalization and is widely applied in neural network applications. PyTorch (Paszke *et al.*, 2019) was used to implement the proposed network architecture. An Adam optimizer (Kingma & Ba, 2014) was used to train the proposed network with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and learning rate $lr = 10^{-3}$. A mini-batch size of 30 and a dropout probability of 0.5 were used. In addition, $s_x$ and $s_q$ were initialized to 0 and −3, respectively. The network was trained for 300 epochs for every scene and a single NVIDIA GeForce 2080Ti graphics card was used for training and inference.

## 3.3 Experimental results

A summary of the localization results for the Cambridge Landmarks dataset is provided in Table 2. The best localization results for the location and orientation are presented in bold for clarity. The training dataset was used to train the proposed network, which was subsequently validated using a test dataset that was not included in the training dataset. The location error was calculated using the root-mean-square error, and the orientation error was calculated by averaging the angular difference between the estimated and ground-truth quaternions using Equation (1). The proposed method provided a high localization accuracy for position and a comparable accuracy for orientation compared to other state-of-the-art approaches.

Figures 6 and 7 present the localization results for the training and test datasets of the King's College scene in the Cambridge Landmarks dataset, respectively. Because only a small number of changes existed in the *z* values of the Cambridge Landmarks dataset, the top-view images in the *xy* plane were used to visualize the results. Each axis corresponds to the positional range of the scene on the *x*- and *y*-axes. The test results indicate that the proposed method yielded high localization results, particularly for the training dataset.

Figure 8 presents the localization errors for position and orientation as a cumulative histogram for the test dataset of the King's College scene. The *x*-axes of the figures represent the positional and angular errors of the localization results. The *y*-axis represents the number of images as a percentage. The more the graph is skewed to the upper-left side, the more images have small errors, leading to better localization accuracy. The proposed method yielded the best localization results.

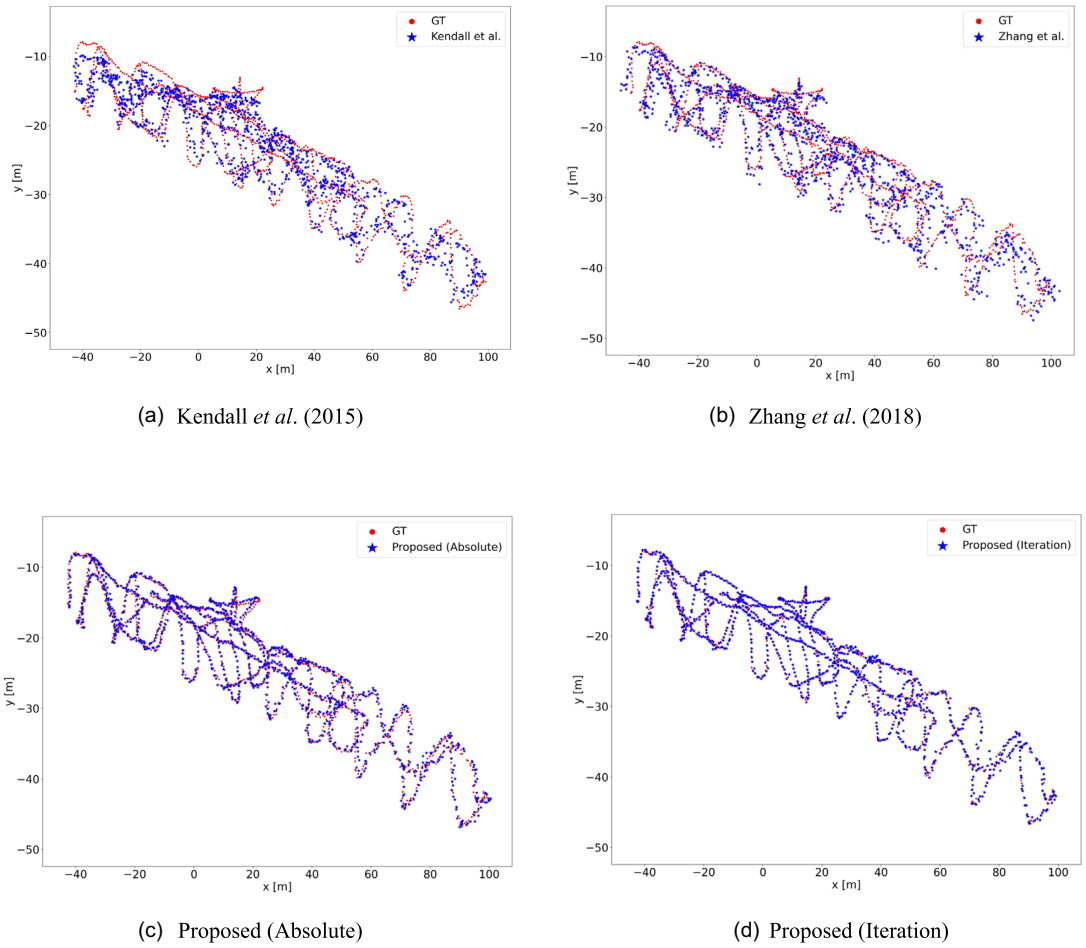


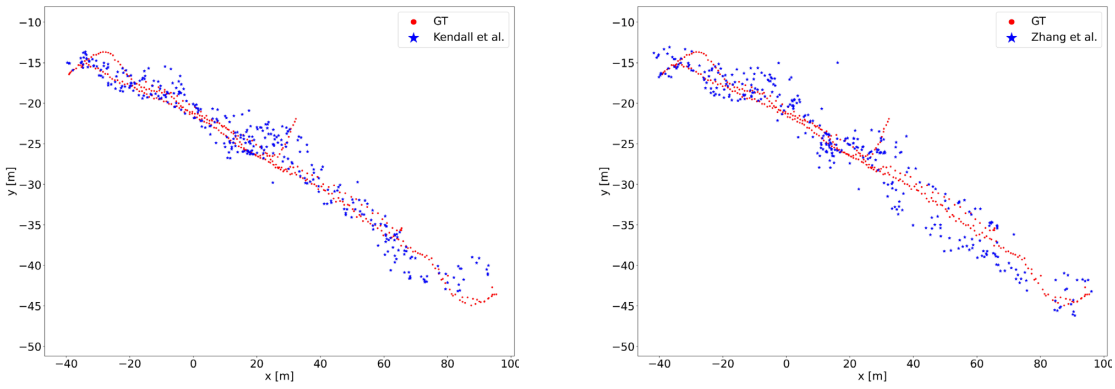

**Figure 6:** Localization results for King's College scene in Cambridge Landmarks training dataset.

Figure 9 illustrates the cumulative histograms of the localization errors for the training dataset of the King's College scene. The localization performance was evaluated for cases with and without CIPS during the training stage. If the CIPS was applied during the training stage, an image in the training dataset was selected first, and one of its correlated images was chosen as the input to the proposed network architecture. The total $n$ training images $I_{\text{train}} = \{I_1, I_2, ..., I_n\}$ exists and the correlated images of the test image are $I_{\mathbb{C}} = \{I_b^1, I_b^2, \ldots, I_b^{\mathbb{C}}\} \subseteq I_{\text{train}}$, where the number of correlated images is $\mathbb{C}$ for $I_{\text{test}}$. One of the correlated images was chosen as a batch image out of $\mathbb{C}$. Thus, for a test image $I_{\text{test}}$, there are $\mathbb{C}$ cases to select as batch image pairs for the proposed Siamese network $\{(I_{\text{test}}, I_b^1), (I_{\text{test}}, I_b^2), \ldots, (I_{\text{test}}, I_b^{\mathbb{C}})\}$. If the CIPS was not applied during training, a random image in the training dataset was selected as the batch image. There are $n$ cases $\{(I_{\text{test}}, I_1), (I_{\text{test}}, I_2), \ldots, (I_{\text{test}}, I_n)\}$. Obviously, $\mathbb{C} \ll n$, if the proposed network was trained with CIPS, the correlated image pairs were trained intensively. If CIPS was not applied, random image pairs were trained sparsely compared to the case of training without applying CIPS. Thus, to obtain similar training results for the correlated image pairs, more training epochs are required if the CIPS is not applied. Figure 9 shows the effect of applying the CIPS in training. If the CIPS is not applied during training, relative pose training with correlated image pairs is not sufficiently performed, and the inference localization results are adversely affected.

The proposed method was compared to other approaches using the TOD dataset. The localization errors are summarized in Table 3, which demonstrates that the proposed method yields the best localization results. Figure 10 presents AR examples of the TOD dataset. Equation (14) was used to augment the 3D point cloud for the captured image, where the point cloud was obtained from SfM. It was used to project world coordinate-based 3D points onto 2D images with a specific camera pose. In addition, it was used to calculate the re-projection error (Kendall & Cipolla, 2017). The estimated orientation of an input image expressed in the quaternion was converted to a rotation matrix, an extrinsic matrix $E$; constructed, and Equation (14) was obtained.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = KEX = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{14}$$

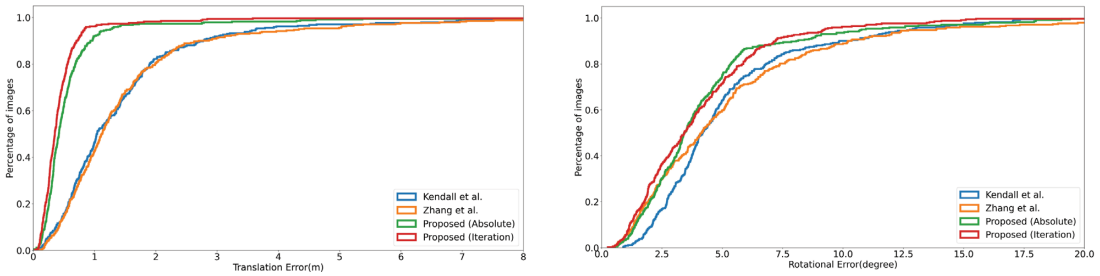
Because the intrinsic camera matrix $K$ was provided, only the extrinsic matrix $E$ was required to augment the point cloud. Because the proposed approach provided better localization results than the other methods, it also achieved better augmentation results. The proposed method yielded higher accuracy localization results for different dataset spacings, as presented in Table 3. The performance of the proposed method was evaluated using different numbers of training images. Our experimental results indicate that the proposed method performed best when a large number of training images were used, i.e. Scene1 (3:1), where the number




(a) Kendall *et al*. (2015) (b) Zhang *et al*. (2018)

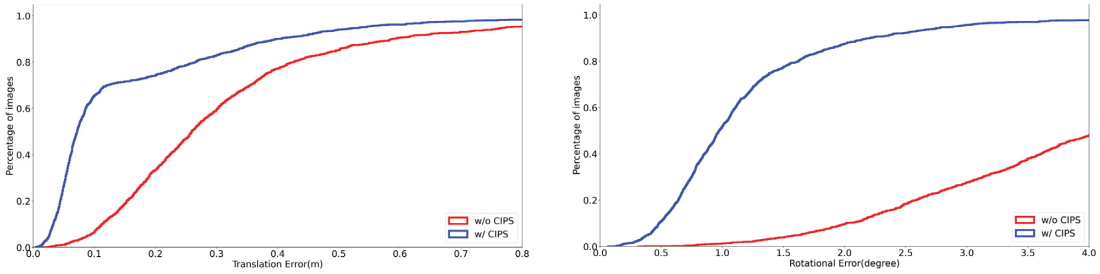
(c) Proposed (Absolute) (d) Proposed (Iteration)

**Figure 7:** Localization results for the King's College scene in the Cambridge Landmarks test dataset.




(a) Positional error (b) Rotational error

**Figure 8:** Cumulative histograms of the location and orientation errors of the King's College scene in the Cambridge Landmarks test dataset.

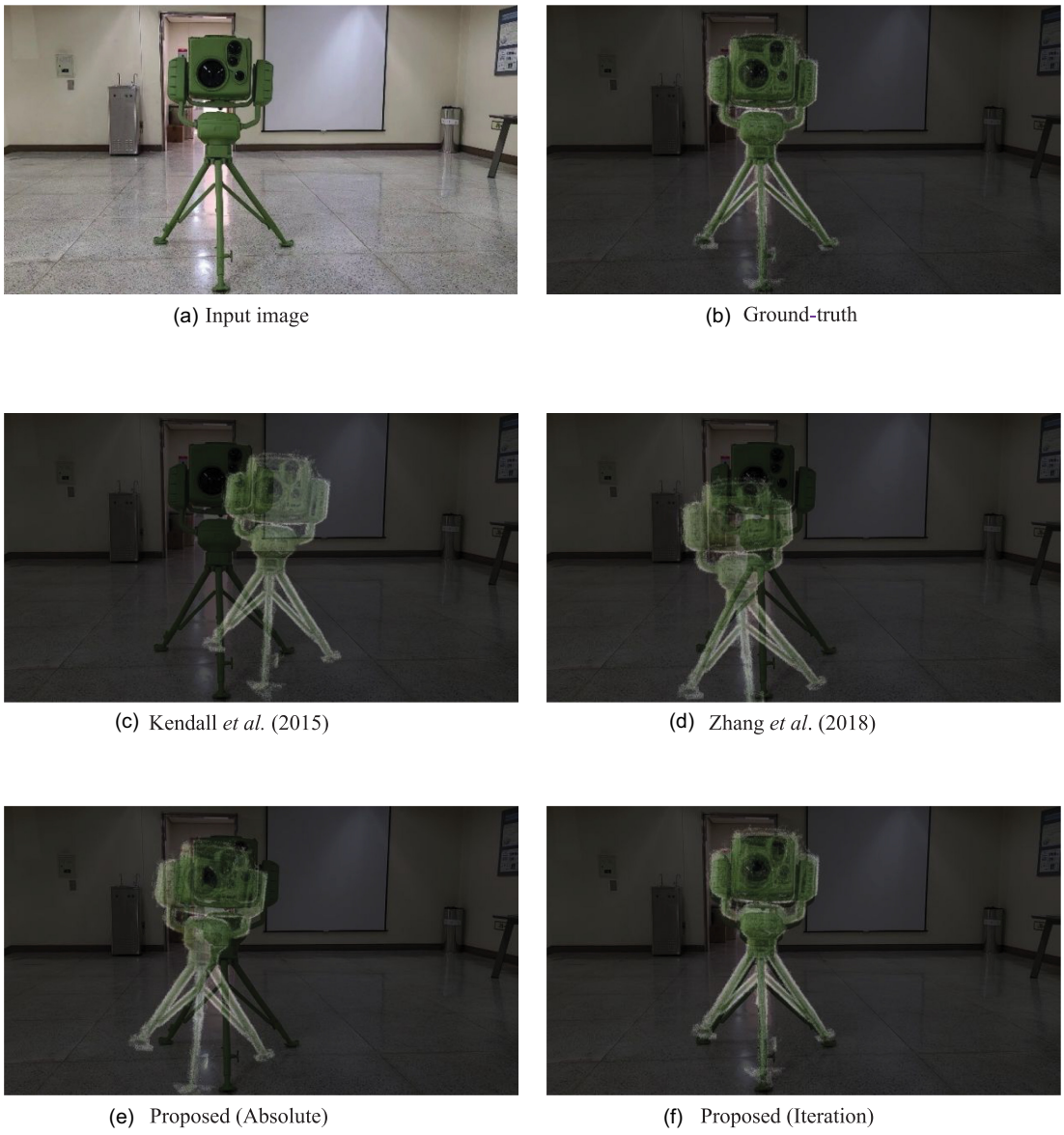



(a) Positional error (b) Rotational error

**Figure 9:** Cumulative histograms of the location and orientation errors of the King's College scene in the Cambridge Landmarks training dataset.

**Table 3:** Localization error comparison of TOD dataset.

| Models | Scene1 (3:1) | Scene2 (2:1) | Scene3 (1:1) | Scene4 (1:2) | Scene5 (1:3) | Average |
|---|---|---|---|---|---|---|
| PoseNet (Kendall *et al.*, 2015) | 0.097 m, 1.554° | 0.105 m, 2.250° | 0.061 m, 1.253° | 0.084 m, 1.412° | 0.095 m, **1.311**° | 0.088 m, 1.550° |
| PoseNet++ (Zhang *et al.*, 2018) | 0.077 m, 1.234° | 0.084 m, 1.469° | 0.067 m, 0.781° | 0.069 m, 1.015° | 0.081 m, 1.705° | 0.075 m, 1.240° |
| Proposed (absolute) | 0.073 m, 1.206° | 0.077 m, 6.850° | 0.071 m, 0.840° | 0.042 m, 1.212° | **0.717 m**, 5.881° | 0.196 m, 3.428° |
| Proposed (iteration) | **0.040 m, 0.962**° | **0.044 m, 1.239**° | **0.059 m, 0.671**° | **0.038 m, 0.984**° | 0.076 m, 1.401° | **0.051 m, 1.167**° |



(a) Input image (b) Ground-truth

(c) Kendall *et al.* (2015) (d) Zhang *et al.* (2018)

(e) Proposed (Absolute) (f) Proposed (Iteration)

**Figure 10:** AR results of projecting point clouds onto image using estimated localization results for TOD dataset.

of training images was three times larger than that of the test images.

The convergence of the iterative relative pose estimation for pose inference and the performance of the iterative relative pose estimation for inference were validated using the TOD dataset. As presented in Table 4, inferences were performed while changing the number of maximum iterations using Scene 1 of the TOD dataset up to five. In the case of Scene 1, convergence was achieved within five iterations for 99% of the test images. Less than 1% of the test images required more than five iterations for convergence.

The possibility of reducing the localization error was investigated using relative and absolute poses for training. Pose inference using only the absolute pose regressor is equivalent to setting the number of maximum iterations to zero, which is presented by 'Proposed (absolute)' in Table 4, while

**Table 4:** Convergence validation of iterative relative pose estimation (Scene1).

| Method | Maximum # of iteration | # of iteration(converged) : # of images/percentage | | | | | Average error |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | |
| PoseNet | | | | | | | 0.097 m, 1.554° |
| PoseNet++ | | | | | | | 0.077 m, 1.234° |
| Proposed (absolute) | (0) | | | | | | 0.073 m, 1.206° |
| Proposed (iteration) | 1 | 559 (100%) | | | | | 0.047 m, 0.967° |
| | 2 | 498 (89%) | 61 (11%) | | | | 0.041 m, 0.966° |
| | 3 | 498 (89%) | 34 (6%) | 27 (5%) | | | 0.041 m, 0.963° |
| | 4 | 498 (89%) | 34 (6%) | 17 (3%) | 10 (2%) | | 0.040 m, 0.963° |
| | 5 | 498 (89%) | 34 (6%) | 17 (3%) | 4 (1%) | 6 (1%) | **0.040 m, 0.962°** |

using both the absolute and relative poses during training. In this case, only the absolute pose was used for inference, which indicates a localization error of (0.073 m, 1.206°), and is smaller than that reported in previous studies that used only the absolute pose for training. Next, the performance of the relative pose estimation during both training and inference was validated, and the results are shown in Fig. 4c. Both the absolute and relative poses for training were used, and the relative pose estimation during inference was used once. This is equivalent to setting the maximum number of iterations to one. Absolute and relative pose regressors were used once for all test images (559). They provided a more accurate result (0.047 m, 0.967°) than when only the absolute pose regressor was used during the inference (0.073 m, 1.206°). This indicates that using relative pose estimation for inference yields better results than using it only for training. Finally, iterative relative pose estimation for inference was evaluated. While changing the number of maximum iterations to five, the number of test images that converged was counted, which corresponds to Fig. 4d. For most test images (89%), the proposed method converged in only one relative pose estimation. For the other images (11%), more accurate localization results were obtained by increasing the number of maximum iterations for inference. Finally, the proposed method obtained the best localization results of 0.040 m and 0.962° via iterations. The test results confirmed that the relative pose estimation for both the training and inference in the iteration loop provided more accurate localization results than when it was not used.

The proposed method yielded better localization results than existing methods for the Cambridge Landmarks and TOD datasets used to validate the proposed method. This is because deep learning-based camera localization is based on image-retrieval. Image-retrieval measures the similarity between two images and expresses the similarity as a value. However, image-retrieval-based methods require images and their features to store the entire dataset. In addition, the nearest neighbour or the most similar image must be identified. Accordingly, handcrafted features are used, and the images must be captured under similar conditions. However, deep learning-based camera localization methods circumvent these limitations by applying high-dimensional feature extraction and matching them using a CNN, and then evading the nearest search process by modelling the image to pose mapping directly, which implies image-retrieval. In pose estimation methods, the absolute pose regressor is used to perform this process automatically, using an encoder (CNN) as the feature extractor and FCLs (regressors) to perform the nearest search process. Because it does not use two image pairs as inputs, it is an indirect image-retrieval method. Relative pose estimation using two image pairs is exactly the same as image-retrieval because it employs one query image and one of the dataset images, and presents the similarity between them as values (poses). Hence, the deep learning-based relative camera pose estimation method can be regarded as a direct image-retrieval approach. The proposed method employs image-retrieval using deep learning-based relative pose estimation during both training and inference.

An ordinary deep-pose regressor utilizes only one image. Brahmbhatt *et al.* (2018) and Valada and Burgard (2018) added additional regressors to the backbone network to estimate the relative pose between two images, thereby increasing the effect of image-retrieval during the training. However, these methods are indirect image-retrieval methods. During inference, only the absolute pose regressor was used, and the image-retrieval was not activated directly. Although this intensifies the effect of image-retrieval during training, only one image is required to estimate its pose during inference. During inference, it did not significantly different from that of the absolute pose regressors. Compared with the ordinary deep pose regressor, the proposed method uses two images as inputs to the network, directly utilizes the image-retrieval effect, and provides more accurate localization results during training and inference.

However, this was limited to location estimation only in our experiments for the Cambridge Landmarks dataset. Although the proposed method offers stable location estimation, it does not demonstrate the best performance in terms of orientation estimation. To improve the orientation accuracy, various sets of images must be used. Each set should contain images captured at positions with different orientations. Because the Cambridge Landmarks dataset was constructed along the route by changing the orientation of the camera, it provided insufficient training images for the proposed method, which showed the same position for different orientations.

The proposed method provides more reliable results for the TOD dataset than for the Cambridge Landmarks dataset. The TOD dataset provides sequential images as input data for a deep neural network (sharing a similar route). Hence, correlated images that are the most similar in the dataset can be obtained easily using the proposed method. However, the Cambridge Landmarks dataset does not provide sequential data as an input. It provides only partial regions of the routes and cannot easily identify the image most similar to the query input image.

A summary of the average computational times for pose inference using the various methods is presented in Table 5. PoseNet, which used GoogLeNet consisting of 23 convolutional layers with 4M trainable parameters, showed the fastest inference speed. Compared to PoseNet, PoseNet++ used VGG-16 with 138M trainable parameters. Owing to the increased number of parameters, it exhibited a slower speed than PoseNet. The proposed network uses ResNet-50 as a backbone network consisting of 50 convolutional layers. Owing to the existence of a residual block, a smaller

**Table 5:** Backbone network comparisons with the computational times for single image inference.

| | PoseNet | PoseNet++ | Proposed (absolute) | Proposed (iteration) |
|---|---|---|---|---|
| Backbone network | GoogLeNet | VGG-16 | ResNet-50 | Siamese ResNet-50 |
| The number of convolution layers | 23 | 13 | 50 | 50 |
| The number of trainable parameters | 4M | 138M | 23M | 46M |
| Average computational time | 4 ms | 45 ms | 26 ms | 109 ms |

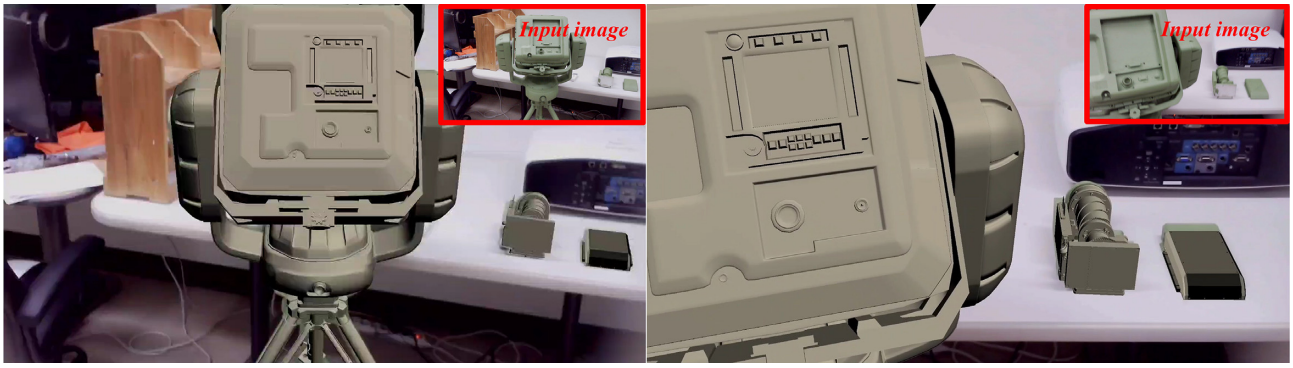



**Figure 11:** Examples of AR application.

number of trainable parameters was required. Thus, the proposed network inferred the absolute pose faster than PoseNet++ did. However, it ran slower than PoseNet owing to the iterative use of the Siamese network architecture.

## 4 Conclusions and Future Works

In this paper, a deep learning-based camera localization method using iterative relative pose estimation was proposed. The proposed method applies a CNN to regress the poses of images and adopts two Siamese encoders to estimate their absolute poses. In addition, one relative pose regressor is concatenated to the feature maps of the two absolute pose regressors to train the relative pose between the two input images. The proposed network architecture simultaneously uses the absolute poses of the training dataset images and the relative poses of the correlated image pairs. For inference, absolute pose estimation was performed on the test image, and relative pose estimation was applied to update the estimated pose iteratively. The proposed method achieved higher localization accuracy than other state-of-the-art deep learning camera localization methods for the Cambridge Landmarks and TOD datasets. Correlated image-pair selection and relative pose estimation for the inference stage improved localization accuracy.

The proposed method adopted an iterative approach and exhibited a slower inference speed than other methods. However, its computational speed can be improved by applying the concept of a mini-batch during the inference stage, to obtain the relative poses and avoid iterations. Moreover, several relative poses between the images can be estimated simultaneously using a parallel computation scheme. A future endeavour is to refine the absolute pose from the initial guess using the estimated relative poses. The proposed method iteratively performs the refining process, it is possible to apply an average relative pose to refine an absolute pose. RANSAC can be used to exclude outliers. In addition, it is possible to convert the iteration into a one-time inference. Furthermore, the proposed localization method can be applied to AR glasses as illustrated in Fig. 11. A 3D model was augmented on the display of the AR glass using the camera of the AR glass, which demonstrates the potential of the proposed method for use in various AR applications.

## Acknowledgments

This study was supported by the Research Program of Institute of Civil-Military Technology Cooperation through the Defense Acquisition Program Administration of Korea and Ministry of Trade, Industry and Energy, Korea (20-SN-GU-01).

## Conflict of interest statement

None declared.

## References

Balntas, V., Li, S., & Prisacariu, V. (2018). Relocnet: Continuous metric learning relocalisation using neural nets. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Proceedings of the 2018 European Conference on Computer Vision (ECCV)* (pp. 751–767). Springer. https://doi.org/10.1007/978-3-030-01264-9_46.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, **110**, 346–359. https://doi.org/10.1016/j.cviu.2007.09.014.

Brahmbhatt, S., Gu, J., Kim, K., Hays, J., & Kautz, J. (2018). Geometry-aware learning of maps for camera localization. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2616–2625). IEEE. https://doi.org/10.1109/cvpr.2018.00277.

Carlone, L., Kira, Z., Beall, C., Indelman, V., & Dellaert, F. (2014). Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4290–4297). IEEE. https://doi.org/10.1109/icra.2014.6907483.

Drummond, T., & Cipolla, R. (1999). Real-time tracking of complex structures with on-line camera calibration. In *Proceedings of the*

*British Machine Vision Conference (BMVC'99)* (pp. 574–583). Elsevier. https://doi.org/10.1016/S0262-8856(02)00013-6.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)* (pp. 1440–1448). IEEE. https://doi.org/10.1109/iccv.2015.169.

Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, **2**, 31–43. https://doi.org/10.1109/mits.2010.939925.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). IEEE. https://doi.org/10.1109/cvpr.2016.90.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Kendall, A., Grimes, M., & Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)* (pp. 2938–2946). IEEE. https://doi.org/10.1109/iccv.2015.336.

Kendall, A., & Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4762–4769). IEEE. https://doi.org/10.1109/icra.2016.7487679.

Kendall, A., & Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5974–5983). IEEE. https://doi.org/10.1109/cvpr.2017.694.

Kim, D., Park, J., & Ko, K. H. (2018). Development of an AR based method for augmentation of 3D CAD data onto a real ship block image. *Computer-Aided Design*, **98**, 1–11. https://doi.org/10.1016/j.cad.2017.12.003.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint* arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems* (pp. 1097–1105). Massachusetts Institute of Technology Press. https://doi.org/10.1145/3065386.

Laskar, Z., Melekhov, I., Kalia, S., & Kannala, J. (2017). Camera relocalization by computing pairwise relative poses using convolutional neural network. In *Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (pp. 929–938). IEEE. https://doi.org/10.1109/iccvw.2017.113.

Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011). BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision* (pp. 2548–2555). IEEE. https://doi.org/10.1109/ICCV.2011.6126542.

Li, M., Qin, J., Li, D., Chen, R., Liao, X., & Guo, B. (2021). VNLSTM-PoseNet: A novel deep convnet for real-time 6-DoF camera relocalization in urban streets. *Geo-spatial Information Science*, **24**, 422–437. https://doi.org/10.1080/10095020.2021.1960779.

Liu, L., Li, H., & Dai, Y. (2017). Efficient global 2D–3D matching for camera localization in a large-scale 3D map. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 2372–2381). IEEE. https://doi.org/10.1109/ICCV.2017.260.

Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. *Sustainable Cities and Society*, **65**, 102600. https://doi.org/10.1016/j.scs.2020.102600.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 3431–3440). IEEE. https://doi.org/10.1109/cvpr.2015.7298965.

Lourakis, M. I., & Argyros, A. A. (2009). SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, **36**, 1–30. https://doi.org/10.1145/1486525.1486527.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, **60**, 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.

Mahmoodzadeh, A., Mohammadi, M., Noori, K. M. G., Khishe, M., Ibrahim, H. H., Ali, H. F. H., & Abdulhamid, S. N. (2021). Presenting the best prediction model of water inflow into drill and blast tunnels among several machine learning techniques. *Automation in Construction*, **127**, 103719. https://doi.org/10.1016/j.autcon.2021.103719.

Mahmoodzadeh, A., Mohammadi, M., Farid Hama Ali, H., Hashim Ibrahim, H., Nariman Abdulhamid, S., & Nejati, H. R. (2022). Prediction of safety factors for slope stability: Comparison of machine learning techniques. *Natural Hazards*, **111**, 1771–1799. https://doi.org/10.1007/s11069-021-05115-8.

Melekhov, I., Ylioinas, J., Kannala, J., & Rahtu, E. (2017a). Image-based localization using hourglass networks. In *Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (pp. 879–886). IEEE. https://doi.org/10.1109/iccvw.2017.107.

Melekhov, I., Ylioinas, J., Kannala, J., & Rahtu, E. (2017b). Relative camera pose estimation using convolutional neural networks. In *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems* (pp. 675–687). Springer. https://doi.org/10.1007/978-3-319-70353-4_57.

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, **31**, 1147–1163. https://doi.org/10.1109/TRO.2015.2463671.

Mur-Artal, R., & Tardós, J. D. (2017). ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, **33**, 1255–1262. https://doi.org/10.1109/TRO.2017.2705103.

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). IEEE. https://doi.org/10.1109/iros.2017.8205957.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, **32**, 8026–8037. https://doi.org/10.48550/arXiv.1912.01703.

Radwan, N., Valada, A., & Burgard, W. (2018). VLocNet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, **3**, 4407–4414. https://doi.org/10.1109/lra.2018.2869640.

Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision* (pp. 2564–2571). IEEE. https://doi.org/10.1109/ICCV.2011.6126544.

Shavit, Y., Ferens, R., & Keller, Y. (2021). Paying attention to activation maps in camera pose regression. *arXiv preprint* arXiv:2103.11477. https://doi.org/10.48550/arXiv.2103.11477.

Shavit, Y., & Ferens, R. (2021). Do we really need Scene-specific pose encoders? In *Proceedings of the 2020 25th International Conference on*

*Pattern Recognition (ICPR)* (pp. 3186–3192). https://doi.org/10.1109/icpr48806.2021.9412225.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint* arXiv:1409.1556. https://doi.org/10.48550/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 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–9). IEEE. https://doi.org/10.1109/cvpr.2015.7298594.

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). IEEE. https://doi.org/10.1109/icra.2018.8462979.

Varshni, D., Thakral, K., Agarwal, L., Nijhawan, R., & Mittal, A. (2019). Pneumonia detection using CNN based feature extraction. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)* (pp. 1–7). https://doi.org/10.1109/icecct.2019.8869364.

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 (ICCV)* (pp. 627–637). IEEE. https://doi.org/10.1109/iccv.2017.75.

Wu, C. (2011). VisualSFM: A visual structure from motion system. http://www.cs.washington.edu/homes/ccwu/vsfm.

Yu, H., Zhen, W., Yang, W., Zhang, J., & Scherer, S. (2020). Monocular camera localization in prior LiDAR maps with 2D-3D line correspondences. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4588–4594). IEEE. https://doi.org/10.1109/IROS45743.2020.9341690.

Yuan, Y., & Zhang, Y. (2021). OLCN: An optimized low coupling network for small objects detection. *IEEE Geoscience and Remote Sensing Letters*, **19**, 1–5. https://doi.org/10.1109/lgrs.2021.3122190.

Zhang, R., Luo, Z., Dhanjal, S., Schmotzer, C., & Hasija, S. (2018). Posenet++: A CNN Framework for Online Pose Regression and Robot Re-Localization.