Sensors and Materials, Vol. 32, No. 4 (2020) 1245–1259
MYU Tokyo

S & M 2175

Image-similarity-based Convolutional Neural Network for Robot Visual Relocalization

Li Wang,[1] Ruifeng Li,[1*] Jingwen Sun,[1] Hock Soon Seah,[2]
Chee Kwang Quah,[3] Lijun Zhao,[1**] and Budianto Tandianus[2]

[1]State Key Laboratory of Robotics and System, Harbin Institute of Technology,
92 Xidazhi Street, Harbin 150006, China
[2]School of Computer Science and Engineering, Nanyang Technological University,
Nanyang Avenue 639798, Singapore
[3]ST Electronics (Training & Simulation Systems) Pte Ltd.,
24 Ang Mo Kio Street, 560353 Singapore

(Received July 31, 2019; accepted December 2, 2019)

Keywords: visual relocalization, CNN, image similarity

Convolutional neural network (CNN)-based methods, which train an end-to-end model to regress a six degree of freedom (DoF) pose of a robot from a single red–green–blue (RGB) image, have been developed to overcome the poor robustness of robot visual relocalization recently. However, the pose precision becomes low when the test image is dissimilar to training images. In this paper, we propose a novel method, named image-similarity-based CNN, which considers the image similarity of an input image during the CNN training. The higher the similarity of the input image, the higher precision we can achieve. Therefore, we crop the input image into several small image blocks, and the similarity between each cropped image block and training dataset images is measured by employing a feature vector in a fully connected CNN layer. Finally, the most similar image is selected to regress the pose. A genetic algorithm is utilized to determine the cropped position. Experiments on both open-source dataset 7-Scenes and two actual indoor environments are conducted. The results show that the proposed algorithm leads to better results and reduces large regression errors effectively compared with existing solutions.

1. Introduction

Relocalization is a vital module for the long-term operations of a robot (such as planning and navigation) in an environment.[1–3] A service robot running in several indoor rooms or offices establishes an environment map as it moves around the environment. Usually, the robot needs to move in the environment many times in order to build a complete indoor map. When it restarts in a room where it has been before, it should obtain its 6D pose in the map global coordinate system by using its relocalization module. The core problem of visual relocalization is to estimate the robot's pose through the images from a camera. Recently, visual sensors, such

*Corresponding author: e-mail: lrf100@hit.edu.cn
**Corresponding author: e-mail: zhaolj@hit.edu.cn
https://doi.org/10.18494/SAM.2020.2549

as monocular and RGB-D cameras, are widely utilized in robots for environment mapping and perception because these cameras are very affordable compared with laser sensors.

Owing to strong interest in relocalization, many algorithms have been proposed. One main component of visual-based robot relocalization is visual pose estimation in a world coordinate system as the camera is often fixed on a robot. It can be divided into three main methods: keyframe-based, feature-based, and learning-based methods.

Keyframe-based methods select the most similar image in collected keyframes (with poses) and estimate a relative pose.[4,5] The global pose can be obtained by transferring the pose to the world coordinate system according to the pose of a selected keyframe. Some successful algorithms have been proposed for such methods.

Feature-based methods store feature points extracted in images rather than in a large number of keyframes.[6–9] Corresponding descriptors and positions (in the world coordinate system) of the detected feature points in images are stored in a database. Then, when conducting relocalization, the feature points are detected in the current image and matched with those in the database. The pose will be estimated after optimization.

Most relocalization algorithms adopt these methods because of the availability of robust feature detectors and descriptors to find matches. However, feature matching does not work accurately and robustly enough in all scenarios. The disadvantage of these approaches is the reliance on feature detection and matching. It leads to failure when fewer features are extracted in the presence of motion blur, textureless images, occlusions, dimly lit scenes or similar structures. Another problem is the deterioration of robustness and accuracy when the image similarity between the query image and the collected keyframes is too low owing to the sparsity of keyframes.

Learning-based methods have shown potentially efficient solutions to the pose estimation problem in recent years. Shotton *et al.* proposed a scene coordinate regression forest (SCoRF), which is successfully applied to camera pose estimation.[10,11] However, a depth map associated with an input image is required during training. Therefore, the applicability of the approach is restricted.

With their rapid development in recent years, neural networks have achieved great success in image classification,[12,13] image retrieval,[14–16] semantic segmentation,[17–19] and various applications.[20–23] Nowadays, convolutional neural networks (CNNs) have also been applied to estimate camera pose from images. The pose relocalization is considered as a regression problem as it is directly estimated by a CNN. Initially, Kendall and coworkers proposed an algorithm named PoseNet to directly regress the camera pose using the CNN[24,25] method (adopting the GoogLeNet[26] architecture). Another framework named Bayesian PoseNet considers uncertainty in pose estimation by averaging Monte Carlo dropout samples from the posterior distribution of the Bayesian CNN's weights.[27] These two models have achieved good performance in both indoor and outdoor datasets. Melekhov *et al.* utilized an hourglass network with a symmetric encoder-decoder network structure, which had improved accuracy compared with PoseNet.[28] Motivated by recurrent neural networks in text classification,[29,30] some approaches are proposed. Clark *et al.* used a recurrent model for the 6-DoF pose estimation of video clips, which exploited the temporal smoothness of a video stream in order to improve

global accuracy.[31] The major drawback of this method is that it requires a sequence of adjacent images as inputs.

Although learning-based algorithms can solve many disadvantages of feature-based methods, some issues remain unsolved. For instance, pose error is large when there is a large dissimilarity between a test image and a training dataset. The accuracy of these methods needs to be improved before they can be used in practical applications. In this work, we explore the impact of an input image with a different image similarity on pose regression accuracy for robot relocalization, and we propose an image-similarity-based CNN. The input image is cropped to several small image blocks, and then the similarity between each cropped image block and training dataset images is measured by using the feature vector in a fully connected CNN layer. Finally, the image with the highest similarity is selected for pose regression.

In summary, we make the following contributions:

(1) We contribute a novel idea: a higher similarity of an input image can achieve a higher precision when utilizing CNNs for visual relocalization.
(2) We propose a whole pipeline to select the most similar image as an input to a CNN using only an RGB image.
(3) An algorithm that clusters feature vectors of a training dataset can effectively reduce the computational complexity in the image similarity computation.

The rest of the paper is organized as follows. In Sect. 2, we describe the proposed visual pose regression algorithm in detail. Experiments on open-source datasets and two real environments are explained in Sect. 3. Conclusions and some suggestions for future work are given in Sect. 4.

2. Design for Robot Visual Relocalization Algorithm

In this section, we introduce the image-similarity-based CNN for robot relocalization using a single RGB image. The relocalization problem is considered as pose regression as it utilizes an end-to-end CNN method.

2.1 System structure

Usually, visual pose regression algorithms based on deep learning require images and the corresponding poses to train network parameters. Then, the pose regression is performed on the test dataset. It can be observed that the accuracy of the pose regression is higher when the trajectory of the test dataset is closer to the training dataset. Higher similarities between images on the test and training datasets can result in a higher accuracy for the estimated image pose. Therefore, it is possible to improve the accuracy by obtaining an input image with high similarity to the training dataset.

The visual pose regression system structure is shown in Fig. 1. In general, the main idea is to crop an input RGB image into several small image blocks and find some images with high similarity to the images in a training dataset, and the pose regression is carried out through the CNN. Firstly, transfer learning is utilized to train a pose regression network based on the

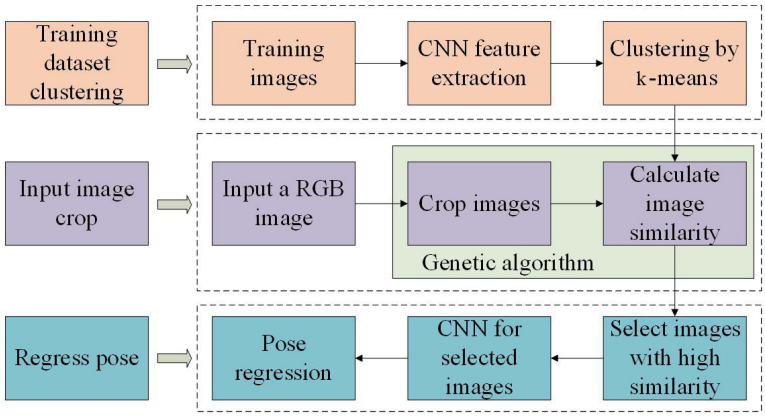


Fig. 1. (Color online) Framework of visual pose relocalization system.

PoseNet network structure that uses the GoogleNet Inception V1 network as the backbone. Secondly, to reduce the computational complexity of image similarity, images of the training dataset are clustered. The trained regression model is used to extract a feature of the image, which will form a feature vector. Then, the *k*-means clustering algorithm is adopted to cluster feature vectors. Thirdly, the cropped position of the image, which is regarded as the optimal variable, is optimized by leveraging a genetic algorithm. The similarity between the cropped image and training dataset images is treated as the fitness function. Finally, the pose can be obtained by utilizing the trained model to calculate the selected cropped image with the highest similarity.

2.2 Pose regression network based on transfer learning

The GoogLeNet Inception V1 network is adopted for transfer learning in the pose regression algorithm, which is trained on the ImageNet dataset. Its output structure is modified as follows. Three fully connected layers named SoftMax are removed and each removed layer is replaced with a 7-dimensional vector that consists of a 3D position vector and a quaternion. The network structure is shown in Fig. 2.

During the training process, a Euclidean loss function is utilized and Stochastic Gradient Descent (SGD) is applied to train the network model. The loss function $L(I)$ is defined as

$$L(I) = \sum_{i=1}^{3} \alpha_i L_i(I), \tag{1}$$

$$L_i(I) = \left\| \hat{t} - t \right\|_2 + \beta_i \left\| \hat{q} - \frac{q}{\|q\|} \right\|_2 \left(i = 1, 2, 3 \right), \tag{2}$$

where I is the input RGB image, $L_i(I)$ is a loss function of the i-th fully connected layer's output, α_i is the weight of the i-th loss function, and t and q are a 3D position vector and a quaternion

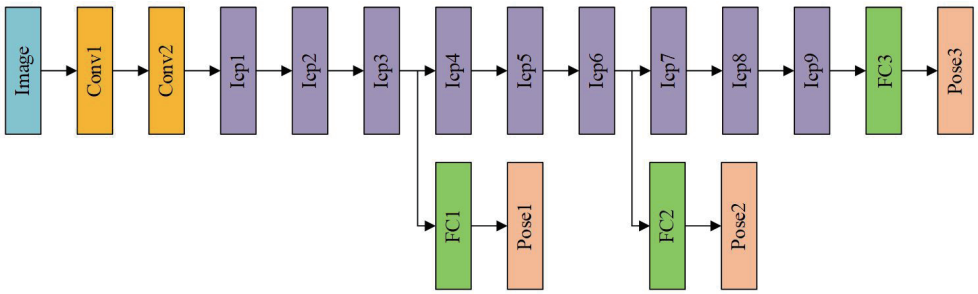


Fig. 2. (Color online) Framework of pose regression CNNs.

vector representing the orientation, $\hat{t}$ and $\hat{q}$ are the position and orientation obtained by pose regression, respectively, and β_i is a scale factor in the i-th loss function for balancing position and orientation errors.

2.3 Feature vector clustering based on *k*-means

To select an image from cropped input images with the highest similarity to a training dataset, a similarity measurement between images needs to be defined. In the field of image retrieval, a CNN has been widely applied and it achieves high accuracy. The neural network can extract a feature vector from an image automatically and perform feature matching. In the above network structure, the previous layer of the output pose is a fully connected layer with a 2048-dimensional vector. Therefore, this fully connected layer can be utilized as a feature vector of the image to measure the image similarity.

The feature vector extraction diagram of training dataset images is shown in Fig. 3. Let the number of images in a training dataset be m, and the set of feature vectors is defined as $\boldsymbol{U}_{train} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_m\}$. The feature vector $\boldsymbol{u}_i$ of the i-th image I_i extracted by a pose regression neural network is defined as

$$\boldsymbol{u}_i = f_{CNN}(I_i) \ (i \in [1,m], i \in N^+), \tag{3}$$

where f_{CNN} is the CNN for extracting feature vectors.

We use the same extraction method to extract feature vectors in the test dataset images. Then, vector distances are calculated for each feature vector of the test dataset image with all feature vectors in the training dataset, and the minimum distance gives the best image similarity. Usually, there are too many images in the training dataset. For example, each scene of Microsoft 7-Scenes datasets has thousands of images, and each feature vector is a 2048-dimensional vector. Therefore, vector distance calculation will suffer from high computational cost.

To reduce the computational cost, the training dataset feature vectors are clustered to obtain several clustering centers. Then, the distances between feature vectors of test images and clustering centers are compared as the measure of image similarity. The *k*-means clustering

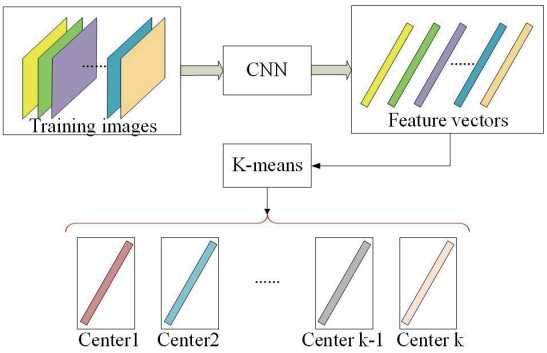


Fig. 3. (Color online) Feature vector extraction of training dataset images.

method is leveraged to cluster feature vectors of training dataset images. Let the number of clustering centers be k and the set of vectors be $\boldsymbol{C}_{train} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_k\}$, and the vector's dimension of clustering centers will remain unchanged. We need to perform data standardization before the clustering. Let the mean value of training vectors be μ, standard deviation be σ, and vector's dimension be s, and the j-th dimension of the i-th feature vector $\boldsymbol{u}'_i(j)$ is

$$\boldsymbol{u}'_i(j) = \begin{cases} \dfrac{\boldsymbol{u}_i(j) - \mu(j)}{\sigma(j)}, & \sigma(j) \neq 0 \\ 0, & \sigma(j) = 0, \end{cases} \quad (i \in [1, m],\ j \in [1, s],\ i, j \in N^+) \tag{4}$$

where $\mu(j)$ and $\sigma(j)$ are the mean and standard deviation of the j-th dimension in the feature vector, respectively.

2.4 Input image crop based on genetic algorithm

The input image size of the pose regression network is generally smaller than those of the test images (for example, the image resolution of Microsoft's 7-Scenes datasets is 640 × 480, whereas the network needs an input image with a resolution of 224 × 224). Thus, we need to preprocess the input image. Normally, the center cropping method is adopted. However, the closer the cropped image to those in the training dataset, the higher the pose accuracy that can be regressed in the experiments. Therefore, to improve accuracy, an appropriate cropped image should be selected. Let the input image resolution of the test dataset be $w \times h$; we crop the image to the resolution of $h \times h$ first, then compress it to $h' \times h'$. The obtained upper left corner of the cropped image is in the range of $[0, w - h]$. Therefore, it needs to obtain a suitable cropped position so that the similarity between the image and the training dataset is the highest.

To determine the cropped position of the input image, the genetic algorithm is utilized for optimization. The genetic algorithm is a type of meta-heuristic search algorithm based on biological evolution, which is to solve the optimization problem. It can directly manipulate objects, and there is no limit to the continuity of the function. The meta-heuristic optimization

of the genetic algorithm fits well to our problem as it can automatically guide the optimization of search space and adaptively adjust the search direction. For these reasons, the genetic algorithm is implemented to search for the optimal cropped position of the input image.

The framework of the input image crop based on the genetic algorithm is shown in Fig. 4. The optimization variable represents the cropped position in the image, and the fitness function is the similarity between a cropped image and a training dataset image. Usually, image similarity adopts a feature-based method such as the bag-of-words model[32,33] for extracting and searching feature points in an image. However, an effective image similarity measure cannot be performed when the feature points are difficult to match if the images suffer from motion blur and are taken at different viewpoints.

As the trained pose regression neural network contains nine perception modules (the same ones as in the GoogleNet Inception V1 network) and convolutional, pooling, and other layers, image information can be abstracted and extracted effectively. Therefore, the image similarity is measured by using the feature vector of the fully connected CNN layer. We need to determine the following criteria: (1) encoding and decoding methods of a feasible solution, (2) fitness function design, and (3) genetic operations of the genetic algorithm for optimizing the input image cropped location. These criteria are defined as follows:

(1) Encoding and decoding methods of feasible solution

The encoding of a feasible solution is for expressing solution space in a chromosome representation in order to facilitate genetic operations, such as crossover and mutation. Owing to the high stability of the binary code and the difficulty in falling into local optimum, it is used to encode the cropped position of the image. The range of the upper left corner in the cropped image is $x \in [0, w-h]$. Because the cropped positions are all integers and the solution accuracy can be set to one pixel, the solution space can be divided into $w - h$ equal parts. The number of bits, n, of the binary string can be determined as

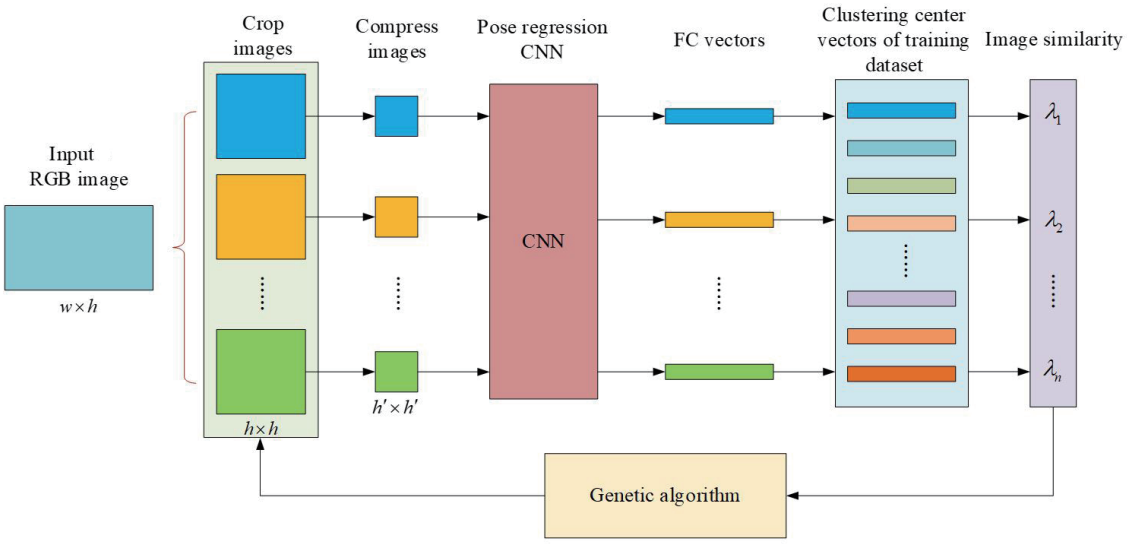


Fig. 4. (Color online) Input image crop based on genetic algorithm.

$$2^{n-1} < w - h < 2^n. \tag{5}$$

Each binary string is represented as a chromosome, which is decoded after processing by genetic operators. Let the chromosome string be X, then the decoded position is

$$x = Round\left(\frac{Decimal(X)}{2^n - 1} \cdot (w - h) \right), \tag{6}$$

where $Round(\cdot)$ is a rounding function and $Decimal(\cdot)$ is a function to transfer the binary representation to the integer representation.

(2) Fitness function design

Fitness function is an evaluation function that judges the fitness of each individual and serves as a basis for future genetic operations. In general, the greater the function value, the higher the quality of the feasible solution. The designed fitness function evaluates multiple cropped images in order to select the optimal cropped position. By utilizing a trained pose regression network to extract feature vectors, the image similarity is measured by calculating feature vector distances between the image and the clustering centers.

The input image is cropped in a random manner to obtain w sheets of cropped images $I_s (s \in [1, w],\ s \in N^+)$ with the size of $h \times h$. The feature vector $\boldsymbol{\mu}_s$ is extracted by leveraging the trained pose regression network according to Eq. (3).

$$\boldsymbol{\mu}_s = f_{CNN}(I_s)\ (s \in [1, w],\ s \in N^+), \tag{7}$$

where f_{CNN} is the pose regression network.

The image similarity is measured by calculating distances between feature vectors. The distance between the cropped image feature vector $\boldsymbol{\mu}_s$ and the clustering center of each feature vector $\boldsymbol{c}_k$ of the training dataset is calculated and the minimum distance λ_s is taken as image similarity:

$$\lambda_s = \min\left(\left\| \boldsymbol{\mu}_s - \boldsymbol{c}_1 \right\|_2, \left\| \boldsymbol{\mu}_s - \boldsymbol{c}_2 \right\|_2, \cdots, \left\| \boldsymbol{\mu}_s - \boldsymbol{c}_k \right\|_2 \right). \quad (s \in [1, w],\ s \in N^+) \tag{8}$$

In a genetic algorithm, the fitness function is utilized to calculate selection probabilities and is generally designed to be in the form of maximum and non-negative function values. Therefore, an exponential function is adopted to define the following fitness function y_s:

$$y_s = f(I_s) = e^{-\lambda_s}. \tag{9}$$

(3) Genetic operation

Genetic operations such as selection, crossover, and mutation are adopted to carry out population evolution. The selection operation is to select the best individuals from the feasible solution of the previous generation to generate the next-generation populations.

A fitness function is applied to evaluate the fitness of each individual and rank it. We adopt the Roulette selection method, which selects two individuals from the population as parents according to the probability (which is derived from their fitness) of each individual. The probability of the cropped image I_s to be selected is

$$p(I_s) = \frac{y_s}{\sum_{r=1}^{w} y_r}. \tag{10}$$

The selected parent chromosomes undergo crossing and mutation with a certain probability in order to generate offsprings. This process is repeated until the number of offsprings reaches the predetermined population size. Finally, the progress stops when the iteration is completed.

2.5 Pose regression

According to the method, the cropped image with the highest similarity to the training dataset is obtained. Then, this image is regarded as the input in the network to regress the pose as described in Sect. 2.2. Finally, we can obtain the pose for each test image.

3. Experimental Evaluation

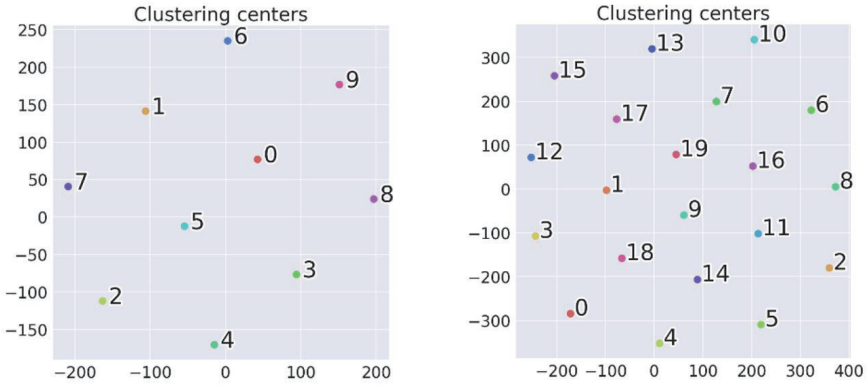
We conduct our experiments on both open-source datasets and actual indoor environments to validate the proposed approach. The dataset is the 7-Scenes captured by Microsoft Research Institute with an RGB-D camera. It has been widely used for visual tracking and relocalization validation. The images are captured at a resolution of 640 × 480. The dataset contains seven indoor scenes including images, 3D densely reconstructed maps, and camera trajectories. Each scene is divided into a training dataset and a test dataset with the ground truth generated from a KinectFusion system. In the experiments, our method is implemented in all seven scenes in order to evaluate the proposed method, and the same sequences are adopted in the original paper.[25] This dataset consists of both RGB and depth images, but we mainly focus on only RGB pose regression in this paper. Furthermore, we carry out two indoor experiments for robot relocalization to verify the adaptability of the algorithm in the real world.

3.1 Training dataset feature vector clustering

Experiments are conducted on the 7-Scenes dataset, and feature vectors of the training dataset are extracted by utilizing the pose regression network. The input image resolution is 640 × 480. By using the central crop method, the cropped image resolution is 480 × 480, and an image with a resolution of 224 × 224 can be obtained after compressing it into the required input size of the network. After extracting feature vectors of the training dataset, *k*-means algorithm is used to perform clustering in order to reduce the complexity of vector calculation. We set the number of clustering centers according to the number of training images, as shown in Table 1.

Table 1
Experiments on all Microsoft's 7-Scenes datasets.

No.	Scene	Train frames	Test frames	Clustering centers k	Position error (m)				Orientation error (°)			
					PoseNet	Bayesian PoseNet	Our method	Error decrease (%)	PoseNet	Bayesian PoseNet	Our method	Error decrease (%)
1	Chess	4000	2000	40	0.32	0.37	**0.21**	−34.4	8.12	7.24	**5.73**	−29.4
2	Fire	2000	2000	20	0.47	0.43	**0.40**	−14.9	14.4	13.7	**12.11**	−15.9
3	Heads	1000	1000	10	0.29	0.31	**0.25**	−13.8	**12.0**	**12.0**	14.38	+19.8
4	Office	6000	4000	50	0.48	0.48	**0.30**	−37.5	7.68	8.04	**7.58**	−1.3
5	Pumpkin	4000	2000	40	0.47	0.61	**0.37**	−21.3	8.42	**7.08**	7.46	−11.4
6	Red kitchen	7000	5000	50	0.59	0.58	**0.42**	−28.8	8.64	7.54	**7.11**	−17.7
7	Stairs	2000	1000	20	0.47	0.48	**0.36**	−23.4	13.8	13.1	**11.82**	−14.3
Average					0.44	0.47	**0.33**	**−25.2**	10.44	9.81	**9.46**	**−9.4**



(a) (b)

Fig. 5. (Color online) Visualization of cluster centers (after dimensional reduction). (a) Heads (k = 10) and (b) stairs (k = 20) (k is the number of clustering centers).

To analyze the clustering performance, the *t*-distributed stochastic neighbor embedding (*t*-SNE) algorithm is employed to reduce the dimension of the clustering centers from 2048 to 2 so that they can be displayed in a 2D diagram. Figure 5 shows the 2D distributions of the clustering center vectors of "Heads" and "Stairs" datasets, which are evenly distributed, indicating that the clustering performs well.

3.2 Training for pose regression network based on transfer learning

The transfer learning algorithm is carried out to train the pose regression network, and the output layer of the network is modified. TensorFlow library of Google is utilized in the training process. During training, the SGD is adopted with a base learning rate of 10^{-5}. Using an NVIDIA Tesla P100 GPU, it takes around 4 h for the batch size of 75 during the training process with the number of iterations being 30000.

3.3 Image crop of test dataset utilizing the genetic algorithm

Each image in the test dataset is cropped and compressed to a suitable size of the network. Each image in the 7-Scenes dataset has a resolution of 640 × 480. After cropping, the image resolution is 480 × 480. It is resized to 224 × 224 for the network. Therefore, the range of the upper left corner in the cropped image is [0, 160]. As the cropped positions are all integers and the solution precision is set to one pixel, the solution space can be divided into 160 equal parts. The chromosome coding mode is adopted and requires 8-bit binary according to Eq. (5).

The operating parameters of the genetic algorithm to select the optimal cropped image are shown in Table 2. According to the algorithm described above, the fitness function and genetic operators are used to solve the problem. After the iteration, the image with the highest similarity is selected to calculate the corresponding pose through the trained network model.

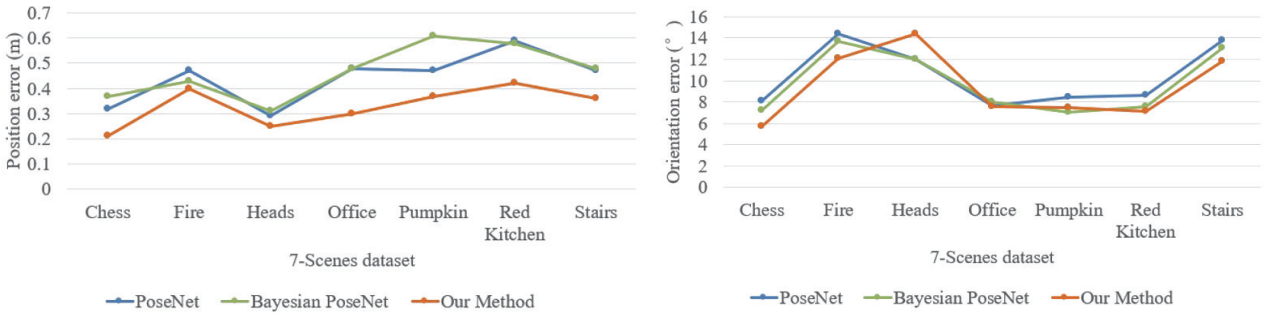
3.4 Experimental comparison and analysis

Experiments on all scenes of the 7-Scenes dataset are performed first to verify the effectiveness of the proposed algorithm. Then, we compare it with the previous PoseNet and Bayesian PoseNet algorithms on the dataset. The experimental results are shown in Table 1 (the percentage of error reduction in Table 1 is compared with the result of the PoseNet algorithm). Figure 6 shows the comparison of position and orientation errors of the three algorithms.

The experimental results show that the proposed algorithm can reduce position error in all the seven scenes and also orientation error for most scenes except for the increase on the "Heads" dataset. Compared with PoseNet, the average position error is reduced by 25.2% and the average orientation error is reduced by 9.4%. In summary, the effectiveness of the proposed algorithm is verified.

Table 2
Operating parameters of the genetic algorithm.

Parameters	Values
Number of population M	10
Crossover probability P_c	0.8
Mutation probability P_m	0.1
Number of iterations N	3



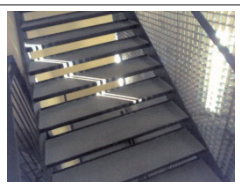
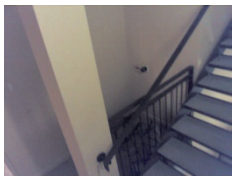
(a) (b)

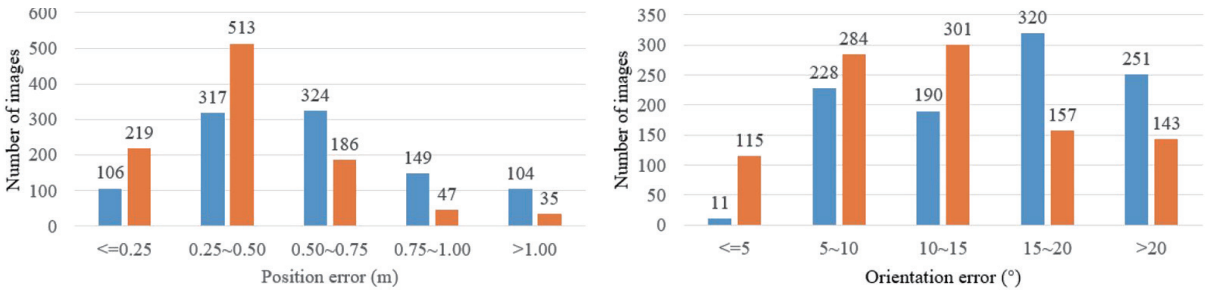
Fig. 6. (Color online) Comparisons on the 7-Scenes dataset. (a) Position and (b) orientation errors.

To prove the effectiveness of the proposed algorithm in detail, we use the "Stairs" dataset as an example. We present a comparative analysis of the proposed method with the PoseNet algorithm. We conduct comparative experiments to verify the effectiveness of image cropping in the proposed method. Two test images from the "Stairs" dataset are randomly selected (the 200th and 700th images) to present details, as shown in Table 3. The resolution of the test image is 640 × 480, while the resolution of the CNN is 224 × 224. In Table 3, we show the input images for PoseNet and our method. For PoseNet, the center cropping method is used; however, the image similarity is utilized in our method to crop the input image. Therefore, the input images of the CNN are slightly different. The regression poses are also given in the table. It can be observed that our method improves accuracy significantly.

We carry out the comparative experiments in all the test images of "Stairs", which has 1000 images. Pose regression is performed for each image by utilizing the above algorithms, and the position and orientation errors are calculated. Figure 7 shows two histograms of the error distribution on the test "Stairs" dataset. In the pose regression of 1000 images, for the proposed algorithm, the ratio of position error less than 0.5 m is 73.2%, and the ratio of orientation error less than 15° is 70.0%, while they are 42.3 and 42.9% for PoseNet, respectively. In terms of

Table 3
(Color online) Comparative experiments of the input image for CNN.

No.	Test image (640 × 480)	Input image of CNN (224 × 224)		Position error (m)		Orientation error (°)	
		PoseNet	Our method	PoseNet	Our method	PoseNet	Our method
1				0.42	0.22	23.2	17.4
2				0.64	0.40	22.9	16.3



(a) (b)

Fig. 7. (Color online) Error histograms of the proposed and PoseNet algorithms on the "Stairs" dataset. (a) Position and (b) orientation errors.

larger error magnitude, the proposed algorithm has 3.5% position error greater than 1.0 m and 14.3% orientation error greater than 20°, while they are 10.4 and 25.1% for PoseNet, respectively. Thus, the position and orientation errors of the proposed algorithm are concentrated in the range of smaller errors, and the number of large errors is significantly reduced. To observe the details clearly, all the results are shown in Figs. 8(a) and 8(b), which present the contrast in position and orientation errors, respectively. It can be observed that the proposed algorithm achieves lower errors in the pose estimation for most images and its error fluctuation is gentle. Moreover, the maximum error is significantly reduced.

In addition, we carry out two actual experiments using an indoor mobile robot. Two indoor environments, a floor and a room are selected. We control the robot attached with a Kinect sensor to move around a floor and a room separately. RGB images are recorded and ORB-SLAM2[34] is implemented to obtain the pose of each image. The 3D reconstruction and trajectories are shown in Fig. 9. We consider two laps as training data and one lap as test data.

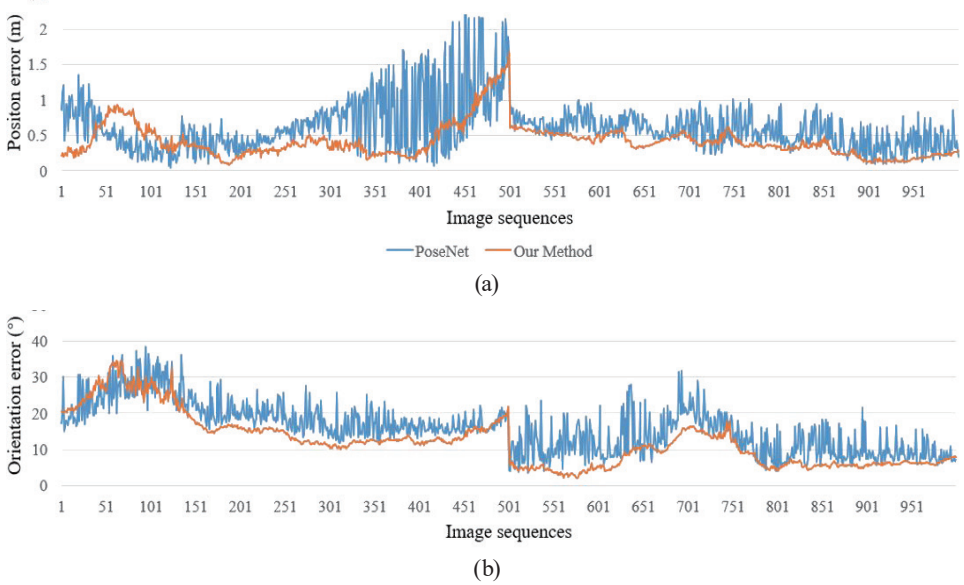


Fig. 8. (Color online) Comparison of the proposed and PoseNet algorithms on the "Stairs" dataset. (a) Position and (b) orientation errors.

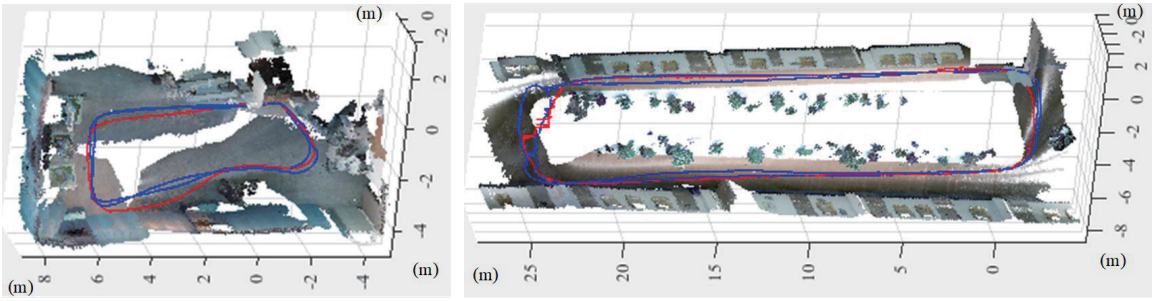


Fig. 9. (Color online) Two actual relocalization experiments in a floor and a room, respectively. The training trajectory is drawn in blue and the test trajectory is drawn in red. Experiments in (a) a floor and (b) a room.

Table 4
Comparative experiments with PoseNet in actual environments.

No.	Scenes	Train frames	Test frames	Position error (m)			Orientation error (°)		
				PoseNet	Our method	Error decrease(%)	PoseNet	Our method	Error decrease(%)
1	Room	3395	1548	0.42	**0.21**	-50.0	8.26	**2.34**	−71.7
2	Floor	5450	2655	1.65	**0.57**	-65.5	7.87	**1.20**	−84.8
Average				1.04	**0.39**	-57.7	8.07	**1.77**	−78.2

It can be clearly seen that the trajectories are not coincident. The same parameters are utilized to train the model with the dataset. The PoseNet method is also trained for comparison. The results are shown in Table 4. The average position error is reduced by 57.7% and the average orientation error is reduced by 78.2%, which verifies the significant advantage of our proposed algorithm.

4. Conclusions

In this study, we investigated the visual relocalization problem for a robot based on a CNN. We proposed a novel image-similarity-based CNN algorithm in this paper. In addition, a pipeline to select the most similar image for pose regression was presented. The effectiveness of the algorithm was verified by experiments on both datasets and real environments. Compared with PoseNet, the average position error was reduced by 25.2% and the average orientation error was reduced by 9.4% on the datasets. The average errors were reduced by 57.7 and 78.2% in real environments, respectively. As the pose regression does not consider the temporal change of the robot pose, it suffers from temporal incoherence. In future work, continuous pose taking into account temporal element will be regressed to improve accuracy.

Acknowledgments

This work was supported by the National Key Research and Development Program “Intelligent Robot” Key Special Project (2018YFB1308900), the National Natural Science Foundation of China (61673136), the Self-Planned Task of State Key Laboratory of Robotics and System (HIT) (Nos. SKLRS201906B and SKLRS201715A), the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (No. 51521003), ST Engineering-NTU Corporate Lab through the NRF corporate lab@university scheme, and the China Scholarship Council (No. 201706120137).

References

1 B. Glocker, J. Shotton, A. Criminisi, and S. Izadi: IEEE Trans. Visual Comput. Graphics **21** (2015) 571. https://doi.org/10.1109/TVCG.2014.2360403
2 R. Mur-Artal and J. D. Tardós: Int. Conf. Robotics and Automation (IEEE, 2014) 846. https://doi.org/10.1109/ICRA.2014.6906953
3 C. Yang, Y. Jiang, J. Na, Z. Li, L. Cheng, and C.-Y. Su: IEEE Trans. Fuzzy Syst. **27** (2019) 574. https://doi.org/10.1109/TFUZZ.2018.2864940
4 A. P. Gee and W. Mayol-Cuevas: Cuevas, 2012, pp. 1–11. https://doi.org/10.5244/C.26.113

5 B. Glocker, S. Izadi, J. Shotton, and A. Criminisi: IEEE Int. Symp. Mixed and Augmented Reality (IEEE, 2013) 173. https://doi.org/10.1109/ISMAR.2013.6671777
6 B. Williams, G. Klein, and I. Reid: IEEE Trans. Pattern Anal. Mach. Intell. **33** (2011) 1699. https://doi.org/10.1109/TPAMI.2011.41
7 R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos: IEEE Trans. Robot. **31** (2015) 1147. https://doi.org/10.1109/TRO.2015.2463671
8 C. Evers and P. A. Naylor: IEEE Trans. Signal Process. **66** (2018) 863. https://doi.org/10.1109/TSP.2017.2775590
9 F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard: IEEE Trans. Robot. **30** (2014) 177. https://doi.org/10.1109/TRO.2013.2279412
10 J. Shotton, B. Glocker, C. Zach, S. Izadi, and A. Fitzgibbon: IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2013) 2930. https://doi.org/10.1109/CVPR.2013.377
11 J. Valentin, M. Niebner, J. Shotton, A. W. Fitzgibbon, S. Izadi, and P. H. S. Torr: IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2015) 4400. https://doi.org/10.1109/CVPR.2015.7299069
12 K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun: IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2016) 770. https://doi.org/10.1109/CVPR.2016.90
13 C. H. Lin, C. W. Liu, and H. Y. Chen: IET Image Process. **6** (2012) 822. https://doi.org/10.1049/iet-ipr.2011.0445
14 A. Amanatiadis, V. G. Kaburlasos, A. Gasteratos, and S. E. Papadakis: IET Image Process. **5** (2011) 493. https://doi.org/10.1049/iet-ipr.2009.0246
15 J. Yang, J. Liang, H. Shen, K. Wang, P. L. Rosin, and M. H. Yang: IEEE Trans. Image Process. **27** (2018) 5288. https://doi.org/10.1109/TIP. 2018.2845136
16 P. Liu, J. Guo, C. Wu, and D. Cai: IEEE Trans. Image Process. **26** (2017) 5706. https://doi.org/10.1109/TIP.2017.2736343
17 Z. Liu, X. Li, P. Luo, C. Change Loy, and X. Tang: IEEE Trans. Pattern Anal. Mach. Intell. **40** (2018) 1814. https://doi.org/10.1109/TPAMI. 2017.2737535
18 G. Pagnutti, L. Minto, and P. Zanuttigh: IET Comput. Vis. **11** (2017) 633. https://doi.org/10.1049/iet-cvi.2016.0502
19 W. Su and Z. Wang: IET Image Process. **11** (2017) 880. https://doi.org/10.1049/iet-ipr.2017.0070
20 C. Yang, Z. Wang, W. He, and Z. Li: Multimed Tools Appl. **77** (2018) 25369. https://doi.org/10.1007/s11042-018-5789-8
21 C. Yang, C. Chen, N. Wang, Z. Ju, J. Fu, and M. Wang: IEEE T. Cogn. Dev. Syst. **11** (2019) 281. https://doi.org/10.1109/TCDS.2018.2866477
22 C. Yang, G. Peng, Y. Li, R. Cui, L. Cheng, and Z. Li: IEEE Trans. Cybern. **49** (2018) 2568. https://doi.org/10.1109/TCYB.2018.2828654
23 A. H. Abdulnabi, G. Wang, J. Lu, and K. Jia: IEEE Trans. Multimedia **17** (2015) 1949. https://doi.org/10.1109/TMM.2015.2477680
24 A. Kendall and R. Cipolla: IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2017) 6555. https://doi.org/10.1109/CVPR.2017.694
25 A. Kendall, M. Grimes, and R. Cipolla: IEEE Int. Conf. Computer Vision (IEEE, 2015) 2938. https://doi.org/10.1109/ICCV.2015.336
26 C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich: IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2015) 1. https://doi.org/10.1109/CVPR.2015.7298594
27 A. Kendall and R. Cipolla: Int. Conf. Robotics and Automation (IEEE, 2016) 4762. https://doi.org/10.1109/ICRA.2016.7487679
28 I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu: IEEE Int. Conf. Computer Vision Workshops (IEEE, 2017) 870. https://doi.org/10.1109/ICCVW.2017.107
29 T. V. Phan and M. Nakagawa: Int. Conf. Frontiers in Handwriting Recognition (IEEE, 2014) 23. https://doi.org/10.1109/ICFHR.2014.12
30 P. Xu and R. Sarikaya: IEEE Int. Conf. Acoustics, Speech and Signal Processing (IEEE, 2014) 136. https://doi.org/10.1109/ICASSP.2014.6853573
31 R. Clark, S. Wang, A. Markham, N. Trigoni and H. Wen: IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2017) 2652. https://doi.org/10.1109/CVPR.2017.284
32 D. Galvez-López and J. D. Tardos: IEEE Trans. Robot. **28** (2012) 1188. https://doi.org/10.1109/tro.2012.2197158
33 M. M. Farhangi, M. Soryani, and M. Fathy: IET Image Process. **8** (2014) 310. https://doi.org/10.1049/iet-ipr.2013.0449
34 R. Mur-Artal and J. D. Tardos: IEEE Trans. Robot. **33** (2017) 1255. https://doi.org/10.1109/TRO.2017.2705103