Journal of Intelligent & Robotic Systems (2021) 102: 79
https://doi.org/10.1007/s10846-021-01439-6

REGULAR PAPER

Relative Camera Pose Estimation using Synthetic Data with Domain Adaptation via Cycle-Consistent Adversarial Networks

Chenhao Yang[1] · **Yuyi Liu**[2] · **Andreas Zell**[1]

Received: 4 October 2020 / Accepted: 14 June 2021 / Published online: 8 July 2021

Abstract

Learning-based visual localization has become prospective over the past decades. Since ground truth pose labels are difficult to obtain, recent methods try to learn pose estimation networks using pixel-perfect synthetic data. However, this also introduces the problem of domain bias. In this paper, we first build a *Tuebingen Buildings* dataset of RGB images collected by a drone in urban scenes and create a 3D model for each scene. A large number of synthetic images are generated based on these 3D models. We take advantage of image style transfer and cycle-consistent adversarial training to predict the relative camera poses of image pairs based on training over synthetic environment data. We propose a relative camera pose estimation approach to solve the continuous localization problem for autonomous navigation of unmanned systems. Unlike those existing learning-based camera pose estimation methods that train and test in a single scene, our approach successfully estimates the relative camera poses of multiple city locations with a single trained model. We use the *Tuebingen Buildings* and the *Cambridge Landmarks* datasets to evaluate the performance of our approach in a single scene and across-scenes. For each dataset, we compare the performance between real images and synthetic images trained models. We also test our model in the indoor dataset *7Scenes* to demonstrate its generalization ability.

Keywords Relative camera pose estimation · Domain adaptation · Image style transfer

1 Introduction

Simultaneous localization and mapping (SLAM) has prosperously evolved in recent years and is largely used in augmented reality and robot navigation. Visual SLAM infers camera movement from pixels as in dense SLAM system [1, 2] or by extracting sparse keypoints (e.g., SIFT [3], ORB [4]). In many cases, 3D geometry has been used to solve the localization problem.

Visual SLAM has been maturely applied to ground mobile robots and self-driving vehicles. However, for the localization of unmanned aerial or ground systems, traditional visual SLAM meets some challenges. First, the high-speed movement of Unmanned Aerial Vehicles (UAVs) causes massive changes in the viewpoints, which leads to a large appearance difference among keyframes [5, 6]. Second, the keypoints correspondences will decrease with untextured objects such as ceramic tile. Also, sunlight and shadows variations and glass / water reflections may also hamper feature matching. Third, the feature matching with repeated textures such as similar windows becomes noisy. Last, some approaches can only estimate the translation value proportionally, so good initialization is required [7].

Convolutional Neural Network (CNN) is largely applied in object recognition, image classification [8] and place recognition [9]. Meanwhile, structure from motion (SfM) [10] has shown great progress in 3D reconstruction, and obtained centimeter-level accuracy in localizing cameras and 3D points. SfM method generates a 3D model by images around the object, and the 6DoF poses of images can be used as training labels for camera pose estimation. CNN combined with SfM reduces the workload of constructing a database, makes it possible for deep-learning-based camera relocalization, and proposes new solutions to the problems faced by traditional visual SLAM.

Absolute camera pose regression based on deep learning usually trains a model to predict the pose of a input image to a certain scene. The model network implicitly remembers

✉ Chenhao Yang
chenhao.yang@uni-tuebingen.de

[1] Department of Computer Science, Chair of Cognitive Systems, University of Tübingen, Tübingen, Germany

[2] Department of Social Informatics, Graduate School of Informatics, HRI Laboratory, Kyoto University, Kyoto, Japan

the spatial information of the scene by the weights, so the system is not very versatile [11–13]. In contrast, estimating the image pair's relative pose is a more common problem. The ideal relative camera pose estimation model can not only be trained and tested in a fixed scene, but also can be tested in multiple seen locations or even new places.

Generally, a CNN-based camera relocalization system can be employed in 3 ways of increasing complexity and usefulness: the first way is to train and test the model on a fixed scene. The second way is to use multiple scenes to train a model and test it on each scene, which is the work of across-scenes training in this work. The third way is to train a model on multiple locations and test it in new environments. The last two ways can be developed more easily by relative camera pose estimation than by absolute camera pose regression. Across-scenes absolute camera pose regression is more difficult since different locations have skewed camera poses distribution due to the scale-inconsistency. In the meantime, the relative camera pose estimation has versatility in many robot applications. It could be used as a neighbor frame pose estimator in visual odometry systems. It can also predict robots' relative pose in a multi-robot cooperation system. The relative camera pose estimation could even combine with a global place recognition method such as NetVLAD [9] to get a more precise absolute pose prediction in large environments with a two-step localization pipeline.

Although SfM methods can generate pose labels for co-visualized images, the collection of images and the SfM process are extremely time-consuming. Compared to real-world data, synthetic images and related poses are much easier to obtain. As a result, synthetic data is used for many visual tasks [14–16]. However, the domain shift from synthetic to real has always been the most significant challenge in this area. The system trained on synthetic data is often not directly applicable to real data. In order reduce the discrepancy between domains, style transfer [17, 18] and domain adaptation methods [19, 20] are utilized.

Motivated by the above analysis, we present an across-scene relative camera pose estimation network (RCPNet) for urban outdoor camera relocalization. We collect over 10,000 images with drones in eight city locations to construct a dataset (hereafter called *Tuebingen Buildings* dataset) and obtain each image's absolute pose by the SfM method. We generate over 300,000 image pairs for relative camera pose estimation. The previous work of RCPNet and original *Tuebingen Buildings* dataset has been published in [65]. To further expand the training dataset, twice the amount of synthetic images are rendered from the 3D models of *Tuebingen Buildings* and *Cambridge Landmarks* [11] datasets. Inspired by CycleGAN [21], we further employ translations of real-to-synthetic and synthetic-to-real to realize cycle consistency. We train RCPNet by three schemes to demonstrate the effect of the cycle-consistent adversarial network [21], namely: train on mixed images and test on real images, train on synthetic-to-real images and test on real images; train on synthetic images and test on real-to-synthetic images.

RCPNet is first compared with other learning-based pose estimation approaches PoseNet [11, 55] and RPNet [7], using the real images from two datasets, namely, *Tuebingen Buildings* and *Cambridge Landmarks*. We then compare the accuracy between the real images and synthetic images trained RCPNet models on the two datasets. We also test our model in the indoor dataset *7Scenes* [22] to demonstrate its generalization ability.

Our main contributions can be summarized as follows:

- First, we develop RCPNet to estimate relative camera pose in multiple urban outdoor environments;
- Second, we build the *Tuebingen Buildings* dataset with drone collected images. We further expand it by rendering synthetic images from 3D models produced by the SfM method;
- Third, we take advantage of synthetic images to reduce the human labor in dataset generation and improve the performance of RCPNet further with domain adaptation via image-to-image translation.

The rest of the paper is as follows: Section 2 lists the related work. Section 3 presents the method for relative camera pose and domain adaptation via cycle-consistent adversarial networks. Section 4 introduces the data collection and preparation. Section 5 compares the experimental results of RCPNet on a two-stage analysis. In the end, Section 6 concludes the paper.

2 Related Work

2.1 Visual Localization

The visual localization usually includes three tasks [12, 23]: i) Relative camera pose estimation between consecutive keyframes as visual odometry, ii) relative camera pose estimation between the query and the reference images to eliminate the drift of localization in back-end optimization, iii) image matching to recognize viewed places in loop closure. We classify the first two as metric localization and the last one as topological localization.

Topological Localization Given a set of images with known locations and a query image, different feature matching methods will be used to retrieve the images with the closest distance or the most similar appearance. These methods

have successfully relocalized the camera from Google Street View [24], aerial views [25], or satellite imagery [26] to a known location roughly.

Suenderhauf et al. [27] use CNN features as robust landmark descriptors, which can recognize the camera locations under severe changes in viewpoints and other conditions. To achieve cross-view geo-localization, Workman et al. [25] and Vo et al. [28] collected two aerial-ground image datasets (called CVUSA, Vo and Hays). CVM-Net [29] uses Siamese network and NetVLAD [9] to achieve robust cross-view image matching based on the above two datasets.

Majdik et al. [23] collected a dataset in the center of Zurich, Switzerland by a drone that flew a trajectory of two kilometers, to achieve UAVs localization from street view images in GPS-denied urban environments. The dataset includes 113 discrete street view locations and 405 matching aerial images. The above topological localization methods cannot provide continuous and accurate pose estimation but only limited and discretized position estimation for the query image.

Metric Localization The ultimate goal of automated robot applications is to continuously and accurately locate new images in a known environment or map. Using point-based features to create sparse or dense environment maps is known as the classic visual SLAM methods [30, 31]. By combining data from cameras and LIDAR to construct a map, Pascoe et al. [32] realized real-time localization of cameras.

Kendall et al. [11] built the *Cambridge Landmarks* dataset and replaced the three softmax classifiers of GoogLeNet [33] with affine regressors to output poses. They used the SfM method to obtain the images' poses and trained an end-to-end CNN to regress the absolute camera poses, setting a precedent in 6DoF camera relocalization. Based on nearest-neighbor matching and consecutive learning feature descriptors, RelocNet [34] introduced a CNN representation method for camera pose retrieval. To learn a more discriminative regression function, Naseer and Burgard [12] generated synthetic viewpoints and corresponding depth maps to augment dataset. Walch et al. [13] collected the *TMU-LSI* dataset and demonstrated that the classic methods failed in a textureless environment. They also presented the CNN + long short term memory (LSTM) architecture, which performs camera pose regression by modeling the context of the image.

Melekhov et al. [35] proposed a system to solve the relative camera pose estimation problem. The system used a hybrid CNN with fully-connected layers (FCs) as a pose estimator. However, comparisons are difficult since their predicted translations are not full but scaled vectors. As far as we know, all existing camera pose regression methods (absolute and relative methods) that trained and tested in certain location need to improve the generalization ability. Sattler et al. [36] regard absolute camera pose regression as topology localization rather than metric localization. The results show that absolute pose regression within a scene is different from accurate pose estimation based on the 3D structure but more similar to pose approximation with image retrieval. Combining local SfM reconstruction with 2D model-based methods, [37] solved the problem of metric localization, because they believed that building broad-scale 3D models for real-time localization is still a major challenge. On the contrary, our opinion is that one trained model can estimate the relative camera poses in multiple locations, and it is sufficient to perform 3D reconstruction before the training phase.

In addition to visual SLAM, RCPNet can also estimate pose changes between consecutive keyframes, which can be used in some visual odometry systems, such as DeepVO [38], VINet [39] and VINS [40]. RCPNet can also be used for a group of aerial or/and ground robots to directly obtain the relative pose between robots in a centralized or decentralized manner.

2.2 Domain Adaptation

In this work, the relative camera pose estimation model is first trained on real images collected by a hand-held smartphone or a drone. Although the SfM method reduces the labor of labeling the images and offers accurate 6DoF pose for each image, the data collection and SfM procedure are very time-consuming and expensive. Since we can easily download 3D models for some famous landmarks, or build one with several images captured towards the target location, it is a natural idea to consider rendering synthetic images with related poses from 3D models for camera pose estimation network training. However, due to domain bias [41, 42], a system adapted to one dataset usually cannot be generalized to another. As a result, even if the camera pose estimation network can predict the relative pose for synthetic image pairs, it can not directly predict the relative pose for real image pairs, which will make the model impractical.

Domain adaptation aims to solve the dataset/domain bias problem. One strategy is to learn the domain-invariant features [43, 44], while others learn a feature or pixel level mapping from source to target domains [45, 46]. Minimizing the Maximum Mean Discrepancy (MMD) is popular to align feature distribution across the target and source domains [18, 47]. The adversarial loss brings representations that force the generated images (or translated images) to be indistinguishable from photos in the target domain [43, 48].

The image-to-image translation is a promising way to realize domain adaptation. Isola et al. [49] presented the

pix2pix which uses a generative adversarial network [50] for a translation from source to target photos. Similarly, Sangkloy et al. [51] generate photos from sketches while Karacan et al. [52] from the attribute and semantic layouts. Paired training examples are necessary for the above prior works, while CycleGAN [21] realizes unpaired image-to-image mapping via adversarial networks with cycle-consistency loss [53]. Our approach builds on CycleGAN while keeping the training images in a paired way to retain the geometry alignment between the synthetic and real images from the same viewpoint.

3 Method

3.1 Relative Camera Pose Estimation Model

In this part, we discuss the relative camera pose model and the architecture of RCPNet. RCPNet will output the relative pose vector p of the two input images. p is consist of a 3D relative camera translation t and a quaternion rotation q:

$$p = [t, q]. \tag{1}$$

We chose quaternions to represent the rotation because by normalizing them to unit length, it is easy to map a 4-D value to a reasonable rotation.

3.1.1 Learning Relative Translation and Rotation Simultaneously

We generate the two cameras' relative pose for training and testing. (R_1, t_1), (R_2, t_2) are the rotation matrices and translation vectors which project a point from world coordinate to camera 1 and 2's systems, respectively. From camera coordinates 1 to 2, we set P_{12} as the transformation matrix, R_{12} as the rotation matrix, and t_{12} as the translation vector:

$$P_{12} = \begin{bmatrix} R_{12} & t_{12} \\ 0 & 1 \end{bmatrix}; \quad \begin{cases} R_{12} = R_2 R_1^T, \\ t_{12} = R_1(t_2 - t_1). \end{cases} \tag{2}$$

Take (q_1, q_2, q_{12}) as quaternion representations of (R_1, R_2, R_{12}). A quaternion can represent a 3D rotation and is defined by 4 real numbers. x, y, and z represent a vector. w is a scalar that stores the rotation around the vector. Because the unit quaternions q and $-q$ represent the identical rotation, we perform a numerical inversion of all q_{12} with negative w to make the prediction of the network more consistent.

Training translation and rotation regressor separately will affect each other's performance [11]. Therefore, the original framework uses stochastic gradient descent to optimize the following loss function, which minimizes the Euclidean distance between pose predictions ($\hat{t}$ and $\hat{q}$) and ground truth (t and q):

$$\mathcal{L}(I) = ||\hat{t} - t||_2 + \beta||\hat{q} - q||_2. \tag{3}$$

To learn rotation and translation at the same time, they used grid search to fine-tune the weighting factor β to maintain a balance between translation and rotation errors. The result shows that the change interval of β in the outdoor scene is between 250 and 2000. Using cross-validation, RPNet [7] found the most suitable hyperparameter β value in different locations, and spends lots of time clustering the original dataset and testing the trained model for the evaluation. For RCPNet, we use automatic weights that scale on the loss function based on homoscedastic uncertainty (as in [55]) across all the locations, which is numerically more stable than β. In this loss function, the weighting factor β between the translation and rotation error is not static but adaptive during the whole training process. More precisely, if the translation estimation is more accurate than rotation in training, there will be a larger penalty for rotation error in the next epoch, and vice versa:

$$\mathcal{L}_\sigma(I) = \mathcal{L}_t(I)exp(-\hat{s}_t) + \hat{s}_t + \mathcal{L}_q(I)exp(-\hat{s}_q) + \hat{s}_q, \tag{4}$$

where $\mathcal{L}_t$ represents the translation loss and $\mathcal{L}_q$ represents the rotation loss. To maintain the balance between the penalty values of translation and rotation, factors $\hat{s}_t$ and $\hat{s}_q$ are used to force that the error of rotation and translation is not skewed. Given valid values for variance, the exponential mapping allows the regression to unconstrained scalar values since $exp(-s_i)$ is resolved to the positive domain. We set $\hat{s}_t = 0.0$, $\hat{s}_q = -3.5$ as initialization (approximately to β starts from 30, but fine-tuned during training) for all datasets. The adaptive loss makes the model more generalized to adapt to different locations.

3.1.2 Architecture of RCPNet

Different from RPNet [7] and PoseNet [11] based on GoogLeNet, we use two branches of pre-trained ResNet34 networks [57] to construct a weight-sharing Siamese network [56]. The 6DoF relative camera pose is estimated end-to-end.

The relative camera pose estimator of RCPNet is based on FCs and *ReLU* [58] activations, as shown in Fig. 1. We empirically use the output of the second to last layer as a 512-dimension (512D) global feature of each input image for every ResNet34 branch. The last pooling layer, and the 1000 units FC layer from the original ResNet34 are deleted. A 512D vector represents an image's geometrical features, and the distance between two 512D vectors denotes the spatial relationship between two corresponding images.

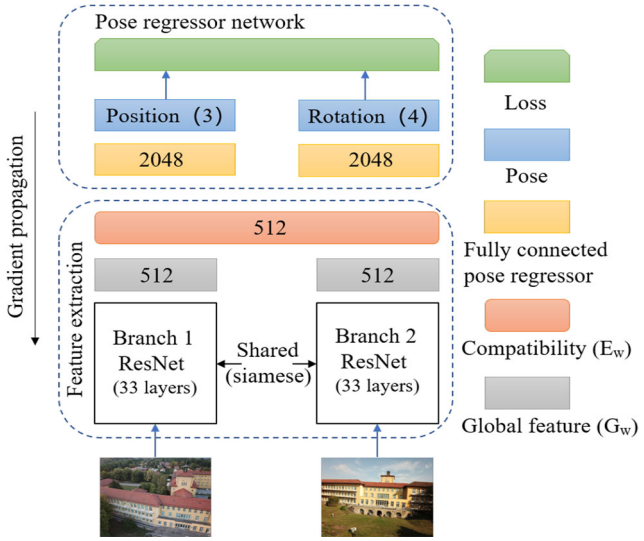


Fig. 1 RCPNet: the Siamese architecture based on ResNet34 branches with final regressors predicts two input images' relative translation and rotation. The blue boxes of pose denote the 3-dimension position vector and the 4-dimension quaternion rotation vector as the network's predictions

Following [56], the compatibility between images I_1 and I_2 is measured as:

$$E_w(I_2, I_1) = ||G_w(I_2) - G_w(I_1)||, \tag{5}$$

where $G_w(I_i)$ presents each image's global feature. In [56], the compatibility between images I_1 and I_2 represents the 'semantic' distance for image similarity metric learning in face verification. However, compatibility denotes the spatial distance between two overlapping outdoor images in this paper. More precisely, $E_w(I_2, I_1)$ is small if I_1 and I_2 are close in both position and rotation, and large if they are far away and have a different appearance.

Afterward, we insert two 2048D FCs as regressors to respectively output relative translation (3D) and rotation (4D). It is worth noting that they are learned together from the objective function of (4). We normalize the quaternion rotation vector to unit length.

3.2 Domain Adaptation via Cycle-Consistent Adversarial Networks

3.2.1 Bidirectional Adversarial Loss

In this paper, the real image denotes the RGB image captured from real environments by hand-held cameras or camera-mounted drones, which is limited and difficult to collect and label. The synthetic image represents the image rendered from 3D models with specific viewpoints, with less time and human labor consumption. Since the synthetic images are easier to obtain for training and real images are the target objects for relocalization testing, we also refer to the synthetic image as the source domain and the real image as the target domain. We aim to learn bidirectional mapping functions G_{s2t} and G_{t2s} to bridge the gap between the synthetic (source) domain X_s to the real (target) domain X_t. D_s and D_t are two adversarial discriminators following [50], where D_s is used to discriminate photos $\{G_{t2s}(x_t)\}$ from photos X_s, D_t aims to distinguish $\{G_{s2t}(x_s)\}$ from X_t. The adversarial losses are expressed as:

$$\begin{aligned}\mathcal{L}_{GAN}(G_{s2t}, D_t, X_t, X_s) &= \mathbb{E}_{x_t \sim X_t}\left[\log D_t(x_t)\right] \\ &\quad + \mathbb{E}_{x_s \sim X_s}\left[\log(1 - D_t(G_{s2t}(x_s))\right], \\ \mathcal{L}_{GAN}(G_{t2s}, D_s, X_t, X_s) &= \mathbb{E}_{x_s \sim X_s}\left[\log D_s(x_s)\right] \\ &\quad + \mathbb{E}_{x_t \sim X_t}\left[\log(1 - D_s(G_{t2s}(x_t))\right].\end{aligned} \tag{6}$$

3.2.2 Cycle Consistency Loss

To constrain two adversarial producers G_{s2t} and G_{t2s} to generate required geometrical consistent images rather than random photos with the target domain style, we use a cycle-consistent loss. As illustrated in Fig. 2(a), for each photo x_s from domain X_s, the transfer cycle is able to bring x_s back to be indistinguishable from the original photo, i.e., $G_{t2s}(G_{s2t}(x_s)) \approx x_s$ and $G_{s2t}(G_{t2s}(x_t)) \approx x_t$. Therefore, the cycle-consistent loss is as follows:

$$\begin{aligned}\mathcal{L}_{cyc}(G_{s2t}, G_{t2s}) &= \mathbb{E}_{x_s \sim X_s}\left[||G_{t2s}(G_{s2t}(x_s)) - x_s||_1\right] \\ &\quad + \mathbb{E}_{x_t \sim X_t}\left[||G_{s2t}(G_{t2s}(x_t)) - x_t||_1\right].\end{aligned} \tag{7}$$

We utilize the open-source software Blender [62] to render synthetic images from different poses on each 3d model. For cycle-consistent adversarial network training, we render images at the exact pose of each real image to generate real-synthetic image pairs.

3.2.3 Full Objective

The joint loss function is as follows:

$$\begin{aligned}\mathcal{L}(G_{t2s}, G_{s2t}, D_s, D_t) &= \mathcal{L}_{GAN}(G_{s2t}, D_t, X_t, X_s) \\ &\quad + \mathcal{L}_{GAN}(G_{t2s}, D_s, X_t, X_s) \\ &\quad + \lambda \mathcal{L}_{cyc}(G_{s2t}, G_{t2s}),\end{aligned} \tag{8}$$

with λ selected empirically as $\lambda = 10$, which is the balancing factor between the two objectives. We need to solve:

$$G^*_{t2s}, G^*_{s2t} = \arg \min_{G_{t2s}, G_{s2t}} \max_{D_s, D_t} \mathcal{L}(G_{t2s}, G_{s2t}, D_s, D_t,) \tag{9}$$

3.2.4 Architecture of Adversarial Network

Based on the work in [54], the adversarial generator network is consist of two convolutional layers followed by

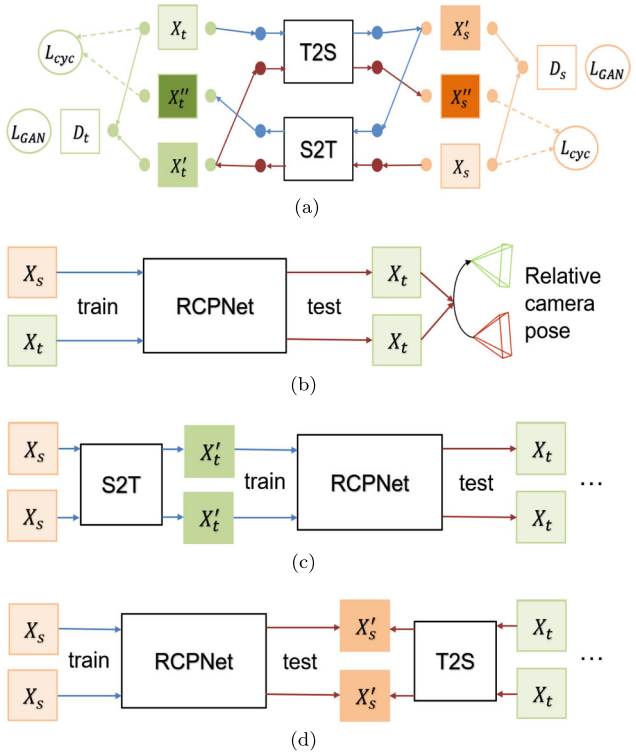


Fig. 2 (a) Our adversarial network architecture contains two image style translators: $G_{s2t} : X_s -> X_t$ and $G_{t2s} : X_t -> X_s$, and two discriminators D_s and D_t. D_t forces G_{s2t} to translate X_s into outputs indistinct from X_t, and vice versa for G_{t2s} and X_t. Two $\mathcal{L}_{GAN}$ are the objective bidirectional losses for adversarial training. Two $\mathcal{L}_{cyc}$ are the cycle consistent losses to further regularize the translators that encourage: $G_{t2s}(G_{s2t}(x_s)) \approx x_s$ and $G_{s2t}(G_{t2s}(x_t)) \approx x_t$. When utilizing the translated result of the cycle-consistent adversarial networks, we have adopted three implementation options: (b) real-synthetic mixed image pairs as input to train the RCPNet, implicitly learn the mapping between the two domains; (c) synthetic (domain X_s) images are firstly translate into synthetic-to-real images before the pairwise training of RCPNet. The query images from the real domain can be directly fed in the trained model; (d) synthetic image pairs are directly fed into RCPNet for training. The real query images need to be translated into real-to-synthetic images before testing

nine residual-blocks [57]. Two up-convolutional layers are inserted afterward to up-sample the image to the input size. The discriminator networks are adopted from PatchGANs [49].

4 Data Collection and Preparation

In this chapter, we introduce how to build a dataset for camera relocalization in urban outdoor environments. Using drones to capture images, we further extend the database to a 3D space with vertical viewpoint changes. We use SfM to generate ground truth poses for more than 10,000 real-world photos. By providing more image matches and a wider range of rotation and translation, the dataset makes the training of the pose estimator more efficient.

4.1 Data Collection

We build the *Tuebingen Buildings* dataset from eight outdoor urban places. The dataset offers data to train and test the absolute and relative pose estimator in different urban environments. We use a DJI Mavic Pro drone to collect the dataset in multiple locations near Tubingen, Germany.

The drone was manually piloted at each location. By keeping the camera always facing the building, we collect images of the entire environment at variant flying heights ranging from 2 to 35 m. In each location, we carried out at least four flights to capture images under varying weather and lighting conditions. Although there are some clutters like vehicles and pedestrians, they have little effect on most images captured from the height above 5 m.

We use Pix4D Mapper [59] to generate image poses as ground truth measurement and training labels. The Table 1 shows the output uncertainty of the absolute camera position and orientation. The vertical variance of the viewpoints brings new 3D constraints, which leads to better localization: the position error is about 10 to 40 cm, and the orientation error is below 1°. Rather than recording videos and sub-sampling to frames, we programmed the drone to capture photos every two meters of movement (measured by GPS) in any direction. To obtain a better 3D reconstruction, images are captured with high resolution (4000×3000) from variant distances (see Fig. 3).

The eight scenes in the dataset (see the examples in Fig. 4) are diverse: i) different styles of modern and traditional; ii) some buildings have repeated concrete structures, and others are surrounded by vegetation; iii) the trajectories have diverse shapes. For example, convex trajectories around central buildings and concave trajectories in a courtyard surrounded by multiple structures.

We use four scenes (*King's College*, *Old Hospital*, *Shop Facade*, *St Mary's Church*) from the *Cambridge Landmarks* dataset. The images are collected from the ground views by a handheld Google LG Nexus 5 smartphone. The difference is that the *Tuebingen Buildings* dataset contains large changes in viewpoint, which is quite general in UAV applications, especially the vertical direction variance.

We use the *7Scenes* indoor dataset [22] to demonstrate the generalization ability of RCPNet. This dataset was built with a Kinect RGBD camera in 7 separate scenes, and each scene consists of a single room. Using Kinect Fusion [22], ground truth poses were generated. The dataset contains many repetitive or texture-less features, which is exceptionally challenging for visual relocalization using traditional features.

Table 1 Mean uncertainties of absolute camera position and orientation

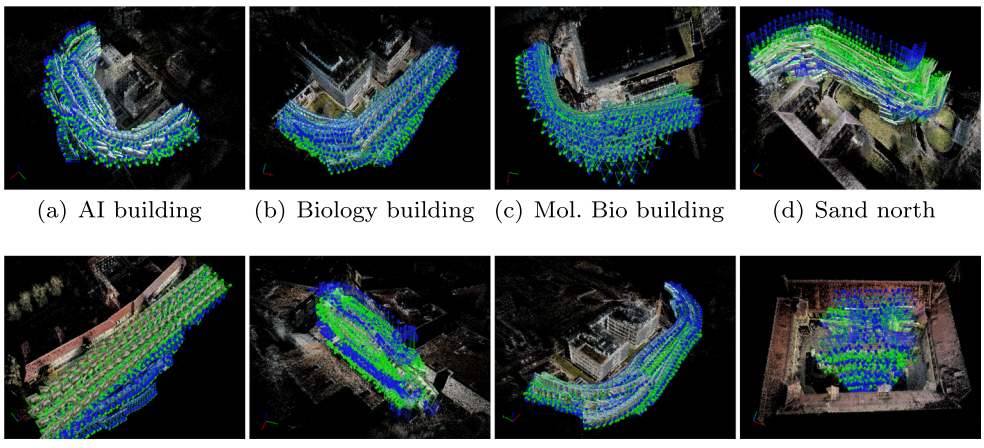
	X	Y	Z	Yaw	Pitch	Roll	Images
AI Building	0.136m	0.142m	0.223m	0.549°	0.089°	0.444°	1438
Biology B.	0.248m	0.246m	0.402m	0.377°	0.160°	0.266°	1209
Mol. Bio. B.	0.120m	0.125m	0.201m	0.236°	0.070°	0.146°	1112
Sand North	0.147m	0.140m	0.227m	0.381°	0.163°	0.326°	1504
Sand South	0.152m	0.138m	0.228m	0.073°	0.084°	0.026°	1035
Shopping M.	0.124m	0.126m	0.205m	0.256°	0.183°	0.191°	1537
Industrial B.	0.403m	0.356m	0.603m	0.302°	0.236°	0.157°	1302
Tue. Castle	0.088m	0.088m	0.152m	0.172°	0.092°	0.083°	1216

4.2 Real Image Pairs Preparation

An effective method is needed to produce image pairs to achieve relative camera pose estimation. For the *Cambridge Landmarks* dataset, En et al. [7] randomly paired every image with eight images in the same sequence. The training sequences and testing sequences are separated beforehand. When using this dataset for relative camera pose estimation, we followed their settings.

In contrast, for the *Tuebingen Buildings* dataset, we first separate images into training and testing subsets in each scene. Then we use SIFT feature matching to generate real image pairs for the subset, traversing each subset for every image in the subset. To further reject the outliers, some SIFT feature matched image pairs are filtered: if they have significant differences in translation (> 30 m) or rotation ($> 75°$), we check their co-visibility manually and delete the wrong matches. The thresholds are measured from the ground truth. The "co-visibility" information means what landmark objects are visible together to two cameras. At last, we obtained around 300,000 valid pairs in all eight scenes, an average of 30 pairs per image. The augmentation is as expected because our dataset covers a wider 3D space, and each images have many co-visible neighbors from many directions. Figure 5 demonstrates that the relative camera pose samples (translation and rotation ranges) in the *Tuebingen Buildings* dataset are more widely distributed.

The images are rescaled to $256 \times n$ or $n \times 256$ pixels, $n \geq 256$, and then are cropped into 224×224 patches as the input of CNN in the previous work [7, 11]. The model is trained by random cropping and then tested by central cropping as data augmentation. However, cropping operation (as shifting, flipping, rotating, and zooming) may affect the spatial information implicit in the image. For the relative camera pose estimation, we keep the random crop coherent in the two input photos, and then test multiple scenes with scale size from 256 to 236. The results indicate that for scenes with shorter object distances such as *Shop Facade*, a smaller rescale ratio will enable a wider field of view and a larger overlap, thus will slightly improve localization. For most of the scenes that are far away from the objects, the data augmentation effect using a bigger rescale ratio is dominant. This indicates that other data augmentation methods need be considered to keep the spatial information as much as possible. Following PoseNet [11], the tuning of brightness, contrast, saturation, and hue, combined with the crop operation, are also adopted in our framework. Same data augmentations have been applied to



(a) AI building (b) Biology building (c) Mol. Bio building (d) Sand north

(e) Sand south (f) Shopping mall (g) Industrial building (h) Tuebingen castle

Fig. 3 The cameras can cover the 3D model uniformly from variant angles and depths. The trajectories have distinct shapes. The raw camera poses provided by UAV are blue, and the camera poses generated by SfM are green

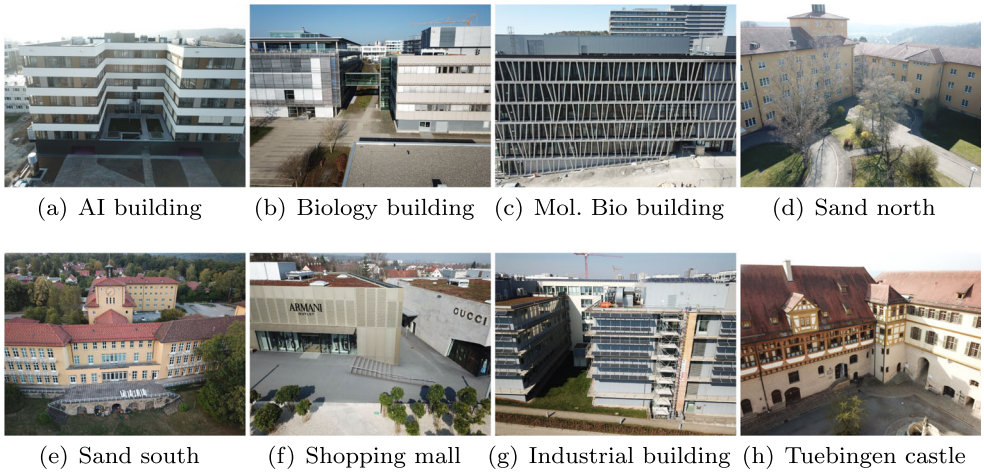


Fig. 4 The *Tuebingen buildings* dataset contains eight scenes, including modern and classical, repeated concrete structures, and natural urban environment appearance. Captured by a manually-flying drone, flying at a height from 2 to 35 m

both real and synthetic images before fed into the network for training.

Our method is first evaluated separately on four scenes of *Cambridge Landmarks*. Meanwhile, we selected three scenes (*King's College*, *Old Hospital*, *St Mary's Church*) for across-scenes training while maintaining *Shop Facade* invisible. We use the same split of training and test as RPNet [7]. Similarly, for *Tuebingen Buildings*, a RCPNet model is firstly trained and tested in each scene, as individual training. Afterward, we train a model with six scenes together (*AI Building*, *Biology Building*, *Mol. Bio Building*, *Sand North*, *Shopping Mall*, and *Tuebingen Castle*) and test RCPNet in each scene, as across-scenes training.

For individual training, we separate the data in each scene randomly into training and test sets at a ratio of 4:1. For across-scenes training, we maintain the test sets unchanged, and extract 20,000 image pairs from the training set of each scene to obtain a 120k pairs training set. Some spatially close photos may be separated into the test and training sets, but these images are completely different, and the similarity among image pairs is very low because they are matched from multiple directions.

For the *7Scenes* dataset, we randomly choose two paired images for the target one from the near frame in the same sequence. We cut off the nearest frames (e.g., in $+/-10$ frames) to ensure enough pose shift and cap on the threshold of the 25th frame to ensure co-visibility. We select 5 of the scenes for across-scene training, and 2 of the scenes for individual training, obtaining very close performance.

4.3 Synthetic Images Generation

We use ContextCapture [60] to build 3D models with poses from two different datasets. For four scenes in the *Cambridge Landmarks* dataset, e.g., *Shop Facade*, *St Mary's Church*, *Old Hospital*, and *King's College*, we build 3D models with poses (obtained by VisualSfM [61]) from the original dataset. For the *Tuebingen Buildings* dataset, we choose six scenes, e.g., *Tuebingen Castle*, *Shopping Mall*, *Sand North*, *Mol. Bio Building*, *Biology Building*, and *AI Building*, to build fine 3D models while keeping the original poses we obtained with Pix4D in Section 4.2. We randomly select 300 images (*Shop Facade* with 200) for each scene to build the model, which is around 25% images of each training scene.

As mentioned in Section 3.2.2, we utilize the open-source software Blender [62] to render synthetic images. First, we render images at the exact pose of each real image. This step

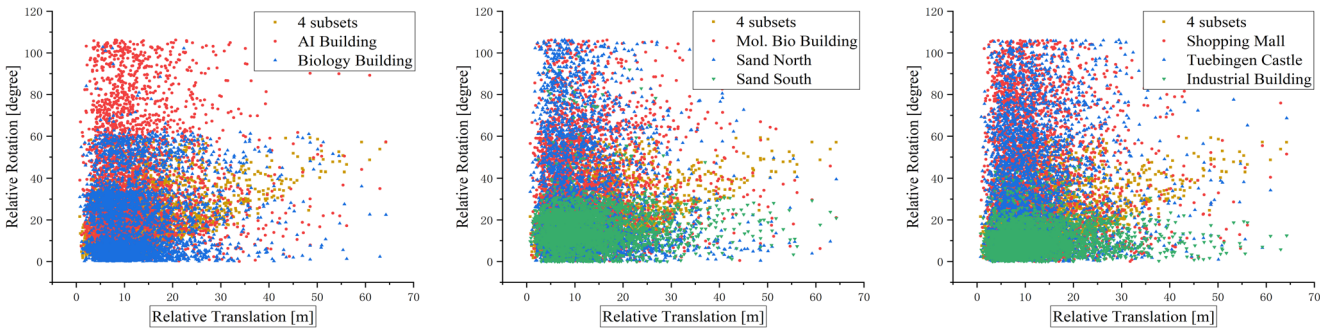


Fig. 5 The relative camera pose distributions of the eight scenes in the *Tuebingen Buildings* dataset are compared with the four scenes together in the *Cambridge Landmarks* dataset on translation and rotation. The *Tuebingen Buildings* dataset has advantages in data diversity and density

is to generate real-synthetic image pairs for cycle-consistent adversarial network training. It is also an excellent proof for the fine quality 3D models generating and accurate pose labeling. Some examples are shown in the first and third column of Figs. 6 and 7.

Second, we randomly generate twice the amount of initial training poses within the spatial area of each scene (see the fourth column in Table 2). The camera intrinsics are calibrated during the 3D model building. For the *Cambridge Landmarks* dataset, the height of poses is set as $z < 2m$ to simulate the human's viewpoint, while for *Tuebingen Buildings*, we set the threshold of the highest viewpoint in the training set. To ensure the generated poses are facing toward the 3D model, we set a threshold for Euler angle difference between a generated pose and its nearest real pose at 10°. Each rendered image in a new pose is then paired with around 30 images via a co-visibility check.

As a result, we use an average of 300 images to build a 3D model for an urban scene and generate an average of 3000 synthetic images for training. The later result proves that this scenario promises to generate training labels for visual localization and other visual tasks while requiring minimal human labor consumption.

4.4 Training of Cycle-Consistent Adversarial Networks for Domain Adaptation

We train two cycle-consistent adversarial networks separately on the *Tuebingen Buildings* and the *Cambridge Landmarks* datasets. To maintain the geometry-alignment between images from two domains, each real-synthetic image pair is extracted from the same pose. For each dataset, we randomly select 500 real-synthetic image pairs as a new train set and 150 real-synthetic pairs as a validation set from the initial train set, and 150 real-synthetic pairs are extracted from the initial test set as a new test set. Every sub-scene has an equal proportion in each set.

Figures 6 and 7 demonstrate the qualitative results of the cycle-consistent adversarial networks trained on *Tuebingen Buildings* and *Cambridge Landmarks*, respectively. In each figure, T (target), $S2T$, S (source), and $T2S$ represent the real image, synthetic-to-real image, synthetic image, and real-to-synthetic image. The last row is a typical challenging failure example. We focus on the comparisons of $T - S2T$ and $S - T2S$.

First of all, due to the fine 3D models and accurate pose labels, the cycle-consistent adversarial network preserves the geometric details of the generated and translated images to the greatest extent, e.g., the second and fifth rows in Fig. 6, and the second and the third row (pay attention to the dog at the bottom!) in Fig. 7. The buildings' skyline is well preserved by the synthetic-to-real mapping in many cases except for the most complex buildings. Some small moving objects, like pedestrians, can be eliminated by both direction translation, see the right bottom corner of the images in the first row of Fig. 7. This is an excellent feature for visual localization tasks. However, if the occluded objects are too large or too close to the camera, like the tree in the last row of Fig. 6 or the bus in the last row of Fig. 7, eliminating them is beyond of the capability of our image style translation method.

The illumination in the real domain is always varying. If the distribution of the query image is identical with the training set, the result is consistent, e.g., the fifth and sixth rows in Fig. 7, if not, then the brightness of the real image and related synthetic-to-real image is inconsistent, e.g., the fourth row in Fig. 7. Naturally, we can imagine that seasonal changes will bring similar results. However, synthetic images and real-to-synthetic images do not have such trouble. We can observe it from the last two columns in Figs. 6 and 7. This difference gives us an interesting hint. In Fig. 2, three options for utilizing the translated images have been discussed. The $S2T$ Training scenario could directly use real query images, thus saving time during the inference stage. However, the $T2S$ Test scenario is promising in generalization ability due to the consistent feature distribution between synthetic and real-to-synthetic images. Further details will be discussed in the next section.

5 Experiments

5.1 Experimental Setup

We obtain a large number of synthetic images in Section 4.3. We utilize these synthetic images in three scenarios, as shown in Fig. 2(b) to (d): (1) mixed real-synthetic pair consisting of a synthetic image and a real image is fed into RCPNet for training, forcing it to implicitly learn the mapping between the two domains; (2) synthetic (domain X_s) images are first translated into synthetic-to-real images before the pairwise training of RCPNet. In the inference stage, the real query image pairs can be directly fed in the trained model; (3) in the training stage, synthetic image pairs are directly fed into RCPNet. The query images from the real domain need to be translated into real-to-synthetic images before testing. The number of training samples is doubled with the three scenarios mentioned above, compared with the initial real image dataset.

Now we present the quantitative evaluation of different methods on multiple datasets. As benchmarks, we also test PoseNet [11] with absolute camera pose regression and RPNet [7] with relative camera pose estimation on eight scenes in the *Tuebingen Buildings* dataset. For the *Cambridge Landmarks* dataset, we quote the results from their papers. We also cite the results of PoseNet [11,

Fig. 6 Examples of image style transfer result compared with original synthetic and real images in *Tuebingen Buildings* dataset. From left to right, T, $S2T$, S, and $T2S$ represent the real image, synthetic-to-real image, synthetic image, and real-to-synthetic image. The last row is a typical challenging failure example

55] and RelocNet [34] on the indoor dataset *7Scenes* as baselines. The translation errors are measured in meters and rotation errors are in degrees, as the cited works did. Since the *Tuebingen Buildings* and *Cambridge Landmarks* datasets only provide discrete images with ground truth poses for testing, we evaluate RCPNet with discrete image relocalization in this paper instead of path tracking of robot-captured motion videos. However, relative camera pose

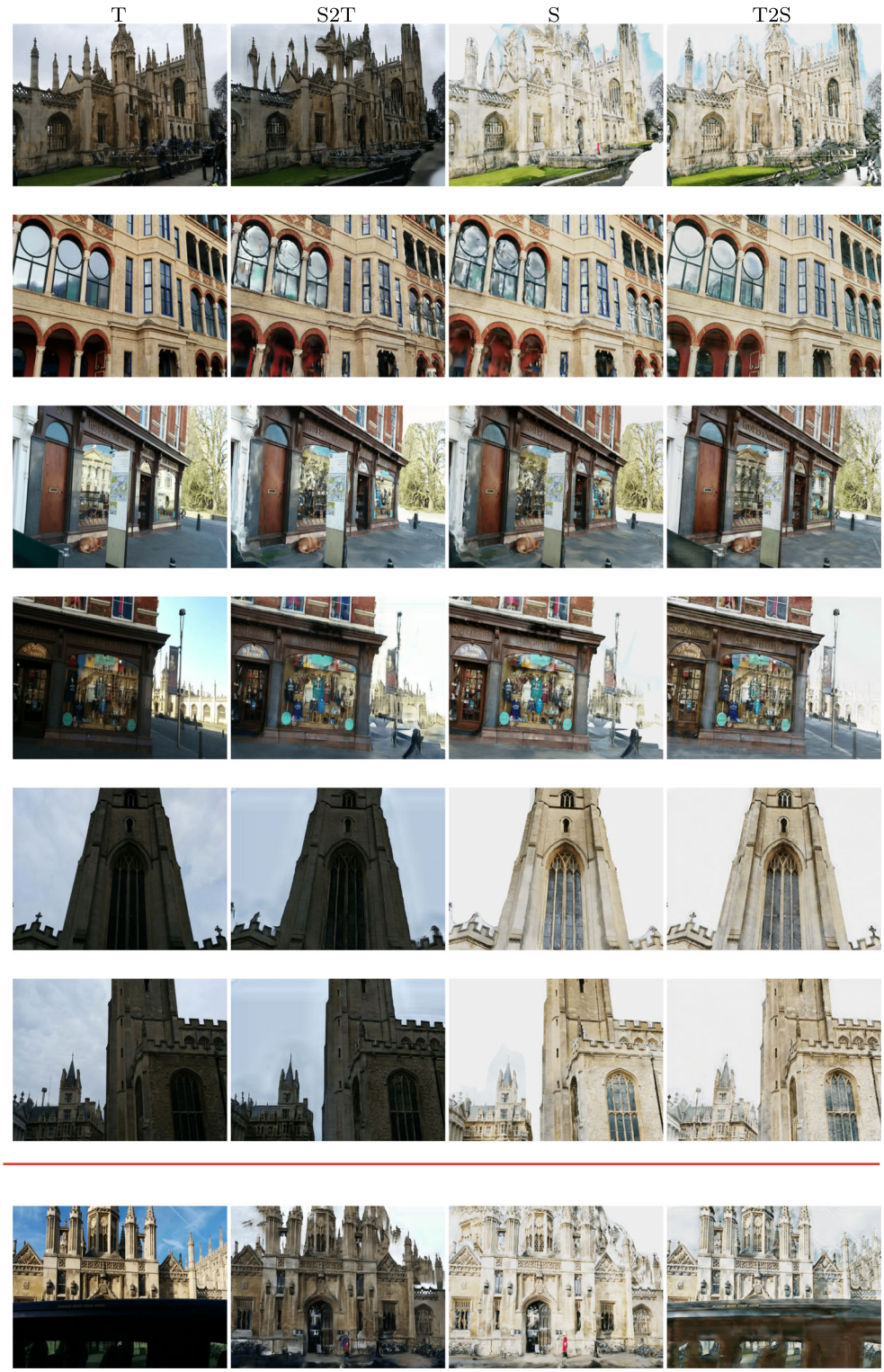


Fig. 7 Examples of image style transfer result compared with original synthetic and real images in *Cambridge Landmarks* dataset. From left to right, T, $S2T$, S, and $T2S$ represent the real image, synthetic-to-real image, synthetic image, and real-to-synthetic image. The last row is a typical challenging failure example

estimation, combined with the global matching method, has a great potential to realize long-term camera relocalization in large environments.

We normalize pixel intensities of the input images from range -1 to 1. We use an implementation in PyTorch [63] to train RCPNet and the adversarial networks. We use ADAM [64] for optimization with a learning rate of 1×10^{-4}. For individual trained RCPNet, we use a batch size of 32 on an NVIDIA 1080Ti GPU; training takes 20k - 100k iterations, i.e. 10 hours - 2 days. We use a batch size of 128 for across-scenes trained RCPNet on two NVIDIA 1080Ti GPUs, and training takes 2 days. The learning rate of the adversarial

networks is constant for 100 epochs and linearly decreases to zero over the last 100 epochs. 500 real-synthetic image pairs with 200 epochs training takes 6 hours on an NVIDIA 1080Ti GPU.

5.2 Experimental Results

5.2.1 Results of Real Images Training

In Table 2, we compare different approaches on each scene, individually or across-scenes, seen or unseen. The baselines are PoseNet trained by images and RPNet trained by real image pairs, both within each scene. Generally, the accuracy of absolute pose regression cannot be directly compared with relative pose estimation. However, since PoseNet [11] is the basis of RPNet [7] and RCPNet, we think it is a valid reference. When one implements relative camera pose estimation in a real environment, it is common to estimate the relative pose between an unknown query image and a known reference image with a ground truth pose. Under this circumstance, the relative pose estimation accuracy is equal to the absolute pose regression accuracy for the query image. When we compare the right four columns in Table 2, the result shows the individually trained RCPNet outperformed the other two baselines in most scenes. The results of across scenes trained RCPNet in the last column of Table 2 has an average 5% decline compared with the results of individual trained RCPNet in both datasets, but it is still comparable to PoseNet and RPNet in most scenes. In *King's college* and *Shopping Mall*, the across-scenes trained model has even better performance than the individual trained model. The comparison between individual training and across-scenes training shows that different scenes may have general features, and a fine-tuned model can adapt to them simultaneously.

For invisible scenes (*Industrial building*, *Sand South* and *Shop Facade*), the across-scenes model failed, which indicates the limitation of PoseNet-based architecture. Some recent findings [36, 37] on image-based localization show that PoseNet design can degrade the generalization ability in challenging scene variation due to scale-inconsistency. Although the PoseNet-based methods have limited generalization ability, it is still meaningful to expand from individual scene training to across-scenes training. For example, when a vehicle works in a factory that consists of several workshops, or a drone delivers products between multiple GPS-denied places in a city, the robot with a single model can be competent if it was across-scenes trained.

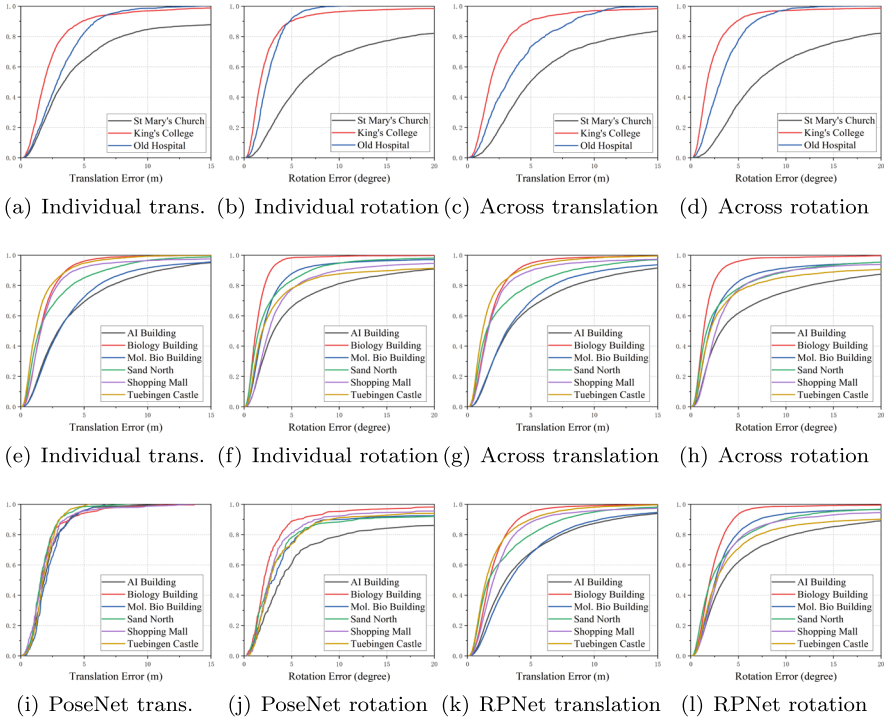
The cumulative distribution functions (CDF) of Fig. 8 show the absolute/relative camera localization errors by PoseNet, RPNet, and RCPNet with individual or across-scenes training in different scenes of *Tuebingen Buildings* and *Cambridge Landmarks*. Individual trained RCPNet performs better in rotation with an average 31.4% increase to PoseNet and 24.2% to RPNet, while on *AI Building*, *St Mary's Church*, and *Mol. Bio building*, PoseNet or RPNet performs better in translation. Comparing with the individual-trained models' performance within each scene, an across-scenes-trained model's performance has an average 5% decline in both translation and rotation.

Table 2 Results with real images training and dataset details: median absolute/relative camera localization errors for the *Cambridge Landmarks* and *Tuebingen Buildings* datasets

	Frames		Pairs		Spatial	(Absolute)	(Relative)	(Individual)	(Across)
Scene	Test	Train	Test	Train	Extent(m)	PoseNet [11]	RPNet [7]	RCPNet	RCPNet
King's C.	343	1220	2424	9227	140×40	1.92m, 5.40°	1.93m, 3.12°	1.85m, **1.72°**	**1.80m**, **1.72°**
Old Hospital	182	895	1228	6417	50×40	**2.31m**, 5.38°	2.41m, 4.81°	2.87m, **2.41°**	3.15m, 3.09°
St Mary's C.	530	1487	3944	10736	80×60	2.65m, 8.48°	**2.29m**, **5.90°**	3.43m, 6.14°	3.84m, 6.93°
Shop Facade*	103	231	607	1643	35×25	**1.46m**, 8.08°	1.68m, **7.07°**	1.63m, 7.36°	13.8m, 28.6°
AI Building	288	1150	9326	38549	145×90×28	**1.87m**, 3.84°	3.01m, 3.47°	2.94m, **3.10°**	3.22m, 3.21°
Biology B.	242	967	8589	34421	120×95×26	1.58m, 2.12°	1.73m, 2.02°	**1.53m**, **1.24°**	1.58m, 1.32°
Mol. Bio B.	223	889	10492	41528	190×95×25	**2.03m**, 3.15°	3.36m, 2.59°	3.02m, **1.95°**	3.09m, **1.95°**
Sand North	301	1203	6680	27315	100×45×23	1.57m, 2.65°	1.67m, 2.15°	**1.45m**, **1.52°**	1.50m, 1.66°
Shopping M.	308	1229	4991	20412	50×55×13	1.66m, 2.75°	2.05m, 2.77°	**1.58m**, 2.66°	1.63m, **2.64°**
Tue. Castle	244	975	5787	23182	40×35×21	1.64m, 2.80°	1.47m, 2.69°	**1.16m**, **1.92°**	1.19m, 2.01°
Sand South*	207	828	6276	25729	190×50×37	2.06m, 2.27°	1.52m, 6.64°	**1.17m**, **2.64°**	17.6m, 21.3°
Industrial B.*	261	1041	9910	40329	170×60×33	1.75m, 2.69°	1.34m, 1.68°	**1.12m**, **1.37°**	16.7m, 18.4°

*Invisible scenes in across-scenes training

The results with the best performance of each scene have been marked as bold



(a) Individual trans. (b) Individual rotation (c) Across translation (d) Across rotation

(e) Individual trans. (f) Individual rotation (g) Across translation (h) Across rotation

(i) PoseNet trans. (j) PoseNet rotation (k) RPNet translation (l) RPNet rotation

Fig. 8 These cumulative distribution functions (CDF) of errors for scenes in *Cambridge Landmarks* (a-d) and *Tuebingen Buildings* (e-l) show localization accuracy of four approaches: the individual trained RCPNet (a, b, e, f), the across-scenes trained RCPNet (c, d, g, h), PoseNet (i, j), and RPNet (k, l). PoseNet performs better in *AI Building* and *Mol. Bio building* scenes, while the individual trained RCPNet leads in other scenes. The across-scenes training keeps up with the individual training in most scenes

In Table 3, we compare our results with PoseNet and RelocNet on indoor dataset *7Scenes*. RelocNet uses multiple candidates for the relative pose regressor in four scenes. We train an across-scenes model across five scenes (*Chess, Fire, Heads, Pumpkin,* and *Stairs*), while individually train two models for *Office* and *Red Kitchen*. The performance of RCPNet is comparable to the three baselines: an average 53.5% and 35.7% increase in translation and rotation accuracy to PoseNet (β weight) [11]; an average 9.3% and 21.3% increase in translation and rotation accuracy to PoseNet (Geometric) [55]; an average -0.7% decrease and 3.6% increase in translation and rotation accuracy, respectively, compared with RelocNet (*7Scenes*) [34]. In the small indoor dataset, the translation accuracy may have reached the upper limit for PoseNet-like methods, but there is room for improvement in rotation accuracy.

5.2.2 Results of Synthetic Images Training

Table 4 demonstrates the quantitative results of RCPNet with across training using synthetic images in the three scenarios, as we discussed in Section 5.1 and shown in Fig. 2. Mixed Input denotes that in the training stage, real-synthetic image pairs are fed into the RCPNet. *S2T* Training represents that in the training stage, synthetic-to-real image pairs are fed into the RCPNet, then the real query image can be tested directly. *T2S* Test means that real-to-synthetic images are tested with a model trained by synthetic image pairs.

Table 3 Median localization errors for the *7Scenes* [22] dataset, with real images training

Scene	PoseNet (β weight) [11]	PoseNet Geometric [55]	RelocNet 7Scenes [34]	**RCPNet**
Chess[1,2]	0.32m, 6.60°	0.13m, 4.48°	**0.12m**, 3.95°	0.13m, **3.46°**
Fire[1,2]	0.47m, 14.0°	0.27m, 11.3°	**0.25m**, 10.1°	0.31m, **9.45°**
Heads[1,2]	0.30m, 12.2°	0.17m, 13.0°	**0.13m**, 10.5°	0.15m, **9.87°**
Office	0.48m, 7.24°	0.19m, 5.55°	0.18m, 5.32°	**0.17m**, **4.81°**
Pumpkin[1]	0.49m, 8.12°	0.26m, 4.75°	0.26m, **4.17°**	**0.22m**, 4.39°
Red K.	0.58m, 8.34°	0.23m, 5.35°	0.23m, **5.08°**	**0.21m**, 5.53°
Stairs[1,2]	0.48m, 13.1°	0.35m, 12.4°	0.27m, 7.31°	**0.26m**, **7.24°**

[1]These scenes are trained together in an across-scenes model in RCPNet

[2]These scenes fuse multiple nearest neighbors before the relative pose regression in RelocNet

The results with the best performance of each scene have been marked as bold

Table 4 Results of RCPNet with across training: median relative localization errors for the *Cambridge Landmarks* and *Tuebingen Buildings* datasets, with 3 schemes using synthetic images

Scene	Mixed input	S2T training	T2S test
King's C.	1.64m, 1.49°	1.76m, 1.53°	**0.88m**, **1.05°**
Old Hospital	2.56m, 2.37°	2.61m, 2.46°	**1.92m**, **2.19°**
St Mary's C.	2.87m, 4.94°	3.22m, 5.82°	**2.14m**, **2.93°**
Shop Facade	1.48m, 5.35°	1.55m, 6.41°	**1.43m**, **4.11°**
AI Building	2.63m, 2.90°	2.94m, 3.10°	**1.99m**, **2.71°**
Biology B.	1.41m, 1.18°	1.53m, 1.24°	**1.24m**, **1.09°**
Mol. Bio B.	2.84m, 1.73°	3.02m, 1.95°	**2.57m**, **1.35°**
Sand North	1.39m, 1.53°	1.45m, 1.52°	**1.16m**, **1.27°**
Shopping M.	1.46m, 2.52°	1.58m, 2.67°	**1.23m**, **2.16°**
Tue. Castle	1.16m, 1.90°	1.15m, 1.92°	**1.02m**, **1.85°**

The results with the best performance of each scene have been marked as bold

Compared with the results of across-scenes real image trained RCPNet in the last column of Table 2, the performance of the three scenarios mentioned above has been improved due to the expansion of training data. The mixed input scenario has an average 14.5% and 16.2% improvement translation and rotation accuracy, respectively. It shows the network implicitly learns the bidirectional mapping between two domains from mixed real-synthetic image pairs. The $S2T$ training scenario has a slightly inferior performance to Mixed Input, which shows that the feature distribution in real-world images is always varying due to the illumination shifts and seasonal changes. Therefore, it's challenging to find an average mapping representation from the consistent synthetic distribution to the inconsistent real distribution. However, the $S2T$ training still has an average 8.3% and 9.5% increase in translation and rotation accuracy, compared with the real image trained model.

The $T2S$ test scenario has an average 32.6% and 32.3% increase in translation and rotation accuracy to the real image trained RCPNet. It also outperforms the other two synthetic image trained competitors with a large margin, proving the robustness of the translation from the real distribution to the synthetic distribution. We need real-to-synthetic translation for every input query image in the $T2S$ test scenario during the test stage. The predicted poses are still compared to the real query images' poses. It caused an extra time-consuming but is still a meaningful example for synthetic images application. Besides, synthetic image training via domain adaptation in the other two scenarios also improves the original RCPNet. For visual localization tasks, a promising bridge has been built between the real data and the synthetic data. As portable camera equipment becomes more and more popular, and 3D modeling becomes more convenient, this method will be more widely used in the future and reduce the heavy data labeling work of humans.

6 Conclusion

We present RCPNet, a learning-based method for relative camera pose estimation across multiple scenes. RCPNet can be used in continuous localization for autonomous navigation of unmanned systems, multi-robot cooperation systems, visual odometry applications, or combined with global image retrieval methods like NetVLAD to obtain more accurate absolute pose estimation. A camera relocalization dataset for both absolute and relative camera pose estimation is built with a drone and the SfM method. We demonstrate that our method outperforms two baseline approaches in two outdoor datasets. The result that across-scenes training has comparable performance to individual training shows that general features of image pairs in different locations exist. For unseen scenes, the results of the across-scenes model are not satisfactory, the PoseNet-based architecture needs to be further modified. We also test our model in an indoor dataset to demonstrate its generalization ability.

Besides, we further expand our dataset with 3D modeling and synthetic image rendering. We utilize cycle-consistent image style transfer and adversarial training to estimate real-image pairs' relative camera pose based on training over synthetic environment data in different schemes. The result demonstrates the effectiveness of the cycle-consistent adversarial networks in domain adaptation between real and synthetic images.

In future work, we aim to improve our network architecture to implement the image style transfer and pose estimation in an across-scenes model with fewer hyperparameters. To improve the practical ability in long-term camera relocalization, we will further test our method in larger and connected 3D datasets.

Author Contributions All authors have contributed to the concept and design of the research. Chenhao Yang provided the research ideas and the theoretical analysis, collected the dataset, and wrote the code and the paper. Yuyi Liu and Andreas Zell strictly revised and edited the previous manuscript. The final manuscript was read and approved by all authors.

Funding Open Access funding enabled and organized by Projekt DEAL. This research was supported by the German Federal Ministry of Education and Research (BMBF) project 'Training Center Machine Learning, Tuebingen' with grant number 01|S17054.

Availability of data and materials The original dataset is available under email request and could only be used for the non-commercial application.

Declarations

Competing interests The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: Dense tracking and mapping in real-time. In: 2011 International Conference on Computer Vision, pp. 2320–2327, Barcelona (2011)
2. Engel, J., Schöps, T., Cremers, D.: LSD-SLAM: Large-scale direct monocular SLAM. In: Fleet D., Pajdla T., Schiele B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014, pp. 834–849 (2014)
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
4. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)
5. Tian, Y., Chen, C., Shah, M.: Cross-view image matching for Geo-localization in urban environments. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1998–2006, Honolulu, HI (2017)
6. Lin, T., Yin, C.ui., Belongie, S., Hays, J.: Learning deep representations for ground-to-aerial geolocalization. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5007–5015, Boston, MA (2015)
7. En, S., Lechervy, A., Jurie, F.: Rpnet: An end-to-end network for relative camera pose estimation. In: Leal-Taixe, L., Roth, S. (eds.) Computer Vision – ECCV 2019 Workshops, pp. 738–745 (2019)
8. Deng, J., Dong, W., Socher, R., Li, L., Li, K.ai., Li, F.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009)
9. Arandjelovi, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J.: Netvlad: Cnn architecture for weakly supervised place recognition. IEEE Trans. Pattern Anal. Mach. Intell. **40**(6), 1437–1451 (2018)
10. Agarwal, S., Snavely, N., Simon, I., Sietz, S.M., Szeliski, R.: Building rome in a day. In: Twelfth IEEE International Conference on Computer Vision (ICCV), pp. 72–79 (2009)
11. Kendall, A., Grimes, M., Cipolla, R.: PoseNet: A convolutional network for real-time 6-DOF camera Re. In: 2015 IEEE International Conference on Computer Vision, ICCV, vol. 284, pp. 1980–1983 (2015)
12. Naseer, T., Burgard, W.: Deep regression for monocular camera-based 6-DoF global localization in outdoor environments. IEEE Int. Conf. Intell. Robot. Syst. **2017**, 1525–1530 (2017)
13. Walch, F., Hazirbas, C., Leal-Taixe, L., Sattler, T., Hilsenbeck, S., Cremers, D.: Image-based localization using LSTMs for structured feature correlation. Proc. IEEE Int. Conf. Comput. Vision **2017**, 627–637 (2017)
14. Le, T.A., Baydin, A.G., Zinkov, R., Wood, F.: Using synthetic data to train neural networks is model-based reasoning. arXiv:1703.00868 (2017)
15. Rajpura, P.S., Hegde, R.S., Bojinov, H.: Object detection using deep cnns trained on synthetic images. arXiv:1706.06782 (2017)
16. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field Serv. Robotics, pp. 621–635 (2017)
17. Li, Y., Wang, N., Liu, J., Hou, X.: Demystifying neural style transfer. arXiv:1701.01036 (2017)
18. Long, M., Ding, G., Wang, J., Sun, J., Guo, Y., Yu, P.S.: Transfer sparse coding for robust image representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 407–414 (2013)
19. Atapour-Abarghouei, A., Breckon, T.P.: Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 18, p. 1 (2018)
20. Chen, Y., Li, W., Sakaridis, C., Dai, D., Gool, L.V.: Domain adaptive faster r-cnn for object detection in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3339–3348 (2018)
21. Zhu, J.-Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proc. Int. Conf. Computer Vision, pp. 2242–2251 (2017)
22. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in RGB-D images. In: Computer Vision and Pattern Recognition (CVPR), pp. 2930–2937 (2013)
23. Majdik, A., Verda, D., Albers-Schoenberg, Y., Scaramuzza, D.: Air-ground matching: Appearance-based gps-denied urban localization of micro aerial vehicles. J. Field Robot. **32**, 1015–1039 (2015)
24. Anguelov, D., Dulong, C., Filip, D., Frueh, C., Lafon, S., Lyon, R., Ogale, A., Vincent, L., Weaver, J.: Google street view: Capturing the world at street level. Computer **43**(6), 32–38 (2010)
25. Workman, S., Souvenir, R., Jacobs, N.: Wide-area image geolocalization with aerial reference imagery. Proc. IEEE Int. Conf. Comput. Vision, Inter **2015**(2), 3961–3969 (2015)
26. Viswanathan, A., Pires, B.R., Huber, D.: Vision based robot localization by ground to satellite matching in gps-denied situations. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 192–198 (2014)
27. Suenderhauf, N., Shirazi, S., Jacobson, A., Dayoub, F., Pepperell, E., Upcroft, B., Milford, M.: Place recognition with ConvNet landmarks: Viewpoint-robust, Condition-robust, Training-free, Robotics: Science and Systems XI (2015)
28. Vo, N.N., Hays, J.: Localizing and orienting street views using overhead imagery. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, vol. 9905, pp. 494–509 (2016)
29. Hu, S., Feng, M., Nguyen, R.M.H., Lee, G.H.: CVM-Net: Cross-view matching network for image-based ground-to-aerial Geo-localization. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 7258–7267 (2018)
30. Klein, G., Murray, D.: Parallel tracking and mapping for small ar workspaces. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 225–234 (2007)
31. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., Dellaert, F.: Isam2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In: 2011

IEEE International Conference on Robotics and Automation, pp. 3281–3288 (2011)
32. Pascoe, G., Maddern, W., Newman, P.: Direct visual localisation and calibration for road vehicles in changing city environments. In: Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop, ser. ICCVW '15, pp. 98–105 (2015)
33. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. arXiv:1409.4842 (2014)
34. Balntas, V., Li, S., Prisacariu, V.: Relocnet: Continuous metric learning relocalisation using neural nets. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, vol. 11218, pp. 782–799 (2018)
35. Melekhov, I., Ylioinas, J., Kannala, J., Rahtu, E.: Relative camera pose estimation using convolutional neural networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), LNCS, vol. 10617, pp. 675–687 (2017)
36. Sattler, T., Zhou, Q., Pollefeys, M., Leal-Taixe, L.: Understanding the limitations of CNN-based absolute camera pose regression. In: 2019 IEEE Conference on Computer Vision and Pattern Recognition CVPR (2019)
37. Sattler, T., Torii, A., Sivic, J., Pollefeys, M., Taira, H., Okutomi, M., Pajdla, T.: Are large-scale 3D models really necessary for accurate visual localization? In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 1637–1646 (2017)
38. Mohanty, V., Agrawal, S., Datta, S., Ghosh, A., Sharma, V.D., Chakravarty, D.: Deepvo: A deep learning approach for monocular visual odometry. arXiv:1611.06069 (2016)
39. Clark, R., Wang, S., Wen, H., Markham, A., Trigoni, N.: VINet: Visual-inertial odometry as a sequence-to-sequence learning problem. In: AAAI Conference on Artificial Intelligence (2017)
40. Qin, T., Cao, S., Pan, J., Shen, S.: A general optimization-based framework for global pose estimation with multiple sensors. arXiv:1901.03642 (2019)
41. Candela, J.Q., Sugiyama, M., Schwaighofer, A., Lawrence, N.: Covariate shift by kernel mean matching. Dataset Shift in Machine Learning pp. 131–160 (2009)
42. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: Computer Vision and Pattern Recognition (CVPR), pp. 1521–1528 (2011)
43. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. J. Mach. Learn. Res. **17**(1), 2096–2030 (2016)
44. Gong, M., Zhang, K., Liu, T., Tao, D., Glymour, C., Scholkopf, B.: Domain adaptation with conditional transferable components. In: International Conference on Machine Learning, pp. 2839–2848 (2016)
45. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: European Conference on Computer Vision, pp. 213–226 (2010)
46. Sun, B., Saenko, K.: Deep coral: Correlation alignment for deep domain adaptation. In: European Conference on Computer Vision, pp. 443–450 (2016)
47. Tzeng, E., Hoffman, J., Zhang, N., Saenko, K., Darrell, T.: Deep domain confusion: Maximizing for domain invariance. arXiv:1412.3474 (2014)
48. Tzeng, E., Hoffman, J., Saenko, K., Darrell, T.: Adversarial discriminative domain adaptation. arXiv:1702.05464 (2017)
49. Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: CVPR (2017)
50. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
51. Sangkloy, P., Lu, J., Fang, C., Yu, F., Hays, J.: Scribbler: Controlling deep image synthesis with sketch and color. In: CVPR (2017)
52. Karacan, L., Akata, Z., Erdem, A., Erdem, E.: Learning to generate images of outdoor scenes from attributes and semantic layouts. arXiv:1612.00215 (2016)
53. Godard, C., Aodha, O.M., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: CVPR (2017)
54. Johnson, J., Alahi, A., Li, F.-F.: Perceptual losses for real-time style transfer and super-resolution. In: Proc. Euro. Conf. Computer Vision, pp. 694–711 (2016)
55. Kendall, A., Cipolla, R.: Geometric loss functions for camera pose regression with deep learning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, pp. 6555–6564 (2017)
56. Chopra, S., Hadsell, R., Lecun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: Proc. of Computer Vision and Pattern Recognition Conference, CVPR, pp. 539–546. IEEE Press (2005)
57. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of Computer Vision and Pattern Recognition Conference, CVPR, pp. 770–778. IEEE Press (2016). arXiv:1512.03385
58. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: 2010 International Conference on Machine Learning ICML (2010)
59. Strecha, C.: Pix4dmapper: The leading photogrammetry software for professional drone mapping. Website, Accessed. https://www.pix4d.com/product/pix4dmapper-photogrammetry-software (2019)
60. Bentley Systems, Inc.: ContextCapture: Create 3D models from simple photographs and/or point clouds. https://www.bentley.com/en/products/brands/contextcapture
61. Wu, C.: VisualSFM: A visual structure from motion system. http://ccwu.me/vsfm (2011)
62. Blender Foundation: Blender: Open source 3D creation. Free to use for any purpose forever. https://www.blender.org/
63. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
64. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2015)
65. Yang, C., Liu, Y., Zell, A.: RCPNet: Deep-learning based relative camera pose estimation for UAVs. In: 2020 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1085–1092, Athens, Greece (2020)

Chenhao Yang received his M.Sc. degree in optical engineering from the National University of Defense Technology, Changsha, China in the year 2015. He is currently pursuing his Ph.D. in computer science and robotics under the supervision of Prof. Dr. Andreas Zell at the Chair of Cognitive Systems, Faculty of Science, University of Tübingen, Germany. His research is focused on robot vision, machine learning, and autonomous navigation.

Yuyi Liu received bachelor's degrees from Fudan University, China and the University of Birmingham, U.K in 2011, and obtained a master's degree in aeronautical engineering at Imperial College London, U.K. Since 2013, he had a doctorate study on control and navigation of aerial robotics at the University of Tübingen and the Max Planck Institute for Biological Cybernetics, Germany. He received his doctor's degree from the University of Tübingen, Germany in 2018. He is currently a researcher at HRI Laboratory, Graduate School of Informatics, Kyoto University, Japan. His research is focused on navigation and human-robot interaction for aerial and mobile social robots.

Andreas Zell received a University diploma in computer science in 1986 from the University of Kaiserslautern, Germany, an M.S. from Stanford University, USA, in 1987, a Ph.D. from the University of Stuttgart, Germany, in 1989, and the Habilitation (venia legendi) in 1994, all in computer science. Since 1995 he has been a full professor of computer science at the University of Tübingen, Germany, specializing in machine learning, bioinformatics and mobile robotics. In the last five years, his research has focused on deep neural networks, robot vision, navigation and SLAM for wheeled and aerial mobile robots.