Camera pose estimation in multi-view environments: From virtual scenarios to the real world

Jorge L. Charco[a,b,*], Angel D. Sappa[b,c], Boris X. Vintimilla[b], Henry O. Velesaca[b]

[a] *Universidad de Guayaquil, Delta and Kennedy Av., P.B. EC090514, Guayaquil, Ecuador*
[b] *Escuela Superior Politécnica del Litoral, ESPOL. Campus Gustavo Galindo Km. 30.5 Vía Perimetral, P.O. Box 09-01-5863, Guayaquil, Ecuador*
[c] *Computer Vision Center, Edifici O, Campus UAB, 08193 Bellaterra, Barcelona, Spain*

Abstract

This paper presents a domain adaptation strategy to efficiently train network architectures for estimating the relative camera pose in multi-view scenarios. The network architectures are fed by a pair of simultaneously acquired images, hence in order to improve the accuracy of the solutions, and due to the lack of large datasets with pairs of overlapped images, a domain adaptation strategy is proposed. The domain adaptation strategy consists on transferring the knowledge learned from synthetic images to real-world scenarios. For this, the networks are firstly trained using pairs of synthetic images, which are captured at the same time by a pair of cameras in a virtual environment; and then, the learned weights of the networks are transferred to the real-world case, where the networks are retrained with a few real images. Different virtual 3D scenarios are generated to evaluate the relationship between the accuracy on the result and the similarity between virtual and real scenarios—similarity on both geometry of the objects contained in the scene as well as relative pose between camera and objects in the scene. Experimental results and comparisons are provided showing that the accuracy of all the evaluated networks for estimating the camera pose improves when the proposed domain adaptation strategy is used, highlighting the importance

*Corresponding author
Email address: jlcharco@espol.edu.ec (Jorge L. Charco)

on the similarity between virtual-real scenarios.

Keywords: Relative camera pose estimation, domain adaptation, siamese architecture, synthetic data, multi-view environments.

---

1. Introduction

During the last years the number of computer vision applications have grown significantly, mainly due to the appearance of different machine learning techniques and neural network architectures. Some of these applications, including driving assistance, human pose estimation, mobile robots, augmented reality, 3D object reconstruction, just to mention a few, have the automatic camera calibration as their principal process. Mainly to obtain the extrinsic camera parameters, which allows to obtain the relationship between the camera and the world coordinate system. In the current work the multi-view camera pose estimation problem is tackled; in this problem, the relative position and orientation, between two or more cameras, is estimated. This process is a challenging problem since factors such as poor illumination, lack of texture, change in scale, among others, affect it and are ever-present as a part of the real-world landscape. During last decades different algorithms have been proposed for extrinsic camera parameters estimation (i.e., relative translation and rotation) [1, 2, 3, 4, 5, 6, 7]—throughout this work both terms, extrinsic camera parameters and camera pose, will be indistinctly used. Classical approaches are based on the usage of common feature points in a pair of images; these detected-described points (e.g., SURF, ORB, SIFT) are used then to estimate the relative pose between the views. The amount of common feature points, detected in both images, is an important factor to get good accuracy.

Recently, several deep learning based approaches have been proposed for computer vision tasks such as segmentation, image classification, face recognition, super-resolution, among other [8, 9, 10]; in all the cases overcoming the corresponding state-of-the-art results. In deep learning based approaches, a Convolutional Neural Network (CNN) is trained, in general with a large number of data, to solve the specific problem. For the camera pose estimation problem, also different CNN based approaches have been recently proposed, showing appealing results [11, 6, 12]. The main challenge in the camera pose estimation problem, in the multi-view context, lies on the fact that images may have different appearances when there is a large pose difference (position

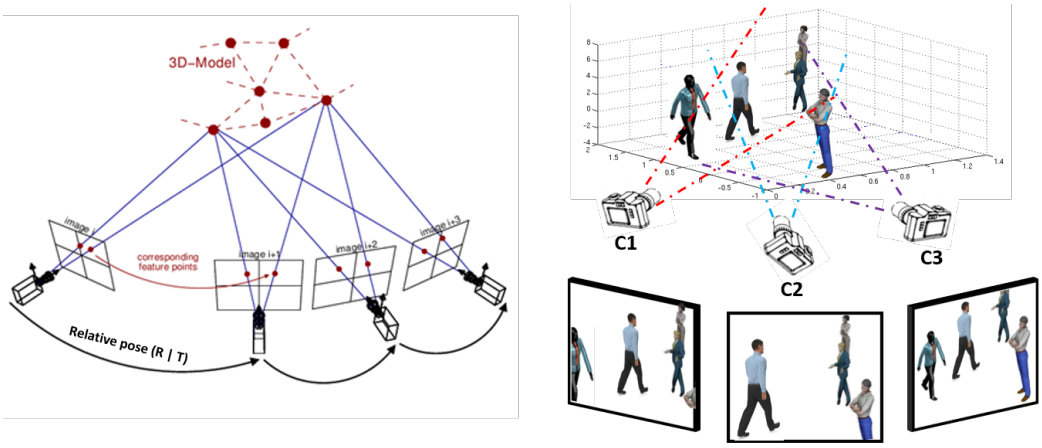

Figure 1: Images captured in a multi-view setup, which are taken at the same time from different positions.

and orientation) between the cameras. This large pose difference is additionally affected by occlusions in the scene, making the camera pose estimation a more challenging problem. Figure 1 shows an illustration of a multi-view scenario containing moving objects with different occlusions.

Having in mind deep learning based approaches' result reaches the best solution, an important element in these schemes is the dataset used for training such architectures. Several datasets have been proposed to tackle this task (e.g., Cambridge [13] and 7-Scene [14]). These datasets contain just a small number of images, which were acquired by a single camera while it moves around the scene. On the contrary to previous datasets, in [15] a novel object oriented dataset has been presented. This dataset, referred to as DTU-Robot, has been acquired using a robotic arm in a scenario that contains a few sets of small static objects (e.g., toys, scale models of buildings, wood blocks, etc.). Although interesting and quite useful dataset, since an accurate ground truth is provided (through the direct/inverse kinematic of the robot's end effector), it has as a limitation the fact that just an indoor and small scenario is considered, being a difficult task transferring knowledge to outdoor large multi-view environments (e.g., video surveillance scenarios).

Despite of different datasets available to tackle the training of deep learning based camera pose estimation approaches, there are some limitations. Main limitation is related with the amount of data in these datasets. The

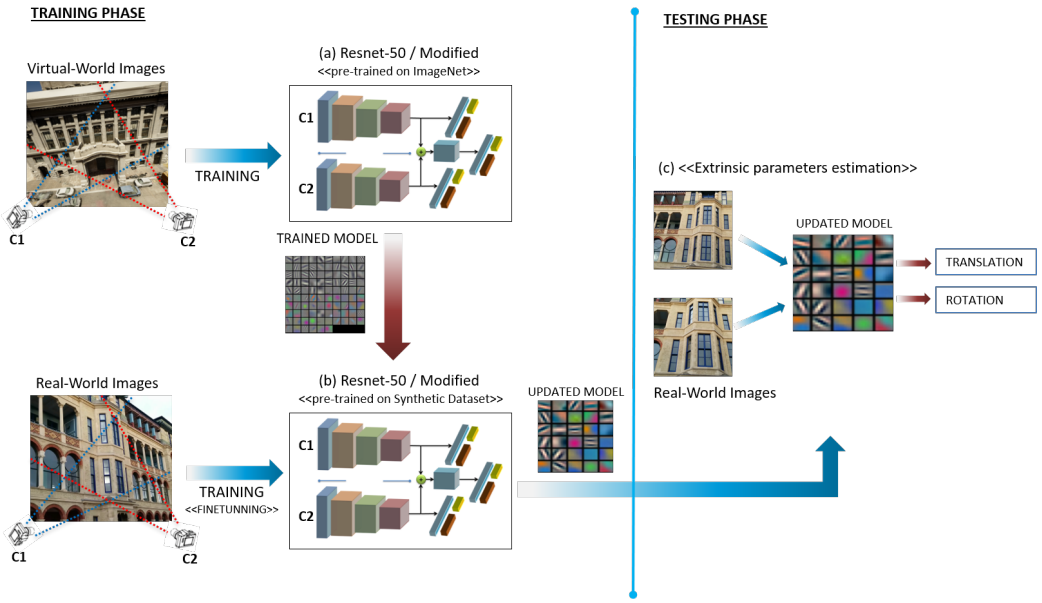

Figure 2: (a) Proposed architecture is trained using synthetic images captured from different positions at the same time. (b) Trained architecture with synthetic images is used to apply DA strategy using real images. (c) Updated weights after DA strategy is used to estimate relative camera pose (i.e., translation and rotation relative).

second problem is related with the type of scenario, in other words, if the scenario used for training is quite different to the one where the approach is used, results may be affected by such a difference (i.e., geometry of the objects contained in the scene as well as their appearance—color and texture). In order to overcome these problems, the usage of virtual environments have been considered in the literature. This strategy has been useful in tasks such as 3D object pose recognition, optical flow estimation, among other [16, 17, 12]. An advantage of using virtual environments lies on the fact that on the one hand, it is possible to generate almost an unlimited set of synthetic images; on the other hand, a large variability, containing different conditions and elements, may be considered; for instance, in the case of pedestrian detection for video surveillance or driving assistance, scenes with different weather conditions, illuminations, pedestrian's shapes, clothes, etc. can be considered during the dataset generation (i.e., synthetic image acquisition). An additional advantage when virtual environments are considered lies on

the fact that ground truth are automatically obtained, reducing human error when the datasets are manually annotated. These virtual environments can be designed by using specific tools such as CARLA Simulator [18], Virtual KITTI [19], including Video Game engines, just to mention a few.

Although the usage of virtual environments could be an option to overcome the limitation of having a reduced amount of data for the training process, the solution is not that much direct. The knowledge acquired in a virtual environment (source) needs to be transferred to the real environment (target) considering the difference between these two domains, for instance in terms of data distribution or feature spaces. This transference is performed by means of Domain Adaptation (DA) techniques. DA techniques have been applied in tasks such as low-resolution image classification [20], attribute-based classification [21], among others, to obtain better results than using just the provided set of real training data.

In the current work, different CNN Siamese architectures are evaluated, which are fed with a set of pairs of images to estimate the relative camera pose. These images are acquired from different points of view at the same time, considering a minimum overlap between them. In order to take advantage of the large amount of data that can be obtained from virtual environments, the architectures are firstly trained using pairs of synthetic images, i.e., pairs of synthetic images obtained from virtual 3D representations of urban environments, which could contain dynamic and static objects such as building, pedestrians, cars and trees; then, the knowledge learned on the trained models is transferred to the real-world using real images, where the camera pose problem is solved. Figure 2 shows an illustration of this DA strategy. In summary, the contributions of this manuscript are as follows:

- Develop a domain adaptation strategy to train a network for tackling the relative camera pose estimation problem.
- Generate different synthetic datasets containing pairs of images from different scenarios.
- Show the importance on the similarity between the real and synthetic images; in other words geometric similarity (i.e., 3D models in real and virtual scenarios) and point of view similarity (i.e., distance between camera and objects in the scene and camera orientation).

The remainder of the paper is organized as follows. In Section 2 previous works are summarized; then, in Section 3 the proposed approach is detailed

together with a description of the used synthetic datasets. Experimental results are reported in Section 4 together with comparisons of results with different synthetic datasets. Finally, conclusions and future work are given in Section 5.

2. Related Works

This section presents a review of works related with the camera pose estimation process together with a review of state-of-the-art approaches on the DA problem.

2.1. Camera Pose Estimation

Automatic estimation of the camera extrinsic parameters, also referred to as camera pose, is a challenging problem. To solve it, some CNN architectures have been recently proposed. In that direction, the authors in [13] have proposed a CNN based method to regress the 6-DOF camera pose from RGB images obtained from a moving camera. This method is robust to changes on illumination, motion blur, and different camera intrinsic parameters. The output is defined by a 3D-vector for the camera's position and a quaternion is used to represent the orientation. The Euclidean distance is used as a loss function and a weighting factor is considered for keeping the position and orientation errors at a similar scale. In [22], the authors update the previous approach with a new loss function, which consists of two components: the residual regression and the uncertainty regularization term; the proposed approach is able to learn camera pose (i.e., position and orientation) optimally, without including any hyperparameters to balance the expected value of position and orientation errors, since both are expressed in different units. In [11] the authors have proposed a novel method for camera pose estimation based on pedestrian's head observations and assuming a known focal length from the input video, which is captured by a static camera. It does not require any special calibration template to infer extrinsic camera parameters from the surveillance video. A synthetic dataset is considered where people's height is predefined. This dataset is used to train the proposed approach, which is then generalized to real-world environments, being robust to false positive detection. Head detector results and known camera focal length are used as inputs of the model to predict extrinsic camera parameters.

A novel architecture based on 3D spatial transformers has been proposed by [23], which tackles the problem of LiDAR-Camera calibration by using

geometric and photometric information. Assumptions about any initial estimate for the extrinsic parameters are avoided, including specific features in the scene. The proposed approach takes as input a LiDAR point cloud, the corresponding monocular image, and the camera calibration matrix K. Instead of directly regress to the calibration parameters, they are obtained maximizing the geometric and photometric consistency of the input images and point clouds. In [7], the authors have proposed an architecture that is able to estimate the relative pose between the two cameras, as well as the pose with respect to the global reference system. This network is based on a Recurrent Convolutional Neural Network (RCNN) where this information (global and relative) is fused, including a Long Short Term Memory in each network to improve the monocular localization accuracy. The proposed approach consists of three parts: a CNN for extracting features of images, a RCNN for relative and global pose regression, and a fully-connected fusion layer to fuse the global and relative pose. Additionally, two loss functions are used: in the first one, cross transformation constraint is employed to enforce the temporal geometric consistency of the consecutive frames. In the second one, mean squared error is used between the predicted pose and ground truth.

On the contrary to previous approaches, where the camera pose estimation problem was related to a single image obtained from a moving camera, the relative camera pose estimation problem is also tackled in multiview scenarios, where more than one camera is used. Note that these two scenarios (single moving camera or multi-camera), although similar, have some little differences. For instance, in the case of a moving camera there are problems when the scene contains moving objects, which is even more difficult in case of the moving objects are articulated or deformable (e.g., a pedestrian, a piece of cloth, etc.). One of the work in this multi-view domain has been presented by [24], where the authors have proposed to use a Siamese CNN architecture to predict the relative camera pose between two cameras. AlexNet architecture is used as a base network for both of the branches and transfer learning is applied to initialize the network with the learned weights from classification task on *ImageNet* [25] and *Places* [26] datasets. Datasets for training neural networks consist of five landmarks (Montreal Notre Dame, Piccadilly, Roman Forum, Vienna Cathedral and Gendarmenmarkt) [27] and DTU Robot Image Dataset [15] for validation. Euclidean distance is used as a loss function to estimate relative camera pose, including a scale factor like in [13] to balance the translation and rotation error. In [6], the authors

have proposed to use a Siamese CNN based on a modified AlexNet architecture with two identical branches and shared weights. The training process was performed from scratch with a set of pairs of images of the same scene simultaneously acquired from different points of view. Unlike the previous approach [24], the obtained features of both pairs of images are concatenated to two fully connected layers to estimate the relative camera pose (i.e., translation and rotation). Euclidean distance is used as a loss function. In [28] the authors have also proposed a Siamese Network architecture to predict relative camera pose, similar to the one proposed by [24, 6]. In this case, the network architecture is based on GoogLeNet where three variants are included in the pose inference module to compute the relative pose between the cameras: a parameter-free module, a parameter-free module with additional losses and a relative pose regressor based on FC-layers. The loss function is the same as the one in [13], where a scale factor (β) is used to balance the loss values between the translation and rotation. Cambridge Landmark dataset [13] is used for training and validation. The output is a 7-Dimensional vector, where rotation is represented as a quaternion, and translation is given by a 3-Dimensional vector (x, y, z).

2.2. Domain Adaptation

During the latest years, machine learning techniques have obtained great success and have been used in several real-world applications. A large-scale dataset is necessary for the training process under the major assumption, which lies in sharing the same feature space or distribution between training and testing data. Some problems between the feature space and distribution in real-world applications could arise due to factors such as illumination change, image quality and scale, poses, just to mention a few. To solve it, a process referred to as Domain Adaptation may be used, where labeled data in one or more relevant source domains are applied on new tasks in a target domain. Some shallow DA methods have been proposed to solve a domain shift problem between two domains. These can be divided into two classes: instance-based DA [29, 30] and feature-based DA [31, 32]. Both classes of DA reduce the discrepancy between domains. The first one, selects certain parts of data from the auxiliary datasets to compose the intermediate domains to train the deep network. These auxiliary datasets could be obtained by combining certain parts of the source and target data. For the second one, shared space is learned to match the distributions of both domains (i.e., source and target).

As mentioned above, deep learning architectures have been used to improve state-of-art results through of feature extraction, which can represent high-level abstractions in different tasks. Some methods based on deep learning, where different DA strategies are embedded into the learning process, have been recently proposed [33]. Depending on relationship between source and target domains, transferring knowledge can be performed in one-step or multi-step DA. In the first case, when both domains (source and target) are directly related, i.e., the feature spaces is the same, it is known as homogeneous DA, which are divided into three types such as supervised DA with labeled data, semi-supervised DA with labeled and unlabeled data and non-supervised DA with unlabeled data; on the contrary, in the second case, when the feature spaces between the source and target domains are nonequivalent, it is known as heterogeneous DA. In the second case, i.e., multi-step DA, an intermediate domain is used, which should be highly related to the source and target domains.

Considering one-step DA, some approaches have been proposed. One of them is the discrepancy-based approach. In [34], the authors experimentally propose to quantify the generality versus specificity of neurons in each layer of a deep convolutional neural network. The performance benefits of transferring features decreases as the distance between the base task and target task increases. They find that using the transfer learning and fine-tuning to a new task can produce a boost to generalization performance. Basically, fine-tuning lies in training a network on a source domain and after reuse the first n layers of the trained network to conduct the training in the target domain. Depending on the amount of data of target domain, and similarity to source domain, these n layers can be fine-tuned or frozen during the training. For the remaining layers, a random initialization is used and trained with loss based on discrepancy, which is also known as class criterion, where a small number of labeled sample is available. It is one of the most basic training loss in deep DA.

In [35] the authors propose a new CNN architecture to exploit unlabeled and sparsely labeled target domain data. Furthermore, simultaneously optimizes for domain invariance to facilitate domain transfer. The domain confusion, which makes that the marginal distributions of the two domains as similar as possible, and softmax cross-entropy losses, are combined to train the target network. It can be used to solve supervised adaptation and unsupervised adaptation. In the first one, a small amount of target labeled data are available, while in the second one, target labeled data are available from

a subset of the categories. A modification of [35] is presented in [36], where the authors propose a new multi-task adaptation approach, which benefits from known cross-category relationships to simultaneously learn and adapt recognition at the attribute and category level.

Additional to the loss functions mentioned above, other methods have been used in supervised DA as training loss to fine-tune the target model. In [37], the authors have proposed a technique that transfers supervision between images from different modalities. The model is able to learn rich representations from unlabeled modalities, being used as a pretraining procedure with limited labeled data. The Euclidean loss is used as a measure of the similarity between the representations. A new deep transfer metric learning method has been proposed by [38], where a set of hierarchical nonlinear transformations is learned by transferring discriminative knowledge from the labeled source domain to the unlabeled target domain. To reduce the distribution difference between both domains (i.e., source and target domain), the marginal Fisher analysis criterion and Maximum Mean Discrepancy are applied. In [21], the authors have tackled the problem of object classification when training and testing classes are disjoint, i.e., the target class information is unavailable. To solve it, they have proposed to introduce attribute-based classification for object detection, considering a human-specified high-level description of the target objects instead of training images, such as: shape, color or even geographic information.

3. Proposed Approach

This section firstly presents the Siamese network architecture, proposed in our previous work [12], for estimating the relative pose (i.e., position and rotation) between the two cameras—this network architecture is fed by a pair of images simultaneously captured by each camera. This architecture is as a benchmark, together with other state-of-the-art approaches, to evaluate the proposed DA strategy. Secondly, the DA strategy proposed to train the networks is introduced. This strategy consists on training the architectures using synthetic datasets and then transferring the learned knowledge to the real world, just by adapting the parameters according to a few images from real scenarios. The synthetic images are acquired by using the CARLA simulator [18]. With this strategy, the need of having a large dataset from the real-world for training the used models is avoided.

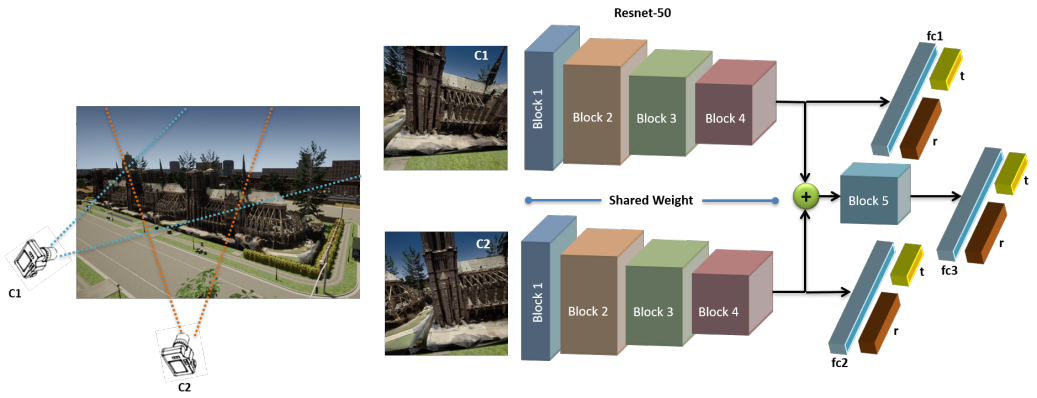

Figure 3: Illustration of the RelPoseTL architecture fed with two images simultaneously captured from different points of views of the same scene.

3.1. Network Architecture

This section presents some details of the network architecture used as a benchmark to evaluate the proposed Domain Adaptation strategy. This network, referred to as RealPoseTL, estimates the relative pose between two cameras by using the acquired images. In the experimental results section other state-of-the-art architectures are also evaluated. RealPoseTL is a Siamese model that consist of two identical branches of a modified Resnet-50 architecture. The weights are shared up to the fourth residual block. Then, the output of each branch is concatenated to feed the fifth block of the used model. Multiple residual units are used, including convolutional layers, batch normalization, pooling, and identity blocks. The activation function of the Resnet-50 architecture is replaced by ELUs function in the used model, since according to [39], ELU helps to speed up convergence, avoiding the vanishing gradient. Three fully connected layers called *fc1*, *fc2* and *fc3* are added on the top of the proposed architecture. The two first fully connected layers (*fc1*, *fc2*) are fed with the output of each branch (i.e., after the fourth residual block). The third fully connected layer (*fc3*) is added after the fifth residual block. The **global pose** can be predicted for both cameras through the features extracted up to the fourth residual block, which are then used to feed the fully connected layers mentioned above. Each fully connected layer, i.e., *fc1* and *fc2*, has a dimension of 1024 followed by two regressors, which help to predict the global translation in 3D vector-format (x,y,z) and rotation in quaternion-format (w,i,j,k). In contrast to the approach mentioned

above, the **relative camera pose** is obtained between both cameras by two regressors, which are fed from the output of the fully connected layer (*fc3*), which has a dimension of 1024 and is connected to the fifth block of the used model. These regressors are used to estimate the relative translation in a 3D vector (*x,y,z*) and rotation in a quaternion (*w,i,j,k*) between the given pair of images.

Relative camera pose is represented by the following equation: $\Delta p = [\hat{t}, \hat{r}]$, where the translation $\hat{t}$ is represented by a 3D vector (*x,y,z*), and the rotation $\hat{r}$ is represented by a 4D vector represented by a quaternion (*w,i,j,k*). As the images are captured at the same time from the same environment, it is reasonable to build one model that is able to predict at the same time the translation and rotation of the pair of images. Normally, the Euclidean distance is used to compute the error between the estimation and ground truth:

$$T_{Global}(I) = \left\| t - \widehat{t} \right\|_{\gamma}, \tag{1}$$

$$R_{Global}(I) = \left\| r - \frac{\widehat{r}}{\|\widehat{r}\|} \right\|_{\gamma}, \tag{2}$$

where the ground truth translation is denoted by t and the predicted translation of used model is represented by $\widehat{t}$. Likewise, r is the ground truth rotation and $\widehat{r}$ denotes the prediction of the quaternion values, which are normalized to a unit length by using $\frac{\widehat{r}}{\|\widehat{r}\|}$. γ is $L2$ Euclidean norm. Both components (i.e., translation and rotation) are trained together within the same loss function, including a factor β to balance both components due to the difference in scale, similar to [13]. Hence, the proposed loss function is defined as:

$$Loss_{global}(I) = T_{Global} + \beta * R_{Global}, \tag{3}$$

since the β parameter depends on several factors (e.g., scene, camera, scale, among others) finding the right value becomes a challenging task. Hence, in order to solve it, two learnable variables called $\widehat{s}_x$ and $\widehat{s}_y$ proposed by [22] are used. These variables act as weights that balance translation and rotation terms, generating a similar effect as β parameter. The modified loss function, which uses the learnable variables, is as follow:

$$Loss_{Global}(I) = (T_{Global} * exp(\widehat{s}_x) + \widehat{s}_x) + (R_{Global} * exp(\widehat{s}_y) + \widehat{s}_y). \quad (4)$$

The loss function mentioned above is useful to independently estimate the global pose for each pair of images using the corresponding branch of the proposed architecture. On the other hand, the output of the fifth block is needed to estimate the relative pose between both cameras, which is obtained as follow:

$$T_{Relative}(I) = \left\| t_{rel} - \widehat{t}_{rel} \right\|_{\gamma}, \quad (5)$$

$$R_{Relative}(I) = \left\| r_{rel} - \frac{\widehat{r}_{rel}}{\|\widehat{r}_{rel}\|} \right\|_{\gamma}, \quad (6)$$

where $T_{Relative}$ and $R_{Relative}$ estimate the relative camera pose between the ground truth and the prediction obtained by the proposed architecture ($\widehat{t}_{rel}$ and $\widehat{r}_{rel}$). Similar to Eq. (2), $\widehat{r}_{rel}$ has to be normalized before. In order to obtain t_{rel} and r_{rel}, the Eq. (7) and Eq. (8) are considered:

$$t_{rel} = t_{C1} - t_{C2}, \quad (7)$$

$$r_{rel} = r^*_{C2} * r_{C1}, \quad (8)$$

where Ci corresponds to the pose parameters of the (i) camera, referred to as a global reference system. In order to obtain the relative translation between both cameras, the Eq. (7) is used, while to determinate the relative rotation between both cameras, the Eq. (8) is considered. In order to compute the relative rotation it is necessary to use the conjugate quaternion of r_{C2} named as r^*_{C2}. Finally, the loss function used to obtain the relative pose is defined as follows:

$$Loss_{Relative}(I) = (T_{Rel} * exp(\widehat{s}_x) + \widehat{s}_x) + (R_{Rel} * exp(\widehat{s}_y) + \widehat{s}_y). \quad (9)$$

Note that the global and relative camera pose parameters are obtained through two different equations. The first one, Eq. (4), predicts the global camera pose using the output from each branch of the trained model, while the second one, Eq. (9), predicts the relative camera pose through of the

Table 1: The initial position and orientation of the two cameras with respect to the global reference system for each virtual scene.

Scene	x	y	z	pitch	yaw	roll
Dataset 1	-52.0	-142.4	10.5	-17.0	90.0	0.0
Dataset 2	-52.0	-148.4	5.5	7.0	90.0	0.0
Dataset 3	-25.0	133.8	3.0	15.0	90.0	0.0
Dataset 4	-29.0	137.8	10.5	-17.0	90.0	0.0
Dataset 5	71.0	238.5	4.0	-17.0	270.0	0.0
Dataset 6	71.0	238.5	11.5	-17.0	270.0	0.0

output of the fifth block of trained model (see Fig. 3). The final loss function for training the proposed architecture is defined as follows:

$$L = Loss_{Global} + Loss_{Relative}. \tag{10}$$

3.2. Synthetic Dataset Generation

This section presents the steps followed to generate the synthetic datasets, which are used for training the different architectures. Synthetic images are obtained from a customized virtual world (i.e., the CARLA Simulator, an open-source software tool [18]). The CARLA simulator has an editor that allows modifying the existing virtual world; this editor integrates both CARLA Simulator and Unreal Engine, which is a video game engine on which this simulator is based. In addition, the open source software tool called OpenMVG [40] was used to guarantee a minimum overlap between a given pair of synthetic images.

The different synthetic image datasets are obtained by the CARLA simulator as indicated below. First, the path to be followed by a mobile vehicle configured for this task is defined. Two virtual cameras are considered; they are placed at two different points on the vehicle. Second, the virtual cameras are configured to generate images with the following attributes: width=2560, height=2560 and FOV=100. Both cameras start the trajectories with predetermined values in their position and orientation (see Table 1); then, these cameras perform the acquisition of images at the same time, which generates a file with the positions (x, y, z) and rotation (pitch, yaw and roll) with respect to the world. The process of acquiring, approximately 3000 synthetic images, from both synchronized cameras, for each dataset, takes approximately three hours. Third, the OpenMVG is used to verify the percentage of

overlap between the pairs of generated synthetic images. The pairs of images that meet the overlap restriction of at least 60% are used to generate a file containing the cameras' relative pose information.

Different virtual scenarios were generated to recreate existing real-world scenarios. The Cambridge Landmarks dataset was used as a reference, which has different real-world scenes [13]. From all the real-world scenarios of Cambridge Landmarks dataset, the KingsCollege and OldHospital scenes have been considered in the current work. In order to generate the virtual scenarios, 3D models of existing buildings, which have similar structures have been used. It is necessary to consider that both scenarios, i.e., virtual and real scenarios, should have the same feature spaces. Additionally, the orientation between the cameras and objects also should be considered.

In order to evaluate the importance on similarity between synthetic and real image datasets for the proposed DA strategy, six datasets have been generated. These datasets are intended to evaluate the importance of geometry similarity between the virtual and real environments and the camera pose similarity.

Figure 4 shows illustrations of the buildings used to generate the virtual scenes. The first two scenarios take as a reference the KingsCollege and OldHospital real-world scenes, while the last scenario is used to demonstrate the importance on the similarity of the geometry of the objects contained in the scene; the trajectories followed by the cameras are also depicted. These trajectories always start from the blue point and move through the red or fuchsia lines. The aforementioned sequence is executed a number of times depending on the virtual scene. At the same time that the configured cameras follow the predefined trajectory, their pose randomly varies at each instant of time (i.e., on the translation the X coordinate could take random values in between [-3, 3]m, the Y coordinate [-1, 3.5]m, while the Z coordinate [-1, 1.5]m; regarding the orientations, the pitch angle could take random values in between [-3.5, 3.5] degrees, yaw angle [-15, 15] degrees while roll angle [-15, 15] degrees). These values were defined according to the characteristics of each scene keeping values that guarantee a large overlap between the views.

4. Experimental Results

As mentioned above, this paper tackles the relative camera pose estimation problem when a reduced amount of data are provided, which are necessary to train the model. This drawback of lack of data is overcome by means of

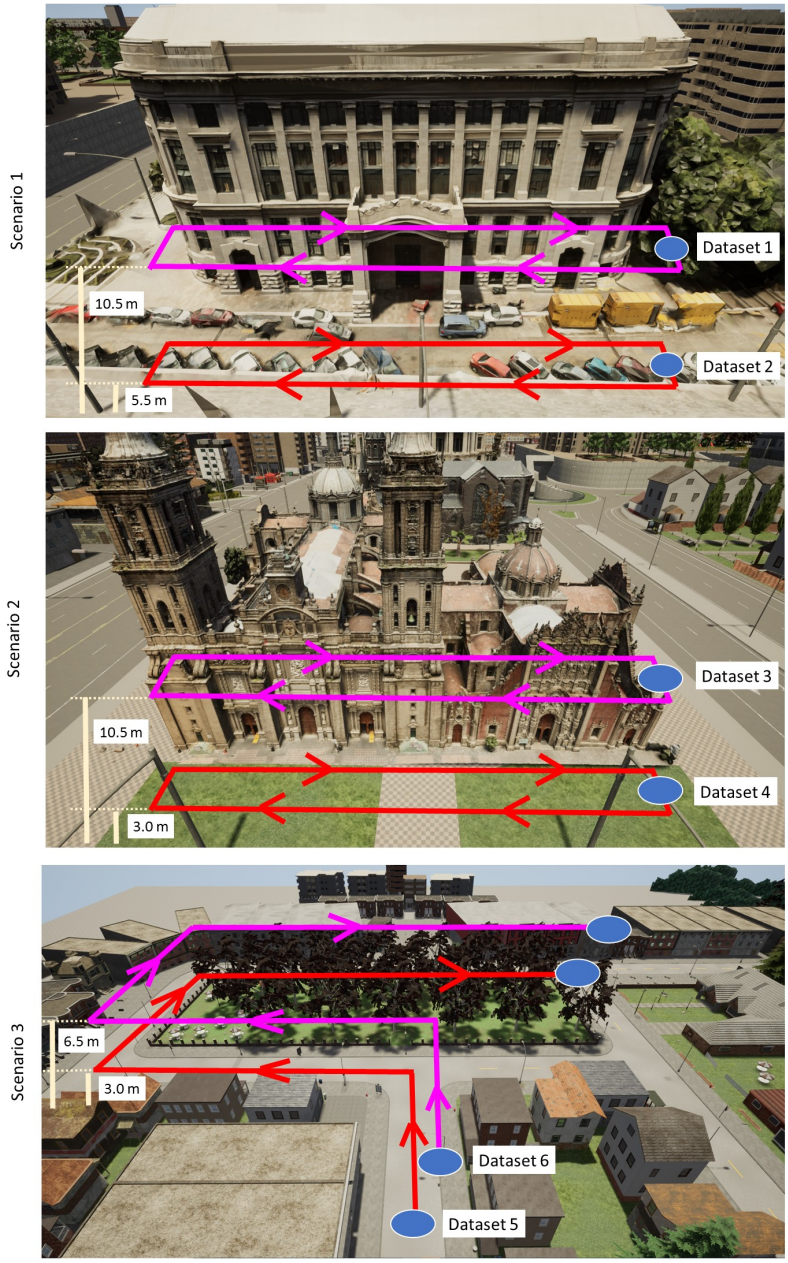

Figure 4: Panoramic views of the virtual scenarios and trajectories followed by the cameras used to create the different datasets. (*1st row*) Scenario 1 used for generating synthetic Datasets 1 and 2. (*2nd row*) Scenario 2 used for generating synthetic Datasets 3 and 4. (*3rd row*) Scenario 3 used for generating synthetic Datasets 5 and 6.

a DA strategy. This section first presents details on the synthetic and real image datasets used for the training and evaluation stages; then, experimental results of the proposed DA strategy are described. This strategy consists of first training with synthetic images, generated by CARLA simulator [18], and then transferring this knowledge to the real-world by adapting the network weights accordingly—the KingsCollege and OldHospital images, from Cambridge dataset [13], are considered. The different architectures evaluated in the current work have been implemented in TensorFlow-Keras and trained with a NVIDIA Titan XP GPU and Intel Core I9 3.3GHz CPU.

4.1. Training with Real Images

This section describes how the evaluated architectures—our RelPoseTL model [12] and state-of-the-art approaches [13, 28]—have been trained using real-world image datasets. For RelPoseTL, all layers were initialized up the fourth residual block with the weights of Resnet-50 pretrained on ImageNet dataset. For the remaining layers, the normal distribution was used. Adam optimizer is used to train the network with a learning rate of 10^{-4} and batch size of 32. The $\widehat{s}_x$ and $\widehat{s}_y$ variables, eq. (9), are initialized with -3.0 and -6.5 in all the experiments respectively.

The architecture in [28], referred to as RPNet^{+}, is a Siamese network proposed to tackle the relative camera pose problem. The layers were initialized with the weights of the GoogLeNet architecture pretrained on Place365 dataset. The stochastic gradient descent with momentum (SGDM) was used as optimizer with a base learning rate of 10^{-5}, decay rate of 0.90 each 80 epochs and batch size of 32.

Finally, regarding PoseNet [13], which is the third architecture evaluated with the proposed DA strategy, it is a slightly modified version of the GoogLeNet architecture. Basically, the softmax layers were removed to output a pose vector of 7-dimensions (position and orientation). SGDM optimizer is used, similarly to our previous work mentioned above [12]. The scale factor β, used in the loss function to keep the position and orientation errors at the same range, has been set to 100 for OldHospital dataset and 500 for KingsCollege of Cambridge dataset.

As a pre-processing stage for all the evaluated architectures, the images were resized to 224 pixels along the shorter side; then, the mean and standard deviation are applied as a normalization process to each image. In the training process, three sets of 256, 512, and 1024 pairs of real-world images were used, which were obtained from KingsCollege and OldHospital of Cambridge

Figure 5: (1st row) Real-world images, OldHospital of Cambridge dataset. (2nd and 3rd row) Synthetic image datasets generated by the CARLA simulator from different points of views (images in Dataset 2 are the most similar to OldHospital dataset).

dataset. For each set, random crops of 224×224 pixels were computed to feed the network architectures, since it allows to better generalize the training. All sets of data were used to train the networks until convergence, which approximately took 60, 90, and 120 minutes respectively for RelPoseTL, and 90, 180 and 360 minutes respectively for RPNet$^+$ and PoseNet.

For the evaluation phase, three sets of 64, 128, and 256 pairs of images were used. The pre-processing mentioned above was also used during the evaluation phase. However, on the contrary to the training phase, central crops were used instead of random crop, since this strategy is generally used in the literature [13, 6, 12].

4.2. Training with Synthetic Images

This section presents details on the training process using synthetic images as a first estimation. All architectures were initialized as presented in Section 4.1. In this case, the network architectures were trained with six different

Figure 6: (1st row) Real-world images, KingsCollege of Cambridge dataset. (2nd and 3rd row) Synthetic image datasets generated by the CARLA simulator from different points of views (images in Dataset 4 are the most similar to KingsCollege dataset).

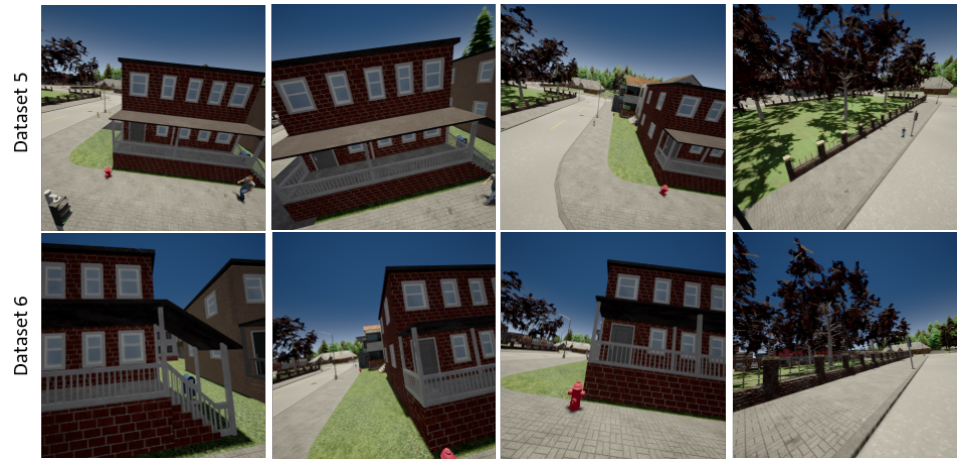

Figure 7: Synthetic image datasets generated by the CARLA simulator from different points of views. These datasets have no similarity to OldHospital or KingsCollege datasets.

synthetic image datasets, which were generated from different virtual environments (see Fig. 4). These synthetic images were captured by different cameras from different orientations at the same time (see Fig. 5, Fig. 6 and Fig. 7). As a pre-processing stage, images were resized to 224×224 pixels, including data normalization, like in the real image case. For the training process, 8192 pairs of images were considered to train the RelPoseTL and RPNet$^+$ architectures; while the PoseNet architecture has been trained with just 900 images. It should be mentioned that PoseNet needs just a set of single images, instead of a set of pair of images, like the previous architectures. Each of the synthetic image datasets contain 900 images. The training process was performed for all synthetic image datasets until convergence, which approximately took 30 hours each one for RelPoseTL and RPNet$^+$, and approximately 20 hours each one for PoseNet. The pre-processing mentioned above was also used during the evaluation phase. In the evaluation, a set of 2048 pairs of images from each synthetic image dataset has been considered for RelPoseTL and RPNet$^+$ architectures, while just 150 synthetic images for the PoseNet architecture.

4.3. Domain Adaptation Strategy

This section presents details on the DA strategy, which transfers the learned knowledge from synthetic images to real-world domain—this strategy is also referred in the literature to as transfer learning. The proposed DA strategy consists on initializing each layer of the architectures evaluated in the current work with the weights learned during the process of training with synthetic images (Section 4.2). Then, the transferring is applied by retraining and refining each network, in an end-to-end way, with real-world images (see Fig. 2); this process is referred to as fine-tunning. The fine-tunning process is independently performed for each one of the synthetic image datasets used to train the architectures mentioned above. In each fine-tunning process three sets of 256, 512, and 1024 pairs of real-world images are considered, these sets of pairs of images belong to OldHospital and KingsCollege of Cambridge dataset (see 1st row of Fig. 5 and Fig. 6). These processes took about 60, 90, and 120 minutes for each set of pairs of real-world images for RelPoseTL architecture, and 90, 180 and 360 minutes respectively for RPNet$^+$ and PoseNet architectures, until convergence was reached. Like in the real-world case (Section 4.1), the evaluation phase has been performed using three sets of 64, 128, and 256 pairs of real-world images.

Table 2: Euclidean distance and angular errors for the DA strategy on pairs of images (PoI) from OldHospital of Cambridge dataset. In each case the network has been initially trained with the six synthetic datasets (RD: Real Data; DTi: Dataset i). The best results for each network are highlighted with boldface and the corresponding training strategy with lightgray color.

		DA strategy on OldHospital dataset		
	Trained with	Train: 256 PoI Test: 64 PoI	Train: 512 PoI Test: 128 PoI	Train: 1024 PoI Test: 256 PoI
RelPoseTL [12]	RD	4.29m, 5.72º	3.93m, 4.04º	3.48m, 3.95º
RelPoseTL [12]	DT1 + RD	4.16m, 6.43º	3.61m, 4.23º	3.66m, 3.99º
	DT2 + RD	**3.55m, 5.59º**	**3.40m, 3.70º**	**3.20m, 3.54º**
	DT3 + RD	3.61m, 6.53º	3.59m, 4.25º	3.31m, 3.72º
	DT4 + RD	3.69m, 5.59º	3.45m, 4.33º	3.29m, 4.31º
	DT5 + RD	4.82m, 6.35º	4.12m, 5.14º	3.94m, 4.65º
	DT6 + RD	4.57m, 7.87º	3.98m, 6.12º	4.11m, 5.89º
RPNet$^+$ [28]	RD	4.06m, 5.70º	3.07m, 5.63º	3.04m, 5.08º
RPNet$^+$ [28]	DT1 + RD	3.19m, 5.74º	3.06m, 5.08º	3.05m, 5.30º
	DT2 + RD	**3.69m, 5.13º**	**3.07m, 5.04º**	**3.03m, 4.57º**
	DT3 + RD	3.21m, 5.55º	3.11m, 5.13º	3.04m, 5.04º
	DT4 + RD	4.01m, 6.08º	3.08m, 5.17º	3.22m, 4.78º
	DT5 + RD	3.23m, 5.78º	3.08m, 5.06º	3.13m, 4.67º
	DT6 + RD	3.57m, 6.05º	3.11m, 5.44º	2.95m, 4.86º
PoseNet [13]	RD	3.28m, 3.56º	3.75m, 5.01º	3.69m, 5.38º
PoseNet [13]	DT1 + RD	2.95m, 3.82º	3.31m, 4.83º	3.68m, 5.29º
	DT2 + RD	**3.19m, 3.39º**	**3.09m, 4.73º**	**3.55m, 5.33º**
	DT3 + RD	2.49m, 3.64º	3.55m, 4.72º	3.71m, 5.23º
	DT4 + RD	3.20m, 3.71º	3.83m, 4.60º	3.81m, 4.62º
	DT5 + RD	3.05m, 3.72º	3.69m, 4.61º	3.58m, 5.33º
	DT6 + RD	3.26m, 3.44º	3.72m, 4.67º	3.87m, 5.43º

Table 3: Euclidean distance and angular errors for the DA strategy on pairs of images (PoI) from KingsCollege of Cambridge dataset. In each case the network has been initially trained with the six synthetic datasets (RD: Real Data; DTi: Dataset i). The best results for each network are highlighted with boldface and the corresponding training strategy with lightgray color.

		DA strategy on KingsCollege dataset		
	Trained with	Train: 256 PoI Test: 64 PoI	Train: 512 PoI Test: 128 PoI	Train: 1024 PoI Test: 256 PoI
RelPoseTL [12]	RD	5.28m, 5.29º	3.86m, 5.08º	2.95m, 4.06º
RelPoseTL [12]	DT1 + RD	4.95m, 8.31º	4.14m, 4.32º	3.38m, 3.75º
	DT2 + RD	4.92m, 5.50º	3.72m, 4.18º	2.86m, 3.68º
	DT3 + RD	4.94m, 5.63º	3.99m, 4.93º	3.04m, 3.82º
	DT4 + RD	**4.89m, 4.96º**	**3.13m, 4.18º**	**2.35m, 3.32º**
	DT5 + RD	5.07m, 8.30º	4.07m, 6.14º	3.85m, 4.94º
	DT6 + RD	4.81m, 7.94º	4.89m, 7.16º	3.94m, 5.99º
RPNet$^+$ [28]	RD	1.90m, 4.35º	1.58m, 3.87º	1.48m, 3.38º
RPNet$^+$ [28]	DT1 + RD	2.39m, 4.16º	1.95m, 3.69º	1.49m, 2.98º
	DT2 + RD	2.30m, 3.93º	1.62m, 3.11º	1.48m, 2.73º
	DT3 + RD	2.33m, 4.19º	1.84m, 3.52º	1.52m, 3.03º
	DT4 + RD	**1.85m, 3.44º**	**1.54m, 3.00º**	**1.27m, 2.69º**
	DT5 + RD	2.13m, 4.09º	1.82m, 3.38º	1.62m, 3.30º
	DT6 + RD	2.35m, 3.74º	1.77m, 3.23º	1.51m, 2.83º
PoseNet [13]	RD	2.14m, 3.92º	2.13m, 3.14º	2.24m, 2.76º
PoseNet [13]	DT1 + RD	2.31m, 4.64º	2.29m, 3.65º	2.36m, 3.03º
	DT2 + RD	1.83m, 3.73º	2.03m, 3.16º	1.99m, 3.17º
	DT3 + RD	2.38m, 3.84º	2.40m, 3.56º	2.24m, 3.34º
	DT4 + RD	**1.95m, 3.55º**	**1.91m, 2.99º**	**1.83m, 2.74º**
	DT5 + RD	2.27m, 4.63º	2.12m, 3.15º	2.32m, 2.85º
	DT6 + RD	2.26m, 3.98º	1.96m, 3.07º	1.95m, 2.98º

4.4. Results

This section presents quantitative evaluations and comparisons when the architectures considered in the current work (RelPoseTL, RPNet$^+$ and PoseNet) are trained with the strategies mentioned above—-i.e., firstly just pairs of real images are considered and secondly, when the proposed DA strategy, which uses synthetic image datasets obtained from the Carla simulator, is considered. Table 2 and Table 3 present the errors obtained for each case. Angular error and Euclidean distance error are used to evaluate the performance of the architectures with/without the usage of the proposed DA strategy. All the datasets presented in Section 3.2 are considered to evaluate the DA strategy. Angular error is used to compute the errors between the estimated rotation and the ground truth, which is represented as a quaternion (i.e., a 4-dimensional vector). On the other hand, Euclidean distance is used to measure the errors between the estimated translation and the ground truth, which is represented as a 3-dimensional vector. The proposed DA strategy is applied over each of the synthetic image datasets in order to evaluate the importance of the similarity between images from a real-world scenario and images from a virtual scenario, both scene geometry similarity as well as camera pose used for the dataset acquisition.

Looking at the results presented in Table 2, it can be appreciated that in the case of OldHospital dataset, the best results are obtained with the DA strategy when synthetic images from Dataset 2 are used to start the training of all architectures mentioned above. With Dataset 2, and by using the proposed DA strategy, in all the cases (256, 512, or 1024 pairs of real images), better results are obtained, both translation and rotation, if they are compared with the architectures trained with just pairs of real images. It should be highlighted that Dataset 2 is the most similar to the OldHospital dataset (see images in Fig. 5 (*1st row and 3rd row*)). Dataset 1 (see images in Fig. 5 (*2nd row*)) has been acquired in the same virtual scenario than Dataset 2, but in this case the cameras were placed further from the building and on a top view. Regarding the KingsCollege dataset, also better results are reached when the proposed DA strategy, using synthetic images, is considered. In this case, the best results correspond to the DA strategy with Dataset 4. Images in Dataset 4 are the most similar to the images in KingsCollege dataset (see images in Fig. 6 (*1st row and 3rd row*)). Dataset 3 (see images in Fig. 6 (*2nd row*)) has been acquired in the same virtual scenario than Dataset 4, but in this case the cameras were placed further from the building and on a top view.

An additional conclusion on the similarity between the real-world image datasets with respect to the synthetic image datasets could be extracted by a comparative analysis between results presented in Table 2 and Table 3 when Dataset 5 and Dataset 6 are considered. These two datasets have been acquired in a synthetic scenario completely different from the OldHospital and KingsCollege of Cambridge dataset (see Fig. 7). None of the networks pre-trained with Dataset 5 or Dataset 6 reach the best result. As mentioned above, in each case the best results have been obtained when the most similar datasets are considered. This similarity should include both the virtual environment (3D scenario used to represent the real environment) as well as the trajectories of the cameras when the synthetic images are acquired, i.e., the distance between the cameras and the objects and the relative camera-object orientation, both features matter.

Finally, regarding the quantitative improvements reached with the proposed DA strategy (i.e., Dataset 2 for the OldHospital dataset and Dataset 4 for the KingsCollege dataset for all network architectures) results are as follow. For OldHospital dataset, the following improvements are reached; first, RelPoseTL arquitecture [12]: *i*) about 17% on translation and 2% on rotation for a set of 256 pairs of real-world images; *ii*) about 13% on translation and 8% on rotation for a set of 512 pairs of real-world images; and *iii*) about 8% on translation and 10% on rotation for a set of 1024 pairs of real-world images. Second, on the RPNet$^+$ architecture [28] the following improvements have been reached: *i*) about 9% on translation and 10% on rotation for a set of 256 pairs of real-world images; *ii*) on translation there is no improvement and about 10% on rotation for a set of 512 pairs of real-world images; and *iii*) about 0.3% on translation and 10% on rotation for a set of 1024 pairs of real-world images. Finally, on PoseNet architecture [13]: *i*) about 3% on translation and 5% on rotation for a set of 256 pairs of real-world images; *ii*) about 18% on translation and 6% on rotation for a set of 512 pairs of real-world images; and *iii*) about 4% on translation and 0.9% on rotation for a set of 1024 pairs of real-world images.

With respect to the second case, the KingsCollege dataset, the following improvements have been reached; first, with the RelPoseTL architecture [12]: *i*) about 7% on translation and 6% on rotation for a set of 256 pairs of real-world images; *ii*) about 19% on translation and 18% on rotation for a set of 512 pairs of real-world images; and *iii*) about 20% on translation and 18% on rotation for a set of 1024 pairs of real-world images. Second, on the RPNet$^+$ architecture [28]: *i*) about 3% on translation and 21% on rotation for a set

of 256 pairs of real-world images; *ii*) about 3% on translation and 22% on rotation for a set of 512 pairs of real-world images; and *iii*) about 14% on translation and 20% on rotation for a set of 1024 pairs of real-world images. Finally, on PoseNet architecture [13] the following improvements have been reached: *i*) about 9% on translation and 9% on rotation for a set of 256 pairs of real-world images; *ii*) about 10% on translation and 5% on rotation for a set of 512 pairs of real-world images; and *iii*) about 18% on translation and 0.7% on rotation for a set of 1024 pairs of real-world images.

The results obtained by the PoseNet architecture after applying the proposed DA strategy, are slightly better compared with the architecture trained with just single images. This little improvement could be explained by the small number of synthetic images used for train the PoseNet architecture in the proposed DA strategy; as mentioned in Section 4.2, this architecture needs single images, instead of pair of images, hence all the images contained in the synthetic image datasets have been considered. This suggests the importance on the number of synthetic images used by the proposed DA strategy. Additionally, the results show in Table 2 and Table 3, indicate that regardless of the architecture, the results are improved when the DA strategy is applied.

5. Conclusions

This paper addresses the challenging problem of relative camera pose estimation by means of a domain adaptation strategy, which avoids the need of having a large dataset of real-world images for training. The proposed DA strategy overcome the dependency mentioned above, when these data are scarce. The manuscript shows how the features extracted on the synthetic images could help to have a better approximation of the weights of the network, and then, to adapt it to the real-world images to reduce the translation and rotation errors. Experimental results and comparisons are provided showing improvements on the obtained results. As a conclusion, it could be stated that the virtual environments' contents should have similar features with respect to the real environments' contents. Furthermore, not only the environments' contents should be similar to the real scenario, but also the image acquisition (distance and point of view between camera and objects).

Acknowledgements

This work has been partially supported by the ESPOL projects EPASI (CIDIS-01-2018) and TICs4CI (FIEC-16-2018); the Spanish Government under Project TIN2017-89723-P; and the "CERCA Programme/Generalitat de Catalunya". The authors acknowledge the support of CYTED Network: "Ibero-American Thematic Network on ICT Applications for Smart Cities" (REF-518RT0559) and the NVIDIA Corporation for the donation of the Titan Xp GPU. The first author has been supported by Ecuador government under a SENESCYT scholarship contract CZ05-000040-2018.

References

[1] R. I. Hartley, Self-calibration from multiple views with a rotating camera, in: Proceedings of the European Conference on Computer Vision, Springer, 1994, pp. 471–478.

[2] A. Sappa, D. Gerónimo, F. Dornaika, A. López, On-board camera extrinsic parameter estimation, Electronics Letters 42 (13) (2006) 745–747.

[3] R. Liu, H. Zhang, M. Liu, X. Xia, T. Hu, Stereo cameras self-calibration based on sift, in: Proceedings of the International Conference on Measuring Technology and Mechatronics Automation, Vol. 1, 2009, pp. 352–355. doi:10.1109/ICMTMA.2009.338.

[4] F. Dornaika, J. M. Álvarez, A. D. Sappa, A. M. López, A new framework for stereo sensor pose through road segmentation and registration, IEEE Transactions on Intelligent Transportation Systems 12 (4) (2011) 954–966.

[5] J. L. Schonberger, J.-M. Frahm, Structure-from-motion revisited, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2016, pp. 4104–4113.

[6] J. L. Charco, B. X. Vintimilla, A. D. Sappa, Deep learning based camera pose estimation in multi-view environment, in: Proceedings of the International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), IEEE, 2018, pp. 224–228.

[7] Y. Lin, Z. Liu, J. Huang, C. Wang, G. Du, J. Bai, S. Lian, Deep global-relative networks for end-to-end 6-dof visual localization and odometry, in: Proceedings of the Pacific Rim International Conference on Artificial Intelligence, Springer, 2019, pp. 454–467.

[8] K. Kamnitsas, C. Ledig, V. F. Newcombe, J. P. Simpson, A. D. Kane, D. K. Menon, D. Rueckert, B. Glocker, Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation, Medical image analysis 36 (2017) 61–78.

[9] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, W. Xu, Cnn-rnn: A unified framework for multi-label image classification, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2016, pp. 2285–2294.

[10] R. E. Rivadeneira, P. L. Suárez, A. D. Sappa, B. X. Vintimilla, Thermal image superresolution through deep convolutional neural network, in: Proceedings of the International Conference on Image Analysis and Recognition, Springer, 2019, pp. 417–426.

[11] E. Shalnov, A. Konushin, Convolutional neural network for camera pose estimation from object detections., International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences 42 (2017).

[12] J. L. Charco., A. D. Sappa., B. X. Vintimilla., H. O. Velesaca., Transfer learning from synthetic data in the camera pose estimation problem, in: Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP,, INSTICC, SciTePress, 2020, pp. 498–505. doi:10.5220/0009167604980505.

[13] A. Kendall, M. Grimes, R. Cipolla, Posenet: A convolutional network for real-time 6-dof camera relocalization, in: Proceedings of the International Conference on Computer Vision, 2015, pp. 2938–2946.

[14] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, A. Fitzgibbon, Scene coordinate regression forests for camera relocalization in rgb-d images, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2013, pp. 2930–2937. doi:10.1109/CVPR.2013.377.

[15] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, A. B. Dahl, Large-scale data for multiple-view stereopsis, International Journal of Computer Vision (2016) 1–16.

[16] M. Jalal, J. Spjut, B. Boudaoud, M. Betke, Sidod: A synthetic image dataset for 3d object pose recognition with distractors, in: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops, 2019.

[17] N. Onkarappa, A. D. Sappa, Synthetic sequences and ground-truth flow field generation for algorithm validation, Multimedia Tools and Applications 74 (9) (2015) 3121–3135.

[18] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator, in: Proceedings of the 1st Annual Conference on Robot Learning, 2017, pp. 1–16.

[19] A. Gaidon, Q. Wang, Y. Cabon, E. Vig, Virtual worlds as proxy for multi-object tracking analysis, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2016, pp. 4340–4349.

[20] X. Peng, J. Hoffman, X. Y. Stella, K. Saenko, Fine-to-coarse knowledge transfer for low-res image classification, in: Proceedings of the International Conference on Image Processing (ICIP), IEEE, 2016, pp. 3683–3687.

[21] C. H. Lampert, H. Nickisch, S. Harmeling, Learning to detect unseen object classes by between-class attribute transfer, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 951–958.

[22] A. Kendall, R. Cipolla, et al., Geometric loss functions for camera pose regression with deep learning, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, Vol. 3, 2017, p. 8.

[23] G. Iyer, R. K. Ram, J. K. Murthy, K. M. Krishna, Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks, in: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1110–1117.

[24] I. Melekhov, J. Ylioinas, J. Kannala, E. Rahtu, Relative camera pose estimation using convolutional neural networks, in: Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems, Springer, 2017, pp. 675–687.

[25] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[26] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, A. Oliva, Learning deep features for scene recognition using places database, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 487–495.

[27] K. Wilson, N. Snavely, Robust global translations with 1dsfm, in: Proceedings of the European Conference on Computer Vision (ECCV), 2014.

[28] S. En, A. Lechervy, F. Jurie, Rpnet: An end-to-end network for relative camera pose estimation, in: Proceedings of the European Conference on Computer Vision, Springer, 2018, pp. 738–745.

[29] L. Bruzzone, M. Marconcini, Domain adaptation problems: A dasvm classification technique and a circular validation strategy, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (5) (2010) 770–787.

[30] W. Chu, F. De la Torre, J. F. Cohn, Selective transfer machine for personalized facial action unit detection, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2013, pp. 3515–3522.

[31] S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Transactions on Neural Networks 22 (2) (2011) 199–210.

[32] Unsupervised domain adaptation via representation learning and adaptive classifier learning, Neurocomputing 165 (2015) 300 – 311. doi:https://doi.org/10.1016/j.neucom.2015.03.020.

[33] M. Wang, W. Deng, Deep visual domain adaptation: A survey, Neurocomputing 312 (2018) 135–153.

[34] J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks?, in: Proceedings of the Advances in Neural Information Processing Systems, 2014, pp. 3320–3328.

[35] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, Simultaneous deep transfer across domains and tasks, in: Proceedings of the International Conference on Computer Vision, 2015, pp. 4068–4076.

[36] T. Gebru, J. Hoffman, L. Fei-Fei, Fine-grained recognition in the wild: A multi-task domain adaptation approach, in: Proceedings of the International Conference on Computer Vision, 2017, pp. 1349–1358.

[37] S. Gupta, J. Hoffman, J. Malik, Cross modal distillation for supervision transfer, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2016, pp. 2827–2836.

[38] J. Hu, J. Lu, Y.-P. Tan, Deep transfer metric learning, in: Proceedings of the Conference on Computer Vision and Pattern Recognition, 2015, pp. 325–333.

[39] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), arXiv preprint arXiv:1511.07289 (2015).

[40] P. Moulon, P. Monasse, R. Marlet, Others, Openmvg. an open multiple view geometry library., `https://github.com/openMVG/openMVG` (2016).