Continual SLAM: Beyond Lifelong Simultaneous Localization and Mapping through Continual Learning

Niclas Vödisch[1], Daniele Cattaneo[1], Wolfram Burgard[2], and Abhinav Valada[1]

Abstract Robots operating in the open world encounter various different environments that can substantially differ from each other. This domain gap also poses a challenge for Simultaneous Localization and Mapping (SLAM) being one of the fundamental tasks for navigation. In particular, learning-based SLAM methods are known to generalize poorly to unseen environments hindering their general adoption. In this work, we introduce the novel task of continual SLAM extending the concept of lifelong SLAM from a single dynamically changing environment to sequential deployments in several drastically differing environments. To address this task, we propose CL-SLAM leveraging a dual-network architecture to both adapt to new environments and retain knowledge with respect to previously visited environments. We compare CL-SLAM to learning-based as well as classical SLAM methods and show the advantages of leveraging online data. We extensively evaluate CL-SLAM on three different datasets and demonstrate that it outperforms several baselines inspired by existing continual learning-based visual odometry methods. We make the code of our work publicly available at http://continual-slam.cs.uni-freiburg.de.

1 Introduction

An essential task for an autonomous robot deployed in the open world without prior knowledge about its environment is to perform Simultaneous Localization and Mapping (SLAM) to facilitate planning and navigation [11, 27]. To address this task, various SLAM algorithms based on different sensors have been proposed, including classical methods [28] and learning-based approaches [3, 18]. Classical methods typically rely on handcrafted low-level features that tend to fail under challenging conditions, e.g., textureless regions. Deep learning-based approaches mitigate such problems due to their ability to learn high-level features. However, they lack the ability to generalize to out-

[1]Department of Computer Science, University of Freiburg, Germany,
[2]Department of Engineering, University of Technology Nuremberg, Germany
This work was funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR.

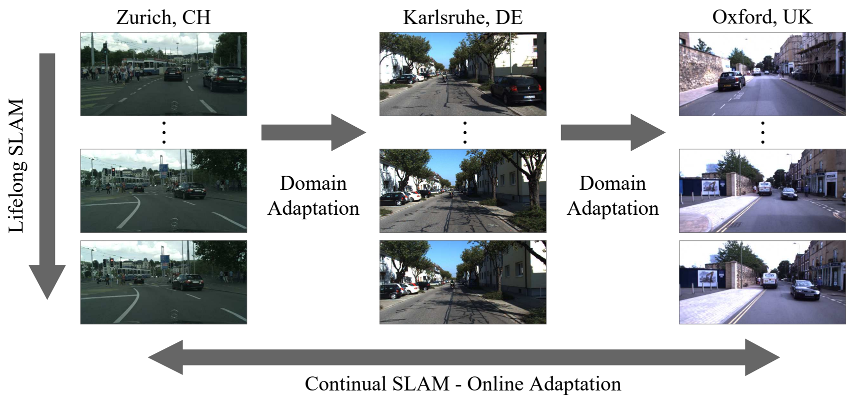

Fig. 1: While lifelong SLAM considers the long-term operation of a robot in a single dynamically changing environment, domain adaptation techniques aim toward transferring knowledge gained in one environment to another environment. The newly defined task of continual SLAM extends both settings by requiring omnidirectional adaptation involving multiple environments. Agents have to both quickly adapt to new environments and effectively recall knowledge from previously visited environments.

of-distribution data, with respect to the training set. For visual SLAM, such out-of-distribution data can correspond to images sourced from cities in different countries or under substantially different conditions. In the following, we use the term *environment* to refer to a bounded geographical area. While different environments can share the same fundamental structure, e.g., urban areas, their specific characteristics prevent the seamless transfer of learned features, resulting in a domain gap between cities [1].

In the context of this work, lifelong SLAM considers the long-term operation of a robot in a single environment (see Fig. 1). Although this environment can be subject to temporal changes, the robot is constrained to stay within the area borders [14], e.g., to obtain continuous map updates [15] within a city. Recent works attempt to relax this assumption by leveraging domain adaptation techniques for deep neural networks, including both regularization [33] and online adaptation of the employed model [19, 20, 23]. While a naive solution for adapting to a new environment is to source additional data, this is not feasible when the goal is to ensure the uninterrupted operation of the robot. Moreover, changes within an environment can be sudden, e.g., rapid weather changes, and data collection and annotation often come at a high cost. Therefore, adaptation methods should be trainable in an unsupervised or self-supervised manner without the need for ground truth data. As illustrated in Fig. 1, the setting addressed in domain adaptation only considers unidirectional knowledge transfer from a single known to a single unknown environment [1] and thus does not represent the open world, where the number of new environments that a robot can encounter is infinite and previously seen environments can be revisited. To address this gap, we take the next step by considering more complex sequences of environments and formulate the novel task of continual SLAM that leverages insights from both continual learning (CL) and lifelong SLAM. We propose a dual-network architecture called CL-SLAM to balance adaptation to new environments and memory retention of preceding environments. To assess its efficacy, we define two metrics, adaptation quality and retention quality, and compare CL-SLAM to several baselines inspired by existing CL-based VO methods involving three different environments. We make the code of this work publicly available at http://continual-slam.cs.uni-freiburg.de. The supplementary material can be found at https://arxiv.org/abs/2203.01578.

2 Related Work

Visual Odometry / SLAM: Visual odometry (VO) and vision-based SLAM estimate camera motion from a video. Allowing for self-supervised training, monocular VO can be tackled jointly with depth estimation based on photometric consistency. SfMLearner [34] uses an end-to-end approach consisting of two networks to predict depth from a single image and camera motion from two consecutive images. The networks are trained in parallel by synthesizing novel views of the target image. Monodepth2 [6] extends the loss function to account for occluded and static pixels. Other works such as DF-VO [32] eliminate the need for a pose network by leveraging feature matching based on optical flow. While these methods show superior performance [20], computing a gradient of the predicted pose with respect to the input image is not possible using classic point matching algorithms. To reduce drift, DeepSLAM [18] combines unsupervised learning-based VO with a pose graph backend taking global loop closures into account. In this work, we use a trainable pose network with velocity supervision [8] to resolve scale ambiguity. Similar to DeepSLAM, we detect loop closures and perform graph optimization.

Continual Learning: Traditionally, a learning-based model is trained for a specific task on a dedicated training set and then evaluated on a hold-out test set sampled from the same distribution. However, in many real-world applications, the data distributions can differ or even change over time. Additionally, the initial task objective might be altered. Continual learning (CL) and lifelong learning [31] address this problem by defining a paradigm where a model is required to continuously readjust to new tasks and/or data distributions without sacrificing the capability to solve previously learned tasks, thus avoiding catastrophic forgetting. Most CL approaches employ one of three strategies. First, experience replay includes rehearsal and generative replay. Rehearsal refers to reusing data samples of previous tasks during adaptation to new tasks, e.g., the replay buffer in CoMoDA [16]. Minimizing the required memory size, the most representative samples should be carefully chosen or replaced by more abstract representations [7]. Similarly, generative replay constructs artificial samples by training generative models [30]. Second, regularization [21] prevents a CL algorithm from overfitting to the new tasks to mitigate forgetting, e.g., knowledge distillation. Third, architectural methods [13] preserve knowledge by adding, duplicating, freezing, or storing parts of the internal model parameters. They further include dual architectures that are inspired by mammalian brains [25], where one model learns the novel task and a second model memorizes previous experience. In this work, we combine architectural and replay strategies by leveraging a dual-network architecture with online adaptation incorporating data rehearsal.

Online Adaptation for Visual Odometry and Depth Estimation: Recently, Luo *et al.* [23] employed a subtask of CL for self-supervised VO and depth estimation, opening a new avenue of research. Online adaptation enables these methods to enhance the trajectory and depth prediction on a test set sourced from a different data distribution than the originally used training set. Both Zhang *et al.* [33] and CoMoDA [16] primarily target the depth estimation task. While Zhang *et al.* propose to learn an adapter to map the distribution of the online data to the one of the training data, CoMoDA updates the internal parameters of the depth and pose networks based on online data and a replay buffer. The work in spirit most similar to ours is done by Li *et al.* [19]. They propose to substitute the standard convolutional layers in the depth and pose networks with convolutional LSTM variants. Then, the model parameters are continuously updated using only the online data. In subsequent work, Li *et al.* [20] replace the learnable pose network by point matching from optical flow. Note that all existing works purely focus on one-step adaptation, i.e., transferring knowledge gained in one environment to a single new environment. In this paper, we introduce continual SLAM to take the next step by considering more complex deployment scenarios comprising more than two environments and further alternating between them.

3 Continual SLAM

Problem Setting: Deploying a SLAM system in the open world substantially differs from an experimental setting, in which parameter initialization and system deployment are often performed in the same environment. To overcome this gap, we propose a new task called *Continual SLAM*, illustrated in Fig. 1, where the robot is deployed on a sequence of diverse scenes from different environments.

Ideally, a method addressing the continual SLAM problem should be able to achieve the following goals: 1) quickly adapt to unseen environments while deployment, 2) leverage knowledge from previously seen environments to speed up the adaptation, and 3) effectively memorize knowledge from previously seen environments to minimize the required adaptation when revisiting them, while mitigating overfitting to any of the environments. Formally, continual SLAM can be defined as a potentially infinite sequence of scenes $\mathcal{S} = (s_1 \rightarrow s_2 \rightarrow \dots)$ from a set of different environments $s_i \in \{E_a, E_b, \dots\}$, where s denotes a scene and E denotes an environment. In particular, $\mathcal{S}$ can contain multiple scenes from the same environment and the scenes in $\mathcal{S}$ can occur in any possible fixed order. A continual SLAM algorithm $\mathcal{A}$ can be defined as

$$\mathcal{A} : \; < \theta_{i-1}, (s_1, \dots, s_i) > \; \mapsto \; < \theta_i >, \tag{1}$$

where $(s_1, \dots, s_i)$ refers to the seen scenes in the specified order and θ_i denotes the corresponding state of the learnable parameters of the algorithm. During deployment, the algorithm $\mathcal{A}$ has to update θ_{i-1} based on the newly encountered scene s_i. For instance, given two environments $E_a = \{s_a^1, s_a^2\}$ and $E_b = \{s_b^1\}$, which comprise two and one scenes, respectively, examples of feasible sequences are

$$\mathcal{S}_1 = (s_a^1 \rightarrow s_b^1 \rightarrow s_a^2), \quad \mathcal{S}_2 = (s_a^2 \rightarrow s_a^1 \rightarrow s_b^1), \quad \mathcal{S}_3 = (s_b^1 \rightarrow s_a^2 \rightarrow s_a^1), \tag{2}$$

where the scene subscripts denote the corresponding environment and the superscripts refer to the scene ID in this environment. As described in Sec. 1, the task of continual SLAM is substantially different from lifelong SLAM or unidirectional domain adaptation as previously addressed by Luo *et al.* [23] and Li *et al.* [19, 20].

To conclude, we identify the following main challenges: 1) large number of different environments, 2) huge number of chained scenes, 3) scenes can occur in any possible order, and 4) environments can contain multiple scenes. Therefore, following the spirit of continual learning (CL), a continual SLAM algorithm has to balance between short-term adaptation to the current scene and long-term knowledge retention. This trade-off is also commonly referred to as avoiding catastrophic forgetting with respect to previous tasks without sacrificing performance on the new task at hand.

Performance Metrics: To address the aforementioned challenges, we propose two novel metrics, namely adaptation quality (AQ), which measures the short-term adaptation capability when being deployed in a new environment, and retention quality (RQ), which captures the long-term memory retention when revisiting a previously encountered environment. In principle, these metrics can be applied to any given base metric M_d that can be mapped to the interval $[0, 1]$, where 0 and 1 are the lowest and highest performances, respectively. The subscript d denotes the given sequence, where the error is computed on the final scene.

Base Metrics: For continual SLAM, we leverage the translation error t_{err} (in %) and the rotation error r_{err} (in °/m), proposed by Geiger *et al.* [5], that evaluate the error as a function of the trajectory length. To obtain scores in the interval $[0, 1]$, we apply the following remapping:

$$\widehat{t_{err}} = \max\left(0, 1 - \frac{t_{err}}{100}\right), \quad \widehat{r_{err}} \quad = 1 - \frac{r_{err}}{180}, \tag{3}$$

where we clamp $\widehat{t_{err}}$ to 0 for $t_{err} > 100\%$. The resulting $\widehat{t_{err}}$ and $\widehat{r_{err}}$ are then used as the base metric M to compute AQ_{trans} / RQ_{trans} and AQ_{rot} / RQ_{rot}, respectively.

Adaptation Quality: The adaptation quality (AQ) measures the ability of a method to effectively adapt to a new environment based on experiences from previously seen environments. It is inspired by the concept of forward transfer (FWT) [22] in traditional CL, which describes how learning a current task influences the performance of a future task. Particularly, positive FWT enables zero-shot learning, i.e., performing well on a future task without explicit training on it. On the other hand, negative FWT refers to sacrificing performance on a future task by learning the current task. In our context, a task refers to performing SLAM in a given environment. Consequently, the AQ is intended to report how well a continual SLAM algorithm is able to minimize negative FWT, e.g., by performing online adaptation.

To illustrate the AQ, we consider the simplified example of a set of two environments $\{E_a, E_b\}$ consisting of different numbers of scenes. We further assume that the algorithm has been initialized in a separate environment E_p. Since the AQ focuses on the cross-environment adaptation, we sample one random scene from each environment $s_a \in E_a$ and $s_b \in E_b$ and hold them fixed. Now, we construct the set of all possible deployment sequences $\mathcal{D} = \{(s_p \rightarrow s_a), (s_p \rightarrow s_b), (s_p \rightarrow s_a \rightarrow s_b), (s_p \rightarrow s_b \rightarrow s_a)\}$, where $s_p \in E_p$ is the data used for initialization. The AQ is then defined as:

$$\mathrm{AQ} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} M_d . \tag{4}$$

Retention Quality: To further account for the opposing challenge of the continual SLAM setting, we propose the retention quality (RQ) metric. It measures the ability of an algorithm to preserve long-term knowledge when being redeployed in a previously encountered environment. It is inspired by the concept of backward transfer (BWT) [22] in CL settings, which describes how learning a current task influences the performance on a previously learned task. While positive BWT refers to improving the performance on prior tasks, negative BWT indicates a decrease in the performance of the preceding task. The extreme case of a large negative BWT is often referred to as catastrophic forgetting. Different from classical BWT, we further allow renewed online adaptation when revisiting a previously seen environment, i.e., performing a previous task, as such a setting is more sensible for a robotic setup. It further avoids the necessity to differentiate between new and already seen environments, which would require the concept of environment classification in the open world.

To illustrate the RQ, we consider a set of two environments $\{E_a, E_b\}$ consisting of different numbers of scenes. We further assume that the algorithm has been initialized on data s_p of a separate environment E_p. We sample two random scenes from each environment, i.e., s_a^1, s_a^2, s_b^1, and s_b^2. To evaluate the RQ, we need to construct a deployment sequence S that consists of alternating scenes from the two considered environments. In this example, we consider the following fixed sequence:

$$S = (s_p \rightarrow s_a^1 \rightarrow s_b^1 \rightarrow s_a^2 \rightarrow s_b^2). \tag{5}$$

We then consider all the subsequences $\mathcal{D}$ of S in which the last scene comes from an environment already visited prior to a deployment in a scene of a different environment. In this example, $\mathcal{D} = \{(s_p \rightarrow s_a^1 \rightarrow s_b^1 \rightarrow s_a^2), (s_p \rightarrow s_a^1 \rightarrow s_b^1 \rightarrow s_a^2 \rightarrow s_b^2)\}$.

The RQ is then defined as the sum over all differences of the base metric in a known environment before and after deployment in a new environment, divided by the size of $\mathcal{D}$. For instance, given the sequence in Eq. 5:

$$\mathrm{RQ} = \frac{1}{2} \left(M_{s_p \rightarrow s_a^1 \rightarrow s_b^1 \rightarrow s_a^2} - M_{s_p \rightarrow s_a^1 \rightarrow s_a^2} + M_{s_p \rightarrow s_a^1 \rightarrow s_b^1 \rightarrow s_a^2 \rightarrow s_b^2} - M_{s_p \rightarrow s_a^1 \rightarrow s_b^1 \rightarrow s_b^2} \right). \tag{6}$$

4 Technical Approach

Framework Overview: The core of CL-SLAM is the dual-network architecture of the visual odometry (VO) model that consists of an *expert* that produces myopic online odometry estimates and a *generalizer* that focuses on the long-term learning across environments (see Fig. 2). We train both networks in a self-supervised manner where the weights of the expert are updated only based on online data, whereas the weights of the generalizer are updated based on a combination of data from both the online stream

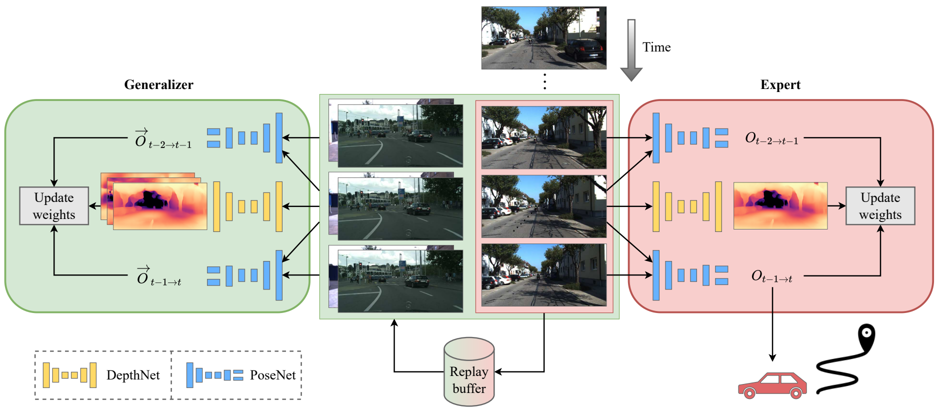

Fig. 2: Online adaptation scheme of our proposed CL-SLAM that is constructed as a dual-network architecture including a generalizer (left) and an expert (right). While the expert focuses on the short-term adaptation to the current scene, the generalizer avoids catastrophic forgetting by employing a replay buffer comprising samples from the past and the present. Note that both subnetworks contain a single PoseNet, shown twice to reflect pose estimation at different steps. The predicted odometry $O_{t-1\to t}$ is sent to the SLAM framework as shown in Fig. 3.

and a replay buffer. We use the VO estimates of the expert to construct a pose graph (see Fig. 3). To reduce drift, we detect global loop closures and add them to the graph, which is then optimized. Finally, we can create a dense 3D map using the depth predicted by the expert and the optimized path.

Visual Odometry: We generate VO estimates following the commonly used approach of using a trainable pose network [2, 6, 8, 18] for self-supervised depth estimation with a stream of monocular images. The basic idea behind this approach is to synthesize a novel view of an input image using image warping as reviewed in the supplementary material.

In this work, we use Monodepth2 [6] to jointly predict the depth map of an image and the camera motion from the previous timestep to the current. To recover metric scaling of both depth and the odometry estimates, we adapt the original loss function with a velocity supervision term as proposed by Guizilini *et al.* [8]. As scalar velocity measurements are commonly available in robotic systems, e.g., by wheel odometry, this does not pose an additional burden. Our total loss is composed of the photometric reprojection loss $\mathcal{L}_{pr}$, the image smoothness loss $\mathcal{L}_{sm}$, and the velocity supervision loss $\mathcal{L}_{vel}$:

$$\mathcal{L} = \mathcal{L}_{pr} + \gamma \mathcal{L}_{sm} + \lambda \mathcal{L}_{vel}. \tag{7}$$

Following the common methodology, we compute the loss based on an image triplet $\{\mathbf{I_{t-2}}, \mathbf{I_{t-1}}, \mathbf{I_t}\}$ using depth and odometry predictions $\mathbf{D_{t-1}}$, $\mathbf{O_{t-2\to t-1}}$, and $\mathbf{O_{t-1\to t}}$. We provide more details on the individual losses in the supplementary material.

Loop Closure Detection and Pose Graph Optimization: In order to reduce drift over time, we include global loop closure detection and pose graph optimization (see Fig. 3). We perform place recognition using a pre-trained and frozen CNN, referred to as LoopNet. In particular, we map every frame to a feature vector using MobileNetV3 small [10], trained on ImageNet, and store them in a dedicated memory. Then, we compute the

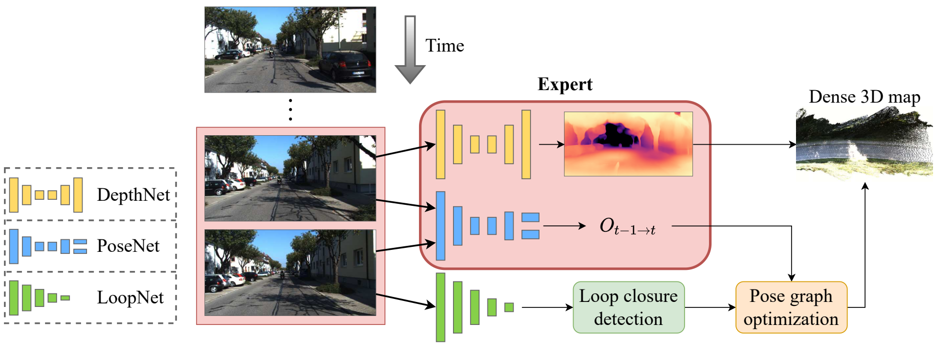

Fig. 3: Full SLAM framework of our proposed CL-SLAM. Global loop closures are detected with a pre-trained CNN. Visual odometry estimates between both consecutive frames and loop closure frames are generated by the PoseNet and added to a pose graph, which is optimized upon the detection of a new loop closure. Finally, a dense 3D map can be created using the predicted depth and the optimized path.

cosine similarity of the current feature map with all preceding feature maps:

$$\mathrm{sim}_{\cos} = \cos(f_{current}, f_{previous}). \tag{8}$$

If $\mathrm{sim}_{\cos}$ is above a given threshold, we use the PoseNet to compute the transformation between the corresponding images. During deployment, we continuously build a pose graph [17] consisting of both local and global connections, i.e., consecutive VO estimates and loop closures. Whenever a new loop closure is detected, the pose graph is optimized.

Online Adaptation: In this section, we describe the dual-network architecture of the VO predictor in CL-SLAM that effectively addresses the trade-off between short-term adaptation and long-term memory retention, a problem also known as catastrophic forgetting. Subsequently, we detail the training scheme including the utilized replay buffer.

Architecture: The dual-network architecture consists of two instances of both the DepthNet and the PoseNet. In the following, we refer to these instances as *expert* and *generalizer*. We build upon the architecture of Monodepth2 [6]. The DepthNet has an encoder-decoder topology, comprising a ResNet-18 [9] encoder and a CNN-based decoder with skip connections, and predicts disparity values for each pixel in the input image. The PoseNet consists of a similar structure using a separate ResNet-18 encoder followed by additional convolutional layers to generate the final output that represents translation and rotation between two input images. Further implementation details are provided in Sec. 5.

Training Scheme: Before deployment, i.e., performing continual adaptation, we pre-train the DepthNet and the PoseNet using the standard self-supervised training procedure based on the loss functions described in Sec. 4. When deployed in a new environment, we continuously update the weights of both the expert and the generalizer in an online manner, following a similar scheme as Kuznietsov *et al.* [16]:

(1) Create an image triplet composed of the latest frame $\mathbf{I_t}$ and the two previous frames $\mathbf{I_{t-1}}$ and $\mathbf{I_{t-2}}$. Similarly, batch the corresponding velocity measurements.
(2) Estimate the camera motion between both pairs of subsequent images, i.e., $\mathbf{O_{t-2\to t-1}}$ and $\mathbf{O_{t-1\to t}}$ with the PoseNet.

(3) Generate the depth estimate $\mathbf{D_{t-1}}$ of the previous image with the DepthNet.
(4) Compute the loss according to Eq. 7 and use backpropagation to update the weights of the DepthNet and PoseNet.
(5) Loop over steps (2) to (4) for c iterations.
(6) Repeat the previous steps for the next image triplet.

Upon deployment, both the expert and the generalizer are initialized with the same set of parameter weights, initially obtained from pre-training and later replaced by the memory of the generalizer. As illustrated in Fig. 2, the weights of the expert are updated according to the aforementioned algorithm. Additionally, every new frame from the online image stream is added to a replay buffer along with the corresponding velocity reading. Using only the online images, the expert will quickly adapt to the current environment. This behavior can be described as a desired form of overfitting for a myopic increase in performance. On the other hand, the generalizer acts as the long-term memory of CL-SLAM circumventing the problem of catastrophic forgetting in continual learning settings. Here, in step (1), we augment the online data by adding image triplets from the replay buffer to rehearse experiences made in the past, as depicted in Fig. 2. After deployment, the weights of the stored parameters used for initialization are replaced by the weights of the generalizer, thus preserving the continuous learning process of CL-SLAM. The weights of the expert are then discarded.

5 Experimental Evaluation

Implementation Details: We adopt the Monodepth2 [6] architecture using separate ResNet-18 [9] encoders for our DepthNet and PoseNet. We implement CL-SLAM in PyTorch [29] and train on a single NVIDIA TITAN X GPU. We pre-train both sub-networks in a self-supervised manner on the Cityscapes dataset [4] for 25 epochs with a batch size of 18. We employ the Adam optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ and an initial learning rate of 10^{-4}, which is reduced to 10^{-5} after 15 epochs. Further, we resize all images during both pre-training and adaptation to 192×640 pixels. Additionally, during the pre-training phase, we mask all potentially dynamic objects using bounding boxes generated by YOLOv5m [12], which was trained on the COCO dataset. We observe that on Cityscapes this procedure yields a smaller validation loss than without masking. We set the minimum predictable depth to 0.1 m without specifying an upper bound. To balance the separate terms in the loss, we set the disparity smoothness weight $\gamma = 0.001$ and the velocity loss weight $\lambda = 0.05$.

During adaptation, we utilize the same hyperparameters as listed above. Inspired by the findings of McCraith *et al.* [26], we freeze the weights of the encoders. Based on the ablation study in Sec. 5.2, we set the number of update cycles $c = 5$. To enhance the unsupervised guidance, we use the velocity readings to skip new incoming images if the driven distance is less than 0.2 m. We construct the training batch for the generalizer by concatenating the online data with a randomly sampled image triplet of each environment except for the current environment as this is already represented by the online data. Finally, we add the online data to the replay buffer.

Table 1: Path accuracy on the KITTI dataset

	Online adaptation to KITTI				Trained on KITTI seq. {0, 1, 2, 8, 9}				No training	
KITTI sequence	CL-SLAM		CL-SLAM (w/o loops)		DeepSLAM [18]		VO+vel [6, 8] (w/o loops)		ORB-SLAM	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
4	–	–	**4.37**	**0.51**	5.22	2.27	10.72	1.69	0.62	0.11
5	4.30	**1.01**	4.41	1.33	**4.04**	1.40	34.55	11.88	2.51	0.25
6	**2.53**	**0.63**	3.07	0.73	5.99	1.54	15.20	5.62	7.80	0.35
7	**2.10**	**0.83**	3.74	1.91	4.88	2.14	12.77	6.80	1.53	0.35
10	–	–	**2.22**	**0.34**	10.77	4.45	55.27	9.50	2.96	0.52

Translation error t_{err} in [%] and rotation error r_{err} in [°/100m]. Sequences 4 and 10 do not contain loops. CL-SLAM is pre-trained on the Cityscapes dataset. The paths computed by ORB-SLAM use median scaling [34] as they are not metric scale. The smallest errors among the learning-based methods are shown in bold.

Datasets: To simulate scenes from a diverse set of environments, we employ our method on three relevant datasets, namely Cityscapes [4], Oxford RobotCar [24], and KITTI [5], posing the additional challenge of adapting to changing camera characteristics.

Cityscapes: The Cityscapes Dataset [4] includes images and vehicle metadata recorded in 50 cities across Germany and bordering regions. Due to the unsupervised training scheme of our VO method, we can leverage the included 30-frame snippets to pre-train our networks despite the lack of ground truth poses.

Oxford RobotCar: The Oxford RobotCar Dataset [24] focuses on repeated data recordings of a consistent route, captured over the period of one year in Oxford, UK. Besides RGB images, it also contains GNSS and IMU data, which we use for velocity supervision. To compute the trajectory error, we leverage the released RTK ground truth positions.

KITTI: The KITTI Dataset [5] provides various sensor recordings taken in Karlsruhe, Germany. We utilize the training data from the odometry benchmark, which includes images and ground truth poses for multiple routes. We further leverage the corresponding IMU data from the released raw dataset to obtain the velocity of the vehicle.

5.1 Evaluation of Pose Accuracy of CL-SLAM

Before analyzing how CL-SLAM addresses the task of continual SLAM, we compare its performance to existing SLAM framework. In particular, in Table 1 we report the translation and rotation errors on sequences 4, 5, 6, 7, and 10 of the KITTI Odometry dataset [5] following Li *et al.* [18]. Since the IMU data of sequence 3 has not been released, we omit this sequence. We compare CL-SLAM to two learning-based and one feature-based approach. DeepSLAM [18] uses a similar unsupervised learning-based approach consisting of VO and graph optimization but does not perform online adaptation. VO+vel refers to Monodepth2 [6] with velocity supervision [8], i.e., it corresponds to the base VO estimator of CL-SLAM without adaptation and loop closure detection. Both learning-based methods produce metric scale paths and are trained on the sequences 0, 1, 2, 8, and 9. Further, we report the results of monocular ORB-SLAM [28] after median scaling [34].

Table 2: Translation and rotation error for computing the AQ and RQ metrics

Used for	Previous scenes	Current scene	$\mathcal{B}_{fixed}$		$\mathcal{B}_{expert}$		$\mathcal{B}_{general}$		CL-SLAM	
			t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
AQ	c_t	k_1	130.74	26.35	**2.50**	**0.37**	$\underline{7.21}$	$\underline{1.26}$	**2.50**	**0.37**
	c_t	r_1	170.76	13.37	**28.94**	$\underline{5.63}$	$\underline{29.05}$	**5.49**	**28.94**	$\underline{5.63}$
	$c_t \to r_1$	k_1	–	–	$\underline{3.66}$	$\underline{0.73}$	14.14	1.79	**3.24**	**0.54**
	$c_t \to k_1$	r_1	–	–	$\underline{32.56}$	$\underline{6.08}$	34.79	6.64	**30.13**	**5.87**
RQ	$c_t \to k_1 \to r_1$	k_2	164.77	25.07	45.20	5.62	$\underline{8.48}$	$\underline{1.79}$	**4.85**	**1.59**
	$c_t \to k_1 \to r_1 \to k_2$	r_2	200.14	28.94	**15.91**	$\underline{4.93}$	$\underline{16.02}$	4.98	20.50	**4.77**
	$c_t \to k_1$	k_2	–	–	15.82	2.50	$\underline{9.37}$	$\underline{2.21}$	**7.48**	**1.63**
	$c_t \to k_1 \to r_1$	r_2	–	–	$\underline{14.89}$	4.62	**12.24**	**4.38**	16.41	$\underline{4.58}$

The *previous scenes* denote the scenes that have been used for previous training of the algorithm, the *current scene* denotes the evaluation scene to compute both errors t_{err} in [%] and r_{err} in [°/100m]. c_t refers to the Cityscapes training set. r_i and k_i are sequences from KITTI and the Oxford RobotCar dataset. Bold and underlined values indicate the best and second best scores on each sequence.

CL-SLAM outperforms DeepSLAM on the majority of sequences highlighting the advantage of online adaptation. Note that CL-SLAM was not trained on KITTI data but was only exposed to Cityscapes before deployment. To show the effect of global loop closure detection, we report the error on sequences 5 to 7 both with and without graph optimization enabled. Note that sequences 4 and 10 do not contain loops. Compared to ORB-SLAM, CL-SLAM suffers from a higher rotation error but can improve the translation error in sequences 6 and 10. The overall results indicate that general SLAM methods would benefit from leveraging online information to enhance performance.

5.2 Evaluation of Continual SLAM

Experimental Setup: In order to quantitatively evaluate the performance of our proposed approach, we compute both the adaptation quality (AQ) and the retention quality (RQ) by deploying CL-SLAM and the baseline methods on a fixed sequence of scenes. In particular, we use the official training split of the Cityscapes dataset to initialize the DepthNet and PoseNet, using the parameters detailed in Sec. 5. The pre-training step is followed by a total of four scenes of the Oxford RobotCar dataset and the KITTI dataset.

$$(c_t \longrightarrow k_1 \longrightarrow r_1 \longrightarrow k_2 \longrightarrow r_2), \qquad (9)$$

where c_t refers to the Cityscapes training set.

Following the setup of Li *et al.* [19], we set k_1 and k_2 to be sequences 9 and 10 of the KITTI Odometry dataset. Note that we omit loop closure detection for this evaluation to prevent graph optimization from masking the effect of the respective adaptation technique. From the Oxford RobotCar dataset, we select the recording of August 12, 2015, at 15:04:18 GMT due to sunny weather and good GNSS signal reception. In detail, we set r_1 to be the scene between frames 750 and 4,750 taking every second frame to increase the driven distance between two consecutive frames. Analogously, we set r_2 to be the scene between frames 22,100 and 26,100. We use a scene length of 2,000 frames in order to be similar to the length of KITTI sequences: 1,584 frames for k_1 and 1,196 for k_2.

Table 3: Comparison of the Adaptation Quality (AQ)

	↑ $\mathrm{AQ}_{\mathrm{trans}}$	↑ $\mathrm{AQ}_{\mathrm{rot}}$
$\mathcal{B}_{fixed}$	0.000	0.890
$\mathcal{B}_{expert}$	<u>0.831</u>	<u>0.982</u>
$\mathcal{B}_{general}$	0.787	0.979
CL-SLAM	**0.848**	**0.983**

$\mathrm{AQ}_{\mathrm{trans}}$ refers to adaptation quality with respect to the translation error, $\mathrm{AQ}_{\mathrm{rot}}$ is based on the rotation error. Bold and underlined values denote the best and second best scores.

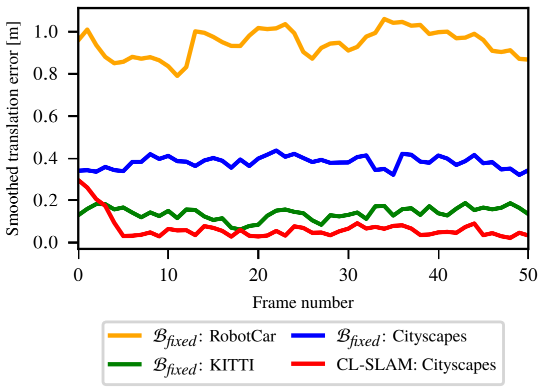

Fig. 4: The translation error on the initial frames of KITTI sequence 4. $\mathcal{B}_{fixed}$ is trained on the different environments indicating the domain gap between them. CL-SLAM overcomes this issue by performing online adaptation.

Baselines: We compare CL-SLAM to three baselines that are inspired by previous works towards online adaptation to a different environment compared to the environment used during training. As noted in Sec. 3, continual SLAM differentiates from such a setting in the sense that it considers a sequence of different environments. First, $\mathcal{B}_{expert}$ imitates the strategy employed by Li *et al.* [19], using a single set of network weights that is continuously updated based on the current data. This corresponds to only using the expert network in our architecture without resetting the weights. Second, $\mathcal{B}_{general}$ follows CoMoDA [16] leveraging a replay buffer built from previously seen environments. This method corresponds to only using the generalizer network. Finally, we compute the error without performing any adaptation, i.e., $\mathcal{B}_{fixed}$ utilizes network weights fixed after the pre-training stage. To further illustrate forward and backward transfer and to close the gap to classical CL, we provide results on an additional baseline $\mathcal{B}_{offline}$ in the supplementary material. This baseline is initialized with the same network weights as CL-SLAM but does not perform online adaptation to avoid masking backward transfer. In reality, it resembles data collection followed by offline training after every new environment.

Adapting to New Environments: In the initial part of the evaluation sequence (Eq. 9), the algorithm has to adapt to unseen environments. In accordance to the definition of the AQ in Sec. 3, we construct four sequences listed in the upper four rows of Table 2. Next, we deploy CL-SLAM and the baselines, initialized with the same set of model weights pre-trained on Cityscapes, on each of these sequences and compute the translation and rotation errors. Note that we do not apply median scaling since the PoseNet in our work produces metric estimates due to the velocity supervision term. Further note that for the first deployment after pre-training, $\mathcal{B}_{expert}$ corresponds to CL-SLAM. We observe that $\mathcal{B}_{expert}$ yields smaller errors than $\mathcal{B}_{general}$. This indicates the importance of online adaptation without diluting the updates with data from unrelated environments, if a high performance on the current deployment is the desideratum, and, thus, supports using the expert network in our approach. To compute the AQ score, after remapping using Eq. 3

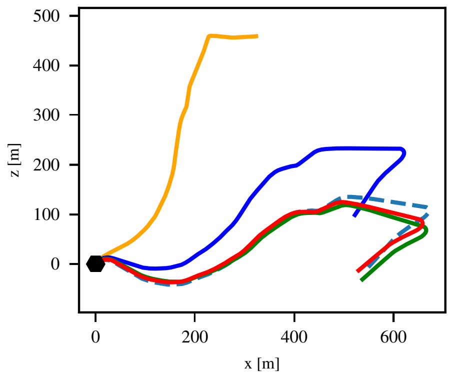

Fig. 5: Comparison of the trajectory on k_2 after previous deployment on k_1 and r_1 predicted by CL-SLAM and the baseline methods. The hexagon indicates the starting point.

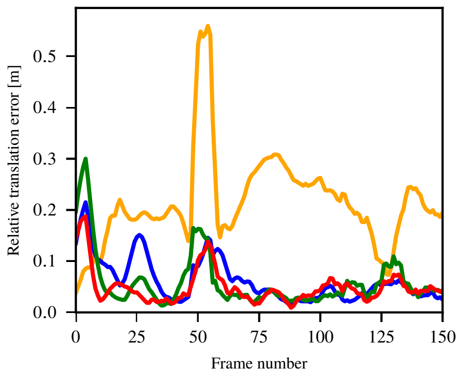

Fig. 6: Relative translation error of the first 150 frames along k_2. Compared to $\mathcal{B}_{expert}$, CL-SLAM reduces the error more quickly due to initialization with the weights of its generalizer network.

we sum the errors and divide by the number of sequences:

$$\text{AQ} = \frac{1}{4}\left(M_{c_t\to k_1} + M_{c_t\to r_1} + M_{c_t\to r_1\to k_1} + M_{c_t\to k_1\to r_1}\right). \tag{10}$$

Comparing the AQ (see Table 3) for all experiments further endorses the previous findings in a single metric. Notably, continual adaptation is strictly necessary to obtain any meaningful trajectory.

Finally, we discuss the effect of consecutive deployments to different environments. In Fig. 4, we plot the translation error of the VO estimates on KITTI sequence 4 without online adaptation, separately trained on the considered datasets, and with adaptation, pre-trained on Cityscapes. As expected, without adaptation, the error is substantially higher if the system was trained on a different dataset showing the domain gap between the environments. By leveraging online adaptation, CL-SLAM reduces the initial error and yields even smaller errors than training on KITTI without further adaptation. Having established the existence of a domain gap, we analyze how the deployment to the current environment effects the future deployment to another environment, resembling the concept of forward transfer (FTW) in continual learning (CL). In detail, Table 2 reveals that the performances of all adaptation-based methods decrease when deploying them to an intermediate environment, e.g., $(c_t \to k_1)$ versus $(c_t \to r_1 \to k_1)$, where the effect is most pronounced for $\mathcal{B}_{general}$. In CL, such behavior is referred to as negative FWT.

Remembering Previous Environments: In the subsequent phase of the evaluation sequence (Eq. 9), the algorithm is redeployed in a new scene taken from a previously encountered environment. In accordance to the definition of the RQ in Sec. 3, we construct four sequences listed in the lower four rows of Table 2. Note that the first two sequences are part of the original evaluation sequence (Eq. 9) and the other two sequences are used as a reference to measure the effect of mixed environments.

Following the same procedure as in the previous section, we compute the translation and rotation errors. The resulting scores (see Table 2) demonstrate the benefit of employing a replay buffer to leverage previously learned knowledge, $\mathcal{B}_{general}$ yields smaller errors than $\mathcal{B}_{expert}$ on the majority of sequences. To compute the RQ, we follow Eq. 6:

$$\mathrm{RQ} = \frac{1}{2}\left[\left(M_{c_t \to k_1 \to r_1 \to k_2} - M_{c_t \to k_1 \to k_2}\right) + \left(M_{c_t \to k_1 \to r_1 \to k_2 \to r_2} - M_{c_t \to k_1 \to r_1 \to r_2}\right)\right]. \tag{11}$$

Comparing the RQ scores in Table 4 clearly shows that the drop in performance when mixing environments is less pronounced for $\mathcal{B}_{general}$. Our proposed CL-SLAM leverages this advantage due to its generalizer, while the expert still focuses on the current scene, achieving the highest RQ across the board.

To bridge the gap to classical CL, we also qualitatively compare the consecutive deployment to scenes from the same environment with introducing an intermediate scene from another environment, e.g., $(c_t \to k_1 \to k_2)$ versus $(c_t \to k_1 \to r_1 \to k_2)$. In CL, an increase/decrease in performance is known as positive/negative backward transfer (BWT). Whereas we observe positive BWT for $\mathcal{B}_{general}$ and CL-SLAM on the KITTI dataset, the sequence with final deployment on RobotCar suffers from negative BWT. A possible explanation for this inconsistent behavior is structural differences between the sequences of the same dataset inducing small domain gaps within a dataset that require a potentially more fine-grained scene classification. However, by always performing online adaptation independent of previous deployments, CL-SLAM circumvents such issues.

In Fig. 5, we visualize the generated trajectories in k_2 given previous deployment in k_1 and r_1 from our method and the evaluated baselines. Although $\mathcal{B}_{expert}$ can reproduce the general shape of the trajectory, it requires a warm-up time causing an initial drift, visible up to frame 40 in Fig. 6. On the other hand, $\mathcal{B}_{general}$ can leverage the experience from k_1 due to the rehearsal of the KITTI data from its replay buffer during the previous deployment in r_1. By following this idea, our proposed CL-SLAM combines the advantages of both baseline strategies.

Table 4: Comparison of the Retention Quality (RQ)

	$\uparrow$ RQ$_{\text{trans}}$ $\times 10^{-3}$	$\uparrow$ RQ$_{\text{rot}}$ $\times 10^{-3}$
$\mathcal{B}_{fixed}$	–	–
$\mathcal{B}_{expert}$	-152.0	-9.5
$\mathcal{B}_{general}$	<u>-14.4</u>	<u>-0.5</u>
CL-SLAM	**-7.3**	**-0.4**

RQ$_{\text{trans}}$ refers to the retention quality with respect to the translation error, RQ$_{\text{rot}}$ is based on the rotation error. $\mathcal{B}_{fixed}$ does not perform adaptation, hence computing the RQ is meaningless. Bold and underlined values denote the best and second best scores.

Table 5: Ablation study on the number of adaptation cycles

		$c_t \to k_1$		$c_t \to k_2$	
Updates	Relative FPS	t_{err}	r_{err}	t_{err}	r_{err}
1	1.00	34.37	6.70	86.36	11.71
2	0.56	31.37	5.83	39.72	7.16
3	0.40	24.21	4.21	**11.15**	4.63
4	0.30	3.24	0.54	13.51	2.03
5	0.24	**2.50**	**0.37**	<u>11.18</u>	<u>1.74</u>
6	0.20	<u>2.84</u>	<u>0.40</u>	12.97	**1.51**

Translation error t_{err} in [%] and rotation error r_{err} in [°/100m] for varying number of weight updates c performed during online adaptation. We use $c = 5$ in CL-SLAM. Bold and underlined values denote the best and second best scores.

Number of Update Cycles: We perform a brief ablation study on the number of update cycles performed during online adaptation, i.e., how often steps (2) to (4) are repeated for a given batch of data (see Sec. 4). For this, we deploy CL-SLAM to both KITTI sequences k_1 and k_2 and compute the translation and rotation error. As shown in Table 5, using five update cycles yields the most accurate trajectory while resulting in a 75% reduction in speed compared to a single cycle. However, please note that in this work, we do not focus on adaptation speed but on showing the efficacy of the proposed dual-network approach to balance the common continual learning trade-off between quick adaptation and memory retention.

6 Conclusion

In this paper, we introduced the task of continual SLAM, which requires the SLAM algorithm to continuously adapt to new environments while retaining the knowledge learned in previously visited environments. To evaluate the capability of a given model to meet these opposing objectives, we defined two new metrics based on the commonly used translation and rotation errors, namely the adaptation quality and the retention quality. As a potential solution, we propose CL-SLAM, a deep learning-based visual SLAM approach that predicts metric scale trajectories from monocular videos and detects global loop closures. To balance short-term adaptation and long-term memory retention, CL-SLAM is designed as a dual-network architecture comprising an expert and a generalizer, which leverages experience replay. Through extensive experimental evaluations, we demonstrated the efficacy of our method compared to baselines using previously proposed continual learning strategies for online adaptation of visual odometry. Future work will focus on transferring the proposed design scheme to more advanced visual odometry methods, e.g., using point matching via optical flow. We further plan to address the currently infinite replay buffer to mitigate the scaling problem, e.g., by storing more abstract representations or keeping only the most representative images.

References

1. Bešić, B., Gosala, N., Cattaneo, D., Valada, A.: Unsupervised domain adaptation for lidar panoptic segmentation. IEEE Robotics and Automation Letters **7**(2), 3404–3411 (2022)
2. Bešić, B., Valada, A.: Dynamic object removal and spatio-temporal rgb-d inpainting via geometry-aware adversarial learning. IEEE Transactions on Intelligent Vehicles (2022)
3. Cattaneo, D., Vaghi, M., Valada, A.: Lcdnet: Deep loop closure detection and point cloud registration for lidar slam. IEEE Transactions on Robotics pp. 1–20 (2022)
4. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The Cityscapes dataset for semantic urban scene understanding. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 3213–3223 (2016)
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: IEEE Conf. on Computer Vision and Pattern Recognition (2012)
6. Godard, C., Aodha, O.M., Firman, M., Brostow, G.: Digging into self-supervised monocular depth estimation. In: Int. Conf. on Computer Vision, pp. 3827–3837 (2019)
7. Gopalakrishnan, S., Singh, P.R., Fayek, H., Ambikapathi, A.: Knowledge capture and replay for continual learning. In: IEEE Winter Conf. on Applications of Computer Vision (2022)

8. Guizilini, V., Ambruș, R., Pillai, S., Raventos, A., Gaidon, A.: 3D packing for self-supervised monocular depth estimation. In: IEEE Conf. on Computer Vision and Pattern Recognition (2020)
9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
10. Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., Le, Q.: Searching for mobilenetv3. In: Int. Conf. on Computer Vision, pp. 1314–1324 (2019)
11. Hurtado, J.V., Londoño, L., Valada, A.: From learning to relearning: A framework for diminishing bias in social robot navigation. Frontiers in Robotics and AI **8**, 69 (2021)
12. Jocher, G.: Yolov5 (2022). URL https://github.com/ultralytics/yolov5
13. Kemker, R., Kanan, C.: Fearnet: Brain-inspired model for incremental learning. In: Int. Conf. on Learning Representations (2018)
14. Kretzschmar, H., Grisetti, G., Stachniss, C.: Lifelong map learning for graph-based slam in static environments. KI - Künstliche Intelligenz **24**(3), 199–206 (2010)
15. Kurz, G., Holoch, M., Biber, P.: Geometry-based graph pruning for lifelong SLAM. In: Int. Conf. on Intelligent Robots and Systems, pp. 3313–3320 (2021)
16. Kuznietsov, Y., Proesmans, M., Van Gool, L.: CoMoDA: Continuous monocular depth adaptation using past experiences. In: IEEE Winter Conf. on Applications of Computer Vision (2021)
17. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: A general framework for graph optimization. In: Int. Conf. on Robotics and Automation, pp. 3607–3613 (2011)
18. Li, R., Wang, S., Gu, D.: DeepSLAM: A robust monocular SLAM system with unsupervised deep learning. IEEE Transactions on Industrial Electronics **68**(4), 3577–3587 (2021)
19. Li, S., Wang, X., Cao, Y., Xue, F., Yan, Z., Zha, H.: Self-supervised deep visual odometry with online adaptation. In: IEEE Conf. on Computer Vision and Pattern Recognition (2020)
20. Li, S., Wu, X., Cao, Y., Zha, H.: Generalizing to the open world: Deep visual odometry with online adaptation. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 13179–13188 (2021)
21. Li, Z., Hoiem, D.: Learning without forgetting. IEEE Trans. on Pattern Analysis and Machine Intelligence **40**(12), 2935–2947 (2018)
22. Lopez-Paz, D., Ranzato, M.A.: Gradient episodic memory for continual learning. In: Adv. in neural information processing systems, vol. 30 (2017)
23. Luo, H., Gao, Y., Wu, Y., Liao, C., Yang, X., Cheng, K.T.: Real-time dense monocular SLAM with online adapted depth prediction network. IEEE Transactions on Multimedia **21**(2), 470–483 (2019)
24. Maddern, W., Pascoe, G., Linegar, C., Newman, P.: 1 year, 1000km: The Oxford RobotCar dataset. Int. Journal of Robotics Research **36**(1), 3–15 (2017)
25. McClelland, J.L., McNaughton, B.L., O'Reilly, R.C.: Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. Psychological review **102**(3) (1995)
26. McCraith, R., Neumann, L., Zisserman, A., Vedaldi, A.: Monocular depth estimation with self-supervised instance adaptation. arXiv preprint arXiv:2004.05821 (2020)
27. Mittal, M., Mohan, R., Burgard, W., Valada, A.: Vision-based autonomous UAV navigation and landing for urban search and rescue. In: Robotics Research, pp. 575–592. Springer (2022)
28. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015)
29. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., et al.: PyTorch: An imperative style, high-performance deep learning library. In: Adv. in neural information processing systems (2019)
30. Shin, H., Lee, J.K., Kim, J., Kim, J.: Continual learning with deep generative replay. In: Adv. in neural information processing systems, vol. 30 (2017)
31. Thrun, S.: Is learning the n-th thing any easier than learning the first? In: Adv. in neural information processing systems, vol. 8 (1995)
32. Zhan, H., Weerasekera, C.S., Bian, J.W., Reid, I.: Visual odometry revisited: What should be learnt? In: Int. Conf. on Robotics and Automation, pp. 4203–4210 (2020)
33. Zhang, Z., Lathuilière, S., Ricci, E., Sebe, N., Yang, J.: Online depth learning against forgetting in monocular videos. In: IEEE Conf. on Computer Vision and Pattern Recognition (2020)
34. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 6612–6619 (2017)