Continual Learning for Image-Based Camera Localization

Shuzhe Wang* Zakaria Laskar* Iaroslav Melekhov Xiaotian Li Juho Kannala
Aalto University
firstname.lastname@aalto.fi

Abstract

For several emerging technologies such as augmented reality, autonomous driving and robotics, visual localization is a critical component. Directly regressing camera pose/3D scene coordinates from the input image using deep neural networks has shown great potential. However, such methods assume a stationary data distribution with all scenes simultaneously available during training. In this paper, we approach the problem of visual localization in a continual learning setup – whereby the model is trained on scenes in an incremental manner. Our results show that similar to the classification domain, non-stationary data induces catastrophic forgetting in deep networks for visual localization. To address this issue, a strong baseline based on storing and replaying images from a fixed buffer is proposed. Furthermore, we propose a new sampling method based on coverage score (Buff-CS) that adapts the existing sampling strategies in the buffering process to the problem of visual localization. Results demonstrate consistent improvements over standard buffering methods on two challenging datasets – 7Scenes, 12Scenes, and also 19Scenes by combining the former scenes[1].

1. Introduction

Camera relocalization is a fundamental problem aimed at estimating 6 degree-of-freedom (DoF) camera pose with respect to a known environment. Visual localization aims to solve this problem requiring only RGB images as input [52–54, 56]. Traditional methods [50–56, 61, 62] require building a 3D map of the environment followed by an explicit matching stage [58, 59] to establish 2D pixels to 3D coordinates. Recently with the success of deep neural networks, the problem can now be solved end-to-end by directly regressing the camera pose [12, 31–33, 42, 47, 57, 63, 67] or 3D scene coordinates [9–11, 37, 68]. This has shown to be more accurate than feature-based methods (at least for small scale environments).

One of the limitations of end-to-end regression methods for visual localization is limited scalability to larger environments with several scenes. Although the methods performed well when trained and evaluated on a single scene, the performance quickly degraded when jointly trained on multiple scenes. This was mitigated by considering a hierarchical approach [37] to localize a given input image – first obtain a coarse localization in terms of the scene or sub-scene, followed by estimating a finer camera pose estimate. In this work, we push the methods further towards a general intelligence setting – learn continually from the incoming stream of data. Under this setting, all the scenes are not available during training but encountered sequentially one after the other as shown in Figure 1. There are several benefits in terms of sample and memory efficiency to learning tasks in a continual manner over the setting of training jointly over all tasks. In the joint training setting, each time a scene has changed, the model needs to be retrained on all the scenes in the database – even the ones that have not undergone any change. Adding new scenes to the database also requires model retraining which affects the scalability. Due to the above issues, the full dataset needs to be stored in memory. In contrast, continual learning (CL) [1, 2, 19, 35] aims to reduce the computational costs by fine-tuning the model only on the changed/new scene and images from the previous scenes stored in a small buffer. Furthermore, the memory costs are also reduced as only the data for the current scene needs to be stored in memory along with a small buffer of images from previous scenes. This is of particular importance for mobile applications where storage capacities are device constrained.

Solely training on the images from the current scene leads to catastrophic forgetting of knowledge gained from prior scenes. This is attributed to the interference of the gradients from the current task images with the model parameters learned on previous scenes. The performance of neural networks in such non-stationary data distribution setting is well studied under the domain of continual learning. The CL problem is broadly categorized into i) *class/task* CL: all the data from current class/task is available and the

*The first two authors contributed equally.
[1]Code and materials are available at https://github.com/AaltoVision/CL_HSCNet.

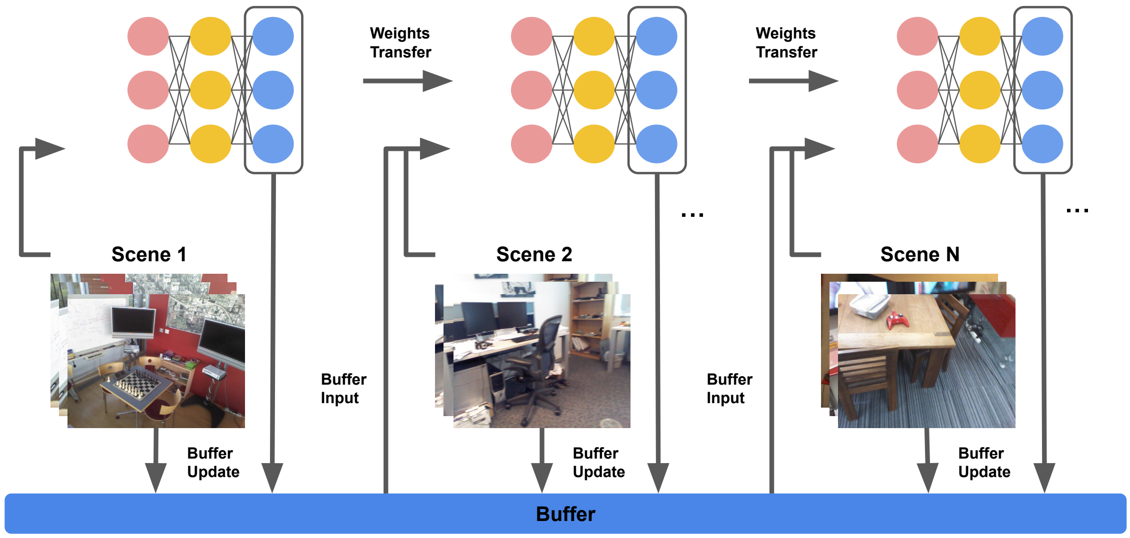

Figure 1. Overview of our replay-based continual learning approach for visual localization. During each scene (task) iteration, the model is updated using the current and previous task samples. The former is sampled from a small fixed-size buffer. After the training is over, a small subset of the current task samples are stored in the buffer by replacing parts of stored data from previous tasks.

model is allowed to have repeated passes through the whole dataset, and ii) *online* CL: the task boundary changes suddenly. To mitigate the challenges of CL, several approaches are proposed: i) *regularization* methods [1, 35, 45] that penalize changes in weights considered important for previous scenes or directly impose orthogonality constraints in the training objective, ii) *modular* methods [2, 28, 49] that increase the model capacity assigning new parameters for each task, and iii) *replay* methods [22, 48] that perform experience replay by storing samples from previous scenes in a fixed size buffer or using generative models to generate images of past scenes. All the three methods incur limited memory and computational costs as regularization methods require past gradients or feature maps to be stored in memory like the replay-based approaches, while modular approaches require an increase in model size. For a fixed model size, experience replay based methods have shown superior performance compared to regularization methods, and in some recent works [19], a combination of both also leads to good results.

In this work, we consider the *task* CL setting for the problem of visual localization in the context of experience-replay based solutions. We adopt some of the buffering methods from literature – *Reservoir* [66], and *Class-balance* [22] to perform experience replay. A strong baseline is created using these methods and challenges specific to the visual localization problem highlighted. Unlike the classification domain where each image is representative of the whole class, visual localization scenes consist of diverse sets of images spanning a whole 3D environment. Existing buffering methods do not take the 3D scene layout into account. Storing images from just one part of the scene does not guarantee generalization to images from other disjoint parts. To retain performance on several parts of the scene, we propose a buffering process that ensures the images stored in the buffer will have higher scene coverage. This is done by computing a coverage score factor that indicates if buffering new incoming image will improve the existing coverage score of buffer images. The proposed buffering algorithm outperforms existing methods on challenging datasets – 7Scenes, 12Scenes, and also 19Scenes obtained by combining the former scenes.

To summarize, we make the following contributions:

- Introduce the problem of continual learning for visual localization.
- Create a strong experience-replay baseline from existing buffering methods across several indoor datasets.
- Propose a new buffering strategy conditioned on the 3D geometry of the scene.

2. Related Work

Visual localization. Visual localization is the task of estimating 6-DoF camera pose from an image. Conventional methods [50–56, 61, 62] solve it by matching image features against a prebuilt 3D map [58, 59]. The camera pose can then be recovered from the 2D-3D matches in a RANSAC [25] optimization loop. Recently with the success of deep neural networks, learning-based methods have been proposed to tackle the problem. To learn the entire localization pipeline end-to-end, PoseNet [33] was first proposed to directly regress the absolute camera pose from an RGB image, and was later improved upon and studied in [12,31,32,42,47,57,63,67]. Instead of directly regressing the absolute pose, in [4,23,36,70], neural networks are trained to predict query pose relative to the database images of which the poses are known. Scene coordinate regression methods [6–11,13,14,16–18,26,37–39,41,41,43,44,60,65,68,69], unlike pose regression methods, focus on predicting 2D-3D correspondences directly from the image, and the camera pose can be solved using the predicted correspondences as in the conventional pipeline.

Continual learning. Regularization methods penalize changes in weight parameters important for previous tasks [1,35,45]. Modular approaches [2,24,28,49] assign new task-specific parameters for each new task amounting to zero forgetting. However, this comes at additional memory requirements. Meta-learning based approaches [5,30,46] use meta-learning to learn sequential tasks. Replay based methods [22,29,48] use knowledge distillation [27] to rehearse using a small episodic memory of data stored from previous tasks. On the other hand, several works [3,21,40] use the episodic memory as an optimization constraint that penalizes increase in loss at previous tasks.

Reservoir sampling [66] samples a subset of data. Aljundi *et al.* [3] proposes two sampling methods that attempt to maximize the gradient directions of the stored samples in buffer memory. Recently proposed sampling method by Chrysakis *et al.* [22] propose a balanced buffering strategy to deal with imbalanced class distribution.

3. Methods

In this section, we provide a brief background of continual learning, and the visual localization pipeline used in the continual learning setting. This is followed by the proposed buffering strategy to adapt existing continual learning solutions for visual localization problem.

3.1. Continual Learning

We consider a parameterized mapping such as deep-neural network $f \;\; o \;\; g : x \rightarrow \hat{y}$, where $x \in \mathbb{R}^{W \times H \times 3}$ is the input image, $g_\Theta : x \rightarrow \tilde{y}$ encodes the intermediate representation and $f_\theta : \tilde{y} \rightarrow \hat{y}$ maps the intermediate representation to the final output space.

Given a stream of non-stationary iid data, D_t, $t = 1...T$, continual learning aims to learn the parameters of the model $f \;\; o \;\; g$ using the following loss function:

$$L = \underset{\theta,\Theta}{\operatorname{argmin}} \sum_{t=1}^{T} L_t \tag{1}$$

where $L_t = \mathbb{E}_{(x,y)\sim D_t} e(\hat{y}, y)$, $e(.)$ refers to the error function such as Euclidean Loss or Cross-Entropy loss between $\hat{y}$ and corresponding ground-truth labels, y.

Buffering. To prevent catastrophic forgetting, a small amount of previous data is stored in a buffer of fixed size, B. Input images from the current task/class and corresponding labels are stored in the buffer. We refer to this process of storing images in the buffer as *Img-buff*.

Apart from images, intermediate representations are also stored that provide a better manifold structure. For example, storing pre-softmax layer logits provides a distribution of class probabilities that encodes inter-class semantic relationships. The buffer B_z stores these representations, $\tilde{y}$ w.r.t each image in the buffer, B. Buffering intermediate representations is referred to as *Rep-buff*.

Replay. Replay is the process of re-iterating through samples from past scenes stored in the buffer while learning the current task. The final loss is computed for both the current task samples and those from buffer, B as:

$$L = L_t + \mathbb{E}_{(x,y)\sim B} e(\hat{y}, y) \tag{2}$$

The intermediate representations stored in B_z can be used as pseudo-labels through the process of knowledge distillation. For example, logits from the current network state are constrained to be similar to corresponding ones stored in buffer memory, B_z.

$$L = L_t + \mathbb{E}_{(x,y,\tilde{y})\sim B_z}(e(\hat{y}, y) + e(\hat{\tilde{y}}, \tilde{y})) \tag{3}$$

Algorithms. The buffering algorithm decides which samples in the current task are to be stored for future replay and which samples stored in the buffer are to be replaced. The first stage consists of filling the buffer until it is full. The second stage then decides buffering probability of additional incoming instances. Here we discuss two baseline approaches:

Reservoir sampling assigns the buffering probability of a new instance as $|B|/N$, where N is the total number of instances observed.

Class-balance: One of the limitations with reservoir sampling is class-imbalanced problem – cardinality difference between different classes in the buffer. As a result, there is also an imbalance in replay rate of different class instances. To create a balanced buffer, only the instances from the largest class are replaced once the buffer is full. If the class c corresponding to current sample (x) itself is the largest class, then one of its sample in the buffer is replaced with probability m_c/n_c where m_c is the number of currently stored instances of c, while n_c is the total instances observed of class c.

The pseudo codes for *Reservoir* and *Class-balance* sampling are presented in the supplementary material.

3.2. Visual Localization

Visual localization aims to estimate the 6DoF camera pose from a given input image. While many deep learning methods have been proposed, we focus on a particular class of methods: deep structured models that have shown to be more accurate than feature based methods [8,9,37]. Among these methods, only a recently proposed *HSCNet* [37] has shown scalability to large number of disjoint scenes with a single deep model. Unlike related works, *HSCNet* [37] maintains an implicit representation of the scene in a set of parameterized hierarchical network layers that predict 3D scene coordinates for each 2D pixel location. Using PnP, the 2D-3D correspondences are used to obtain the final query camera estimate.

The ground-truth 3D points, y_{3D} are hierarchically clustered into coarse-to-fine set of discrete labels, y_l, where $l = 1, 2 ... L$ are the coarse-to-fine cluster levels such that $|y_1| < |y_2| ... < |y_L| < |y_{3D}|$. The combined label set is defined as $\mathbf{y} = y_{3D} \bigcup \{y_l\}_{l=1}^{L}$ with $\mathbf{y}(x)$ corresponding to the labels for input x. We refer the reader to the supplementary for the detailed explanation. For each input pixel in a given image, *HSCNet* [37] predicts the corresponding cluster label in coarse-to-fine manner, where finer predictions are conditioned on coarser predictions from previous layers using conditioning layers. The task loss for visual localization of input image x is the sum of losses at each cluster level summed over all pixels:

$$L_t = \sum_{l=1}^{L} \alpha_l \cdot e(\hat{y}_l, y_l(x)) + \beta \cdot e(\hat{y}_{3D}, y_{3D}(x)) \quad (4)$$

where α_l and β are the weighting coefficients.

In a continual learning setup, the scenes are presented sequentially, and the continual loss function in Eq. 1 is computed using Eq. 4. For *Img-buff* only the input images and corresponding 3D scene coordinates, $\mathbf{y}$ are stored in B. In addition, the intermediate cluster-level predictions, $\tilde{\mathbf{y}} = \hat{y}_{3D} \bigcup \{\hat{y}_l\}_{l=1}^{L}$ are also stored for *Rep-buff* (*c.f.* Sec. 4).

3.3. Coverage Score Buffering

Unlike classification problems, visual localization scenes/classes are visually diverse and independent - learning localization on images of a particular sub-scene does not enable generalization to other parts of the scene. To retain localization performance on all sub-scenes of a given scene, the buffer needs to maintain images that maximize the scene coverage. In this section we propose a method to

Algorithm 1 Buffering

input stream: $(x, \mathbf{y} \sim D_t)$
$c \equiv$ class(x)
Buffer Memory: B
for $i = 1$ to n **do**
  if B is not **filled then**
    $B \leftarrow (x, y)$
  else
    Reservoir / Class-balance / Buff-CS
  end if
end for

Algorithm 2 Buff-CS

if c is not largest **then**
  Select a random instance of largest class
  Replace it with $(x, \mathbf{y})$
else
  Flag $\leftarrow \emptyset$
  cs $\leftarrow$ $CoverageScore$(c, $(x, \mathbf{y})$)
  if cs is not $\emptyset$ **then**
    Flag $\leftarrow 1$
  else
    $m_c \leftarrow$ number of currently stored instances of c
    $n_c \leftarrow$ number of total instances observed of c
    u $\sim$ Random(0,1)
    if $u < m_c/n_c$ **then**
      Flag $\leftarrow 1$
    end if
  end if
  if Flag is not $\emptyset$ **then**
    Replace an instance of c with $(x, \mathbf{y})$
  else
    Ignore
  end if
end if

Algorithm 3 $CoverageScore$

input: class c
input: new instance $(x, \mathbf{y})$
Sample data $\{x_b, \mathbf{y}_b\}_{b=1}^{|B_c|}$ of class c from B
Compute cs_1 using Eq. 6
return cs_1

sample images that provides an improved scene coverage - referred to as *Buff-CS*.

The fundamental difference with *Class-balance* is in the case where the current scene is the largest class in the buffer, and each new instance is buffered with probability m_c/n_c. With increasing n_c, or for small m_c the buffering probability decreases sharply. Thereby, later observations have a lower probability of being buffered[2]. In this work, we increase the buffering probability to 1 if the incoming new instance provides novel scene observations compared to the instances that observed by the buffer images. Example scene observations can be the ground-truth 3D scene coordinates. Given the 3D scene coordinates $y_{3D}(x)$ seen by the new instance x and those by buffer images of class $c, y_{3D}(B_c) = \{y_{3D}(x_b)\}_{b=1}^{|B_c|}$ we compute the coverage score factor:

$$cs_{3D} = y_{3D}(x) \setminus y_{3D}(B_c) \tag{5}$$

If cs_{3D} is not $\emptyset$, x observes novel 3D points that are not seen by the images $I \in B_c$ and hence will be registered into the buffer, B.

As the coverage score is computed for each sample during the buffering process, an efficient method is required to compute the coverage score. Although y_{3D} provides an accurate estimate of coverage score, for dense 3D models this becomes computationally intensive. We propose an efficient method to compute the coverage score by replacing y_{3D} in Eq. 5 with the coarsest cluster level y_1:

$$cs_1 = y_1(x) \setminus y_1(B_c) \tag{6}$$

As $|y_1| \ll |y_{3D}|$ the computational efficiency is significantly improved at the cost of lower coverage score accuracy. The gain in coverage score obtained using coarse-level cluster information (c.f. Table 1) indicates that the approximate method still achieves higher coverage score than *Class-balance* method. Algorithms 2 and 3 are the pseudo codes for *Buff-CS* and our coverage score method.

4. Experiments

In this section, we describe our experimental setup.

4.1. Benchmarks and Environment Setup

Datasets. Different from standard *continual learning* tasks evaluated on classification benchmarks, we select two *visual localization* benchmarks, **7Scenes** [60] and **12Scenes** [64] for the experiments. **7Scenes** records RGB-D image sequences of seven different indoor scenes from a handheld Kinect camera, each sequence consists of 500–1000 frames at 640×480 resolution. **12Scenes** is another indoor RGB-D dataset containing 4 large scenes, with a total of 12 rooms, captured using a Structure.io depth sensor coupled with an iPad color camera. Both datasets also provide dense 3D points and the ground truth camera poses. In order to evaluate the **CL** methods in a sequential manner, we integrate the individual seven scenes and twelve scenes into single coordinate systems and yields to two large scenes similar to [9, 37]. In addition, we also synthesize the largest scene by the combination of all nineteen scenes. These three large scenes are denoted by **i7S** (ca. $125m^3$ total), **i12S** (ca. $520m^3$ total), and **i19S** (ca. $645m^3$ total), respectively.

Baselines. In this work, we adopt two buffering methods as the baselines, namely *Reservoir* [66] and *Class-balance* [22]. These two methods are referred to as *Reservoir* and *Class-balance* respectively. *Reservoir* aims to sample k data instances from an input stream of unknown size, where k is the predefined sample size. The data from the input stream in our experiments are the training frames of **i7S**, **i12S**, or **i19S**, and this method guarantees the same probability for the individual frame to be selected into the buffer. *Class-balance* aims to further solve the class-imbalance problem in online continual learning. This method keeps the classes as balanced as possible, while the distribution of each class/scene is preserved.

Besides the aforementioned methods, we also consider one weak baseline – train our models without buffering and image-replay. We refer to this method as *W/O Buffering*.

Evaluation Metrics. Following [60], we evaluate the performance of these methods using pose accuracy. Pose accuracy is defined as the percentage of the query images with an error below 5 cm and 5°. We consider both the accuracy after the training is complete and the average accuracy over different stages of the training process. Similar to [20], the latter one is defined as:

$$A_i = \frac{1}{N - i + 1} \sum_{j=i}^{N} a_{i,j} \tag{7}$$

where N is the total number of scenes, and $a_{i,j}$ denotes the accuracy of the model on scene i after the training of the model on scene j is complete.

4.2. Implementation Details

In the task of continual learning for visual localization, individual scenes are fed to the training network in an incremental manner – that is to say, data in the first scene is trained to estimate scene coordinates, then the training weights are utilized as the initialization for the second scene.

For training HSCNet [37] in a continual learning setup, the training data of each scene are sampled and stored in the buffer after the training of the corresponding scene is complete. As mentioned before, buffering only the input images and corresponding labels is referred to as *Img-buff*, and buffering additionally the intermediate representations is referred to as *Rep-buff*. For *Img-buff*, we store the RGB images, the depth maps, and ground truth poses to the buffer, and the ground truth labels for training are generated in the same way as in [37]. For *Rep-buff*, we additionally store pre-softmax layer logits and the scene coordinates predicted

[2]Note that when current scene is not largest, the sample will have buffering probability 1

by the current model.

We keep most of the training settings in [37] unchanged. However, some changes are made to adapt it to our continual learning setup. First, all the networks are trained using the Adam [34] optimizer with a smaller learning rate of $5e^{-5}$. Second, different from Li *et al*. [37] who train each individual scene with 300K iterations, we reduce it to 30K for each scene to save the training time. This is because of the sequential property of the continual learning task, *i.e*. the training on each scene begins only after the previous one is finished. It is reported in [37] that the training time for each scene is around 12 hours, and thus the training would become impractical if we keep the number of iterations unchanged for our experiments. Besides, we also found that training with 30K iterations still leads to comparable results.

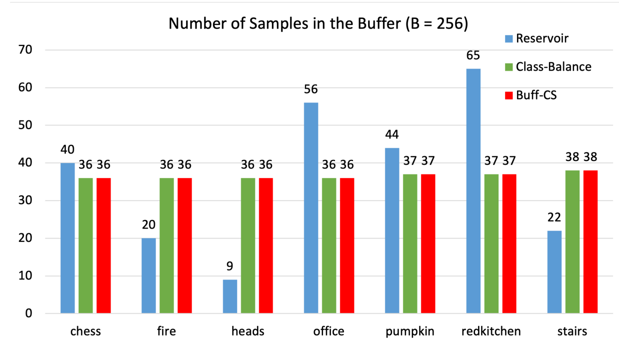

Figure 2. The memory distribution of three methods with buffer size = 256 after the training is complete on **i7S**. *Reservoir* suffers from data imbalance while the samples in *Class-balance* and *Buff-CS* are balanced.

5. Results

5.1. Comparison with Baselines

In this section, we compare our *Buff-CS* method against the two strong baselines (*Reservoir*, and *Class-balance*) on the three combined scenes.

Table 1 reports the performance in terms of pose accuracy averaged over all the scenes after the training is complete, and the coverage score (Eq. 6). Four different buffer sizes are experimented, ranging from $B = 128$ to $B = 1024$. We also present results for the methods with the two types of buffering information, namely *Img-buff* and *Rep-buff*. In addition to the above methods, we report the results of HSCNet in the last row. It is also worth noting that we do not report the 95% confidence interval for the results in Table 1 as in [15,20,22], due to the impractical long training time of our task. However, to support our results, we report the 95 % confidence interval of the accuracy for the results on **i7S** in the ablation study (see Table 4).

We observe that *Buff-CS* achieves the highest coverage score in all settings and outperforms the other two approaches in most of the experiments in terms of the pose accuracy. With the buffer size $B = 256$ and $B = 512$, the higher coverage score of our method yields better pose accuracy on all the three combined scenes. The results indicate that the performance of the *Reservoir* baseline is significantly exceeded by the other two methods due to the low coverage score. We notice that the *Class-balance* method achieves performance comparable to *Buff-CS* with the buffer size $B = 128$ and $B = 1024$. We see that with a large buffer size $B = 1024$, the coverage score for these two approaches are both over 93%, which indicates that nearly all parts of the scene appear in the buffer and the effectiveness of using coverage score is narrowed. Other factors such as the robustness of RANSAC [25] in camera pose estimation also affect the final results. With extremely small buffer size $B = 128$ on **i19S**, both *Class-balance* and *Buff-CS* on an average perform comparably and better than *Reservoir*. When comparing the two buffering strategies *Img-buff* and *Rep-buff* we observe that *Rep-buff* performs better on larger scenes and smaller buffer length. In particular, the biggest performance gap between *Img-buff* and *Rep-buff* is observed in **i19S** and $B = 128$. For larger buffer size, $B = 1024$ the performance is comparable. A detailed analysis of *Rep-buff* is provided in the supplementary. Although the performance of CL approaches lags behind the joint training setting (last row in Table 1), memory (*c.f*. Sec. 5.3) and computational efficiency provides sufficient motivation to pursue visual localization in CL setting.

To make a more detailed comparison among the approaches, we report the accuracy on each scene of i7S after training is complete. Table 2 presents the results for the methods with *Img-buff* and buffer size $B = 256$. Indeed, training on all scenes together as in [37] achieves the best performance. When we train the model in the incremental scenario without buffering, as shown in Table 2, the accuracy on the previously encountered scenes is 0%, which indicates that the visual localization network also suffers from catastrophic forgetting when trained in a continual manner. *Reservoir* exceeds both *Class-balance* and *Buff-CS* methods on Chess, Office, and Redkitchen. This can be attributed to the larger number of samples stored in the buffer corresponding to these scenes, see Fig. 2 for the sample distribution. However, the accuracy drops dramatically on Fire and Heads due to the reduced number of scene samples in the buffer. *Buff-CS* manages to balance the number of samples in the buffer and effectively improves the coverage score compared to *Class-balance*. Thus, it achieves generally better accuracy among all the sampling approaches while maintaining a balanced class distribution in the buffer.

The average accuracy in Table 3 evaluates the performance of the three methods on previous tasks after completing a new task. Table 3 presents the average accuracy

Buffer Size	Buffer methods	i7S			i12S			i19S		
		Coverage Score (average %)	Accuracy (%)		Coverage Score (average %)	Accuracy (%)		Coverage Score (average %)	Accuracy (%)	
			Img-buff	Rep-buff		Img-buff	Rep-buff		Img-buff	Rep-buff
128	Reservoir	26.7	56.8	59.79	58.3	67.51	70.23	39.6	33.2	34.37
	Class-balance	29	59.6	64.25	63.9	74.4	75.72	55.2	47.98	54.12
	Buff-CS (ours)	33.2	61.2	61.13	66.9	75.4	77.33	58.8	46.43	51.86
256	Reservoir	79.3	69.3	69.46	75.4	82.54	85.82	58.6	47.53	49.48
	Class-balance	87.3	70.06	69.76	76.5	85.72	87.36	71.1	64.7	67.31
	Buff-CS (ours)	92.6	72	71.34	86.3	91.85	92.78	76.7	68	70.09
512	Reservoir	91.3	73.4	72.34	89	93.67	94.33	70.5	60.15	62.34
	Class-balance	92.5	74.24	73.6	90.7	95.63	95.63	86.1	78.61	79.06
	Buff-CS (ours)	97.4	75.81	76.06	95.7	96.42	95.85	91.3	80.93	79.51
1024	Reservoir	95.8	75.7	77.07	94.2	97.26	97.14	86.3	80.94	79.5
	Class-balance	96.9	77.09	74.71	96.3	98.43	98.49	93.1	85.36	84.17
	Buff-CS (ours)	98.7	76.89	75.22	97.7	98.9	98.11	96	85.23	85.42
HSCNet (joint training) [37]		100	84.19		100	99.0		100	92.5	

Table 1. Coverage score and accuracy of our method and the two baselines on **i7S**, **i12S**, and **i19S** after the training is complete. The coverage scores are averaged across all the scenes. The best and second best results among approaches are highlighted in blue and red respectively.

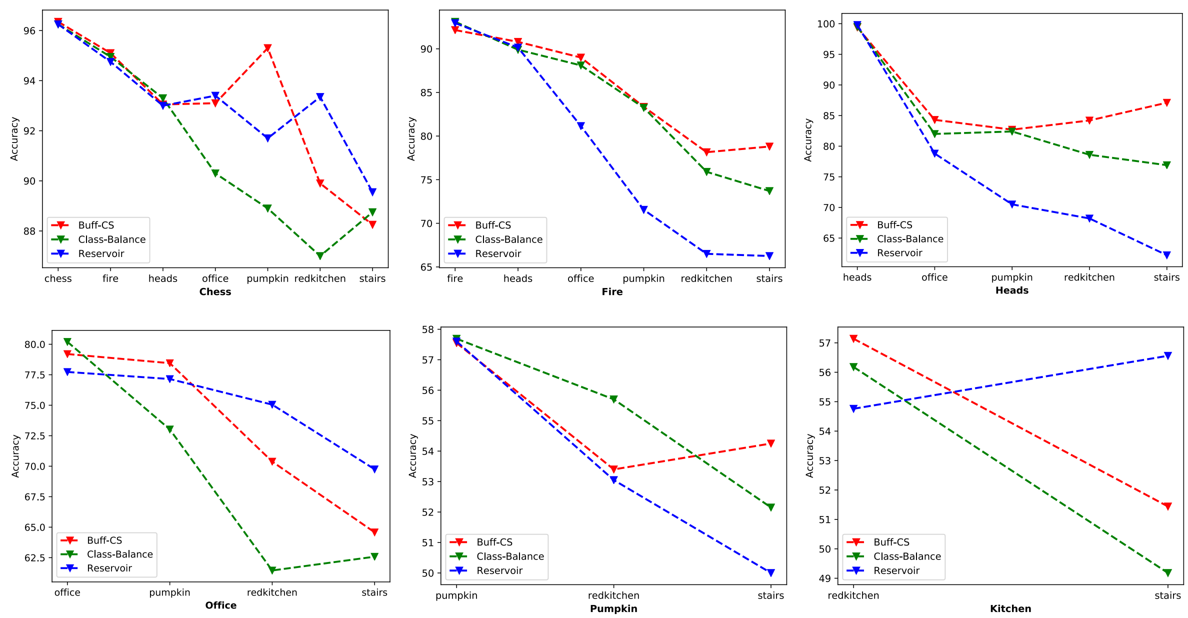

Figure 3. The accuracy (error < 5 cm, 5°) on individual scenes of **i7S** (except for the last scene) at each stage of the training. The x axis indicates the training progress. All methods employs an *Img-buff* buffer of size 256.

for each scene in **i7S** with buffer size $B = 256$. Similar to Table 2, *Reservoir* achieves best performance on Chess, office, and Redkitchen while it drops significantly on Fire, Heads (around 10%) due to the class imbalance. In terms of *overall average*, this method falls behind *Class-balance* and *Buff-CS* by 0.8% and 2.51% respectively. *Class-balance* relieves the problem by balancing the sample distribution. However, it still has weaker results than *Buff-CS* due to the lower coverage score. Fig. 3 shows a more detailed picture in terms of the test accuracy of the three methods after each task is completed. First, we observe that with increasing task length, the performance generally drops across all methods and scenes. This is due to the decreasing of class samples in the buffer. Second, compare to *Class-balance*, *Buff-CS* shows strong performance in the majority of cases. In such a scenario, we believe that the increase of coverage score has a positive effect on test accuracy by providing larger scene observations during the replay process.

5.2. Ablation Study

In this section, we conduct an ablation study to illustrate how different factors affect the performance of the localization system in the continual learning setup. We conduct the experiments on **i7S** with buffer size $B = 256$ and *Img-buff*

Scene	Accuracy (%)				
	W/O Buffering	Revervoir	Class-balance	Buff-CS(ours)	HSCNet [37]
Chess	0.0	91.45	88.50	88.30	97.30
Fire	0.0	67.25	76.35	80.30	96.15
Heads	0.0	69.40	80.40	89.60	98.30
Office	0.0	71.00	62.98	62.08	85.50
Pumpkin	0.0	50.60	51.45	54.25	60.85
Kitchen	0.0	56.14	50.18	49.78	63.74
Stairs	74.30	79.70	80.60	79.70	87.50
Average	10.61	69.30	70.06	72.00	84.19

Table 2. The percentage of accurately localized test images (error < 5 cm, $5°$) on **i7S** with the buffer size $B = 256$, after the training is complete. Here we use *Img-buff* for replay. The best and second best results are highlighted in blue and red respectively.

Scene	Average Accuracy (%)		
	Reservoir	Class-balance	Buff-CS (ours)
Chess	93.14	91.35	93.01
Fire	78.09	83.99	85.38
Heads	75.90	83.88	87.54
Office	74.92	69.31	73.15
Pumpkin	53.55	55.18	55.07
Kitchen	55.66	52.68	54.29
Stairs	78.90	79.40	79.30
Overall Average	72.88	73.68	75.39

Table 3. The average accuracy over different stages of the training process on each scene of **i7S** with the buffer size $B = 256$. Our method has overall better performance compared to the other two methods.

Buff Size = 256	Final Accuracy (%)		
	Reservoir	Class-balance	Buff-CS (ours)
Img-buff	66.99 ± 1.23	69.66 ± 0.54	70.51 ± 0.94
Rep-buff	68.00 ± 1.32	69.12 ± 1.67	70.67 ± 1.10

Table 4. 95% confidence interval of the average accuracy on **i7S** with $B = 256$ after the training is complete. The results are obtained over 5 runs.

information.

Disorder Scenes. We generate random permutations of the scene order being fed to the training network in a continual manner. Results presented in Table 1 of the supplementary shows that the *Buff-CS* performs comparably or better than the baseline methods.

95% confidence interval. We experience intractable training time when trying to report the 95% confidence interval for all of the experiments in Table 1. Thus, only 95% confidence interval of the test set accuracy on **i7S** with buffer size 256 is reported in Table 4. The experiments are run 5 times with different random seeds, and we keep the same seed for all approaches in each run. We observe that the conclusions in Table 1 still holds, *i.e.* our method outperforms the two baselines with both *Img-buff* and *Rep-buff*.

5.3. Training Time and Memory Consumption

Visual localization in continual learning setup aims to achieve data efficiency compared to joint training by learning the tasks sequentially. However, due to catastrophic forgetting, the concept of replay-buffer is used which incurs memory costs of its own. In this section we analyze the memory requirements of buffering different forms of data and compare to the data storage costs of training jointly on all tasks.

To guarantee a fair comparison, all of the experiments are run on NVIDIA Tesla V100 GPUs. We observe that, with buffer size $B = 256$, *Reservoir*, *Class-balance*, and *Buff-CS* require roughly the same amount of time ($\sim 20h$) with both *Img-buff* and *Rep-buff*, which is reasonable since these methods have similar buffering and replay process. When comparing the efficiency between *Img-buff* and *Rep-buff*, we observe that *Rep-buff* is more memory-consuming. Approximately 10 times the more storage space (~ 1091 Mb) is needed compared to *Img-buff* (~ 117 Mb), as it requires larger space for storing dense intermediate cluster predictions.

Compared to the joint training setting which requires to store ~ 35 Gb for **i19S**, the proposed CL method only requires on an average 1.9 Gb and 2.8 Gb with *Img-buff* and *Rep-buff* buffering respectively. From this occupied space, 1.8 Gb corresponds to the average space for task-specific data, while the remaining is allotted to buffer data.

6. Conclusion

In this work we have presented the problem of continual visual localization. A strong baseline is introduced based on experience replay using samples from a small fixed-size buffer. This prevents catastrophic forgetting while learning localization on new scenes. We propose a new buffering strategy that takes into account the 3D scene geometry while keeping a balanced distribution of class samples. The proposed method is evaluated on several indoor localization datasets demonstrating better or competitive performance against the baselines across various settings. Instead of single scene per task, multiple scenes can be considered which makes the problem more challenging and a direction for future work. Although the proposed method balances inter-task data distributions, the above problem setting also requires balancing intra-task data from multiple scenes.

Acknowledgements. This work has been supported by the Academy of Finland (grants 309902, 327911). The authors would like to acknowledge the computational resources provided by the Aalto Science-IT project and CSC-IT Center for Science, Finland.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *ECCV*, 2018. 1, 2, 3

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 2017. 1, 2, 3

[3] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *NeurIPS*, 2019. 3

[4] Vassileios Balntas, Shuda Li, and Victor Adrian Prisacariu. RelocNet: Continuous metric learning relocalisation using neural nets. In *ECCV*, 2018. 3

[5] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O. Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. In *ECAI*, 2020. 3

[6] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC - Differentiable RANSAC for camera localization. In *CVPR*, 2017. 3

[7] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In *CVPR*, 2016. 3

[8] Eric Brachmann and Carsten Rother. Learning less is more - 6D camera localization via 3D surface regression. In *CVPR*, 2018. 3, 4

[9] Eric Brachmann and Carsten Rother. Expert sample consensus applied to camera re-localization. In *ICCV*, 2019. 1, 3, 4, 5

[10] Eric Brachmann and Carsten Rother. Neural-guided RANSAC: Learning where to sample model hypotheses. In *ICCV*, 2019. 1, 3

[11] Eric Brachmann and Carsten Rother. Visual camera relocalization from RGB and RGB-D images using DSAC. *arXiv:2002.12324*, 2020. 1, 3

[12] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-aware learning of maps for camera localization. In *CVPR*, 2018. 1, 3

[13] Ignas Budvytis, Marvin Teichmann, Tomas Vojir, and Roberto Cipolla. Large scale joint semantic re-localisation and scene understanding via globally unique instance coordinate regression. In *BMVC*, 2019. 3

[14] Mai Bui, Shadi Albarqouni, Slobodan Ilic, and Nassir Navab. Scene coordinate and correspondence learning for image-based localization. In *BMVC*, 2018. 3

[15] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. In *NeurIPS*, 2020. 6

[16] Tommaso Cavallari, Luca Bertinetto, Jishnu Mukhoti, Philip Torr, and Stuart Golodetz. Let's take this online: Adapting scene coordinate regression network predictions for online RGB-D camera relocalisation. In *3DV*, 2019. 3

[17] Tommaso Cavallari, Stuart Golodetz, Nicholas A Lord, Julien Valentin, Luigi Di Stefano, and Philip HS Torr. On-the-fly adaptation of regression forests for online camera relocalisation. In *CVPR*, 2017. 3

[18] Tommaso Cavallari, Stuart Golodetz, Nicholas A Lord, Julien Valentin, Victor A Prisacariu, Luigi Di Stefano, and Philip HS Torr. Real-time rgb-d camera pose estimation in novel scenes using a relocalisation cascade. *PAMI*, 42(10):2465–2477, 2019. 3

[19] Arslan Chaudhry, Albert Gordo, Puneet K. Dokania, Philip Torr, and David Lopez-Paz. Using hindsight to anchor past knowledge in continual learning. In *AAAI*, 2021. 1, 2

[20] Arslan Chaudhry, Naeemullah Khan, Puneet Dokania, and Philip Torr. Continual learning in low-rank orthogonal subspaces. In *NeurIPS*, 2020. 5, 6

[21] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2018. 3

[22] Aristotelis Chrysakis and Marie-Francine Moens. Online continual learning from imbalanced data. In *ICML*, 2020. 2, 3, 5, 6

[23] Mingyu Ding, Zhe Wang, Jiankai Sun, Jianping Shi, and Ping Luo. CamNet: Coarse-to-fine retrieval for camera relocalization. In *ICCV*, 2019. 3

[24] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. In *arXiv preprint arXiv:1701.08734*, 2017. 3

[25] Martin A Fischler and Robert C Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *CACM*, 24(6):381–395, 1981. 3, 6

[26] Abner Guzmán-Rivera, Pushmeet Kohli, Ben Glocker, Jamie Shotton, Toby Sharp, Andrew W. Fitzgibbon, and Shahram Izadi. Multi-output learning for camera relocalization. In *CVPR*, 2014. 3

[27] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *arXiv preprint arXiv:1503.02531*, 2015. 3

[28] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. In *NeurIPS workshops*, 2018. 2, 3

[29] Ahmet Iscen, Jeffrey Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. In *ECCV*, 2020. 3

[30] Khurram Javed and Martha White. Meta-learning representations for continual learning. In *NeurIPS*, 2019. 3

[31] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *ICRA*, 2016. 1, 3

[32] Alex Kendall and Roberto Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017. 1, 3

[33] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DoF camera relocalization. In *ICCV*, 2015. 1, 3

[34] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[35] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *PNAS*, 114(13):3521–3526, 2017. 1, 2, 3

[36] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia, and Juho Kannala. Camera relocalization by computing pairwise relative poses using convolutional neural network. In *ICCV Workshops*, 2017. 3

[37] Xiaotian Li, Shuzhe Wang, Yi Zhao, Jakob Verbeek, and Juho Kannala. Hierarchical scene coordinate classification and regression for visual localization. In *CVPR*, 2020. 1, 3, 4, 5, 6, 7, 8

[38] Xiaotian Li, Juha Ylioinas, and Juho Kannala. Full-frame scene coordinate regression for image-based localization. In *RSS*, 2018. 3

[39] Xiaotian Li, Juha Ylioinas, Jakob Verbeek, and Juho Kannala. Scene coordinate regression with angle-based reprojection loss for camera relocalization. In *ECCV Workshops*, 2018. 3

[40] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *NeurIPS*, 2017. 3

[41] Daniela Massiceti, Alexander Krull, Eric Brachmann, Carsten Rother, and Philip HS Torr. Random forests versus neural networks - What's best for camera localization? In *ICRA*, 2017. 3

[42] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Image-based localization using hourglass networks. In *ICCV Workshops*, 2017. 1, 3

[43] Lili Meng, Jianhui Chen, Frederick Tung, James J Little, Julien Valentin, and Clarence W de Silva. Backtracking regression forests for accurate camera relocalization. In *IROS*, 2017. 3

[44] Lili Meng, Frederick Tung, James J Little, Julien Valentin, and Clarence W de Silva. Exploiting points and lines in regression forests for RGB-D camera relocalization. In *IROS*, 2018. 3

[45] Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *ICLR*, 2018. 2, 3

[46] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 2020. 3

[47] Noha Radwan, Abhinav Valada, and Wolfram Burgard. VLocNet++: Deep multitask learning for semantic visual localization and odometry. *RA-L*, 3(4):4407–4414, 2018. 1, 3

[48] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017. 2, 3

[49] Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. In *ICLR*, 2018. 2, 3

[50] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 1, 2

[51] Paul-Edouard Sarlin, Frédéric Debraine, Marcin Dymczyk, Roland Siegwart, and Cesar Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. In *CoRL*, 2018. 1, 2

[52] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2D-to-3D matching. In *ICCV*, 2011. 1, 2

[53] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012. 1, 2

[54] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *PAMI*, 39(9):1744–1756, 2016. 1, 2

[55] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. Benchmarking 6DoF outdoor visual localization in changing conditions. In *CVPR*, 2018. 1, 2

[56] Torsten Sattler, Akihiko Torii, Josef Sivic, Marc Pollefeys, Hajime Taira, Masatoshi Okutomi, and Tomas Pajdla. Are large-scale 3D models really necessary for accurate visual localization? In *CVPR*, 2017. 1, 2

[57] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-Taixe. Understanding the limitations of CNN-based absolute camera pose regression. In *CVPR*, 2019. 1, 3

[58] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1, 2

[59] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 1, 2

[60] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *CVPR*, 2013. 3, 5

[61] Linus Svärm, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson. City-scale localization for cameras with known vertical direction. *PAMI*, 39(7):1455–1461, 2016. 1, 2

[62] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomás Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *CVPR*, 2018. 1, 2

[63] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *ICRA*, 2018. 1, 3

[64] Julien Valentin, Angela Dai, Matthias Nießner, Pushmeet Kohli, Philip Torr, Shahram Izadi, and Cem Keskin. Learning to navigate the energy landscape. In *3DV*, 2016. 5

[65] Julien Valentin, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. Exploiting uncertainty in regression forests for accurate camera relocalization. In *CVPR*, 2015. 3

[66] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985. 2, 3, 5

[67] Florian Walch, Caner Hazirbas, Laura Leal-Taixe, Torsten Sattler, Sebastian Hilsenbeck, and Daniel Cremers. Image-based localization using LSTMs for structured feature correlation. In *ICCV*, 2017. 1, 3

[68] Luwei Yang, Ziqian Bai, Chengzhou Tang, Honghua Li, Yasutaka Furukawa, and Ping Tan. SANet: Scene agnostic network for camera localization. In *ICCV*, 2019. 1, 3

[69] Lei Zhou, Zixin Luo, Tianwei Shen, Jiahui Zhang, Mingmin Zhen, Yao Yao, Tian Fang, and Long Quan. KFNet: Learning temporal camera relocalization using Kalman filtering. In *CVPR*, 2020. 3

[70] Qunjie Zhou, Torsten Sattler, Marc Pollefeys, and Laura Leal-Taixe. To learn or not to learn: Visual localization from essential matrices. In *ICRA*, 2020. 3