Multi-Model Continual Learning for Camera Localisation from Aerial Images

Aldrich A. Cabrera-Ponce, Manuel Martin-Ortiz, and Jose Martinez-Carranza*

*Instituto Nacional de Astrofisica, Optica y Electronica (INAOE), Puebla, Mexico
Benemerita Universidad Autonoma de Puebla (BUAP), Puebla, Mexico

http://www.imavs.org/

Abstract

Camera localisation is the problem of estimating the camera pose from one or more images. Deep Learning (DL) has become one of the most popular methods with convolutional neural networks (CNN) to train models and thus obtain the camera pose in a scenario. However, such methods assume a scenario where the user must wait for the model to become available once ready after training, which has to be trained off-line and involving high computational processing time. This work presents a multi-model continuous learning approach for camera localisation that uses multiple models learned incrementally over time. Each model is learned for a certain segment of the camera trajectory, and a search engine based on histogram colour and keyframes to identify the corresponding learned model w.r.t. the current camera view. The continual learning process is a rehearsal strategy that repeats the old data with the new ones in a latent replay layer. On average, our performance achieved a processing speed of 150 fps with an accuracy of 0.71 to 0.85 for the camera pose estimation. We demonstrate the effectiveness of our system in a geo-localisation application where a model learns to associate aerial images to GPS coordinates continuously using our proposed strategy. The main goal is to provide multiple on-line learned models during the same flight as an alternative geo-localisation method that could prove helpful in cases of GPS failure.

1 Introduction

Camera relocalisation is the problem of obtaining the 6 degrees of freedom (DoF) concerning one or multiple images captured from a scenario. Visual localisation methods have been carried out to solve this problem using the images from a monocular camera. For example, some works acquire the camera's poses using feature matching [1], nearest neighbours [2], visual odometry (VO) [3], and simultaneous localisation and mapping (SLAM) systems [4]. The performance of these techniques has been adequate to acquire information that can solve the visual localisation problem. Nevertheless, with the recent increase in deep learning, works have addressed this problem to directly regress the camera pose from an image with a training model [5].

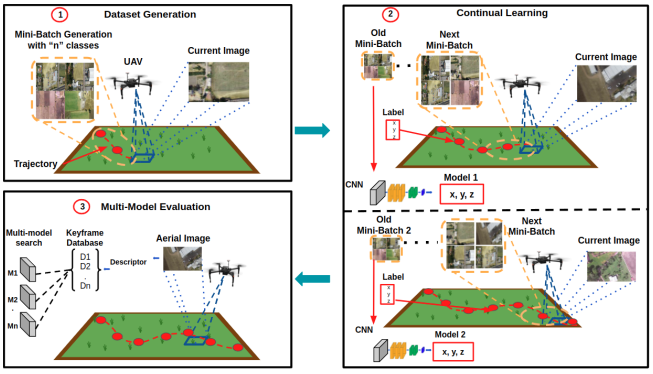

Figure 1: Multi-model continual learning for camera localisation. The framework consists of 3 steps: 1) Mini-batches generation; 2) On-line training using the mini-batches while new ones are created; 3) Frame search based on histogram colour and keyframes to identify the corresponding model. A video illustrating this process can be found at video link.

To carry out the aforementioned, models are trained from a dataset with labels of the camera positions to regress the 6-DOF camera pose from a single RGB image [6]. Nonetheless, traditional training has the limitation of not having a learning model ready for the same data collection mission. To mitigate this, it has been established to speed up the training time and pose estimation, reducing the architecture of a network [7]. On the other hand, continual training aims to learn the information on the fly, reducing the catastrophic forgetting in the training process [8].

With the above mentioned, a compact network and a method that reduces forgetting have achieved relevant results under scenarios requiring a fast training process. As a result, continual learning methods reduce the loss in the training process by keeping the information to a minimum during the same collection data mission, getting results simultaneous. Thus, a model is updated with current data without forgetting previous ones during training. Therefore, continual

*Email address(es): aldrich.cabrera@alumno.buap.mx

learning has a high impact in the area of robotics used in mobile applications with limited storage capacity [9].

Motivated by the above, we consider using a network based on a continual learning strategy with a rehearsal method, allowing the creation of multiple models with information about the poses in a specific area. We adopt the work presented in [10] to repeat the previous data in a latent layer and merge it with the current information. Besides, based on localisation modules in SLAM systems, an image is localised using feature matching between its descriptors and a bag of words (BoW). Thus, we propose a frame searcher to determine the keyframe corresponding to the camera view and identify the learned model w.r.t. the current view (Figure 1). Finally, we argue that using multi-models with information from the camera trajectory segments can give us a result close to the value of the Ground Truth, allowing re-localise a camera.

In order to present this work, our paper is organised as follows. Section 2 provides a literature study of the continual learning approaches for classification and localisation tasks. Section 3 describes the process of generating a multi-model system using the AR1* continual learning method and the generation of mini-batches. The experimental design and results obtained compared with ORB-SLAM2 are conveyed in Section 4. Finally, conclusions and future work are outlined in Section 5.

2 RELATED WORKS

Image-based localisation has become an efficient method in several robotics tasks, obtaining a camera's pose using computer vision methods. For example, a UAV's camera can capture images to determine the position in a scenario, thus achieving its localisation. ORB descriptors are a technique used for camera localisation by building a map of the scenario [4]. On one side, deep learning have been used to get the camera's pose using regression in combination with 3D structures [11], visual odometry [12] and SLAM systems [13, 14] in monocular, stereo, and RGB-D cameras. These works employ a preprocessing that aligns in-depth features with a reference 3D model, crops sub-images, and uses inverse depth to learn better the image's shape, appearance and feature.

However, the works aforementioned require 3D models, crop images and the use of depth to achieve an accurate result. On the other hand, PoseNet [5] is a CNN designed to estimate the camera's pose from a single image without using depth and 3D models. Therefore, several works have used PoseNet to improve and change its architecture to get a fast and accurate pose estimation. For example, in [7] shows a brief analysis of these architectures, proposing a compact network to accelerate the learning process and pose estimation. Nevertheless, PoseNet is an end-to-end architecture that involves collecting a large dataset. Thus, new works have focused on continuous training within the robotics [9] to learn new data without forgetting the previous ones in detection and classification tasks [15, 16].

The concept of continual learning has led to creating datasets from a sequence of partial experiences where all data is unavailable at once. For example, in [17] developed CoRE50, a dataset for continual learning in object recognition tasks, and [18] presents CLRS dataset for scene classification, partitioning the training batch. However, the catastrophic forgetting is still unsatisfactory in results, arguing the design of a method to restrict this loss. Unlike these works, an elastic network with a multistable behaviour can allow the development of different configurations to learn continuously without forgetting the previous states, avoiding thus the knowledge forgetting [19].

Another work presents a continual SLAM [20] proposing a dual-network based on a VO model to produce online odometry estimates and generalise the long-term learning, using consecutive frames, image warping, and dense 3D map. However, since this method achieves the localisation of a camera by merging 3D depth maps, it may fall into an estimation error if the depth is not acquired correctly. To avoid this, [21] analyses and identifies previously unobserved phenomena of gradient-based optimisation using visual data and an adaptable buffer for the geo-localisation of an image. Besides, they keep the images with lower noisy time stamps in the form of the default date and time for training.

Finally, rehearsal methods based on repeating the previous data for visual localisation tasks achieve promising results. For example, using an adaptive buffer keeps samples in subsets to update the training model over current information and previous ones [22]. In [10], presents the AR1* method with a latent replay layer to retain volumes of information of the previous data as new information is created in mini-batches. The latter has been carried out in a localisation methodology using multicamera learning and updating the model while acquiring new information [23]. Motivated by these works, we implemented a multi-model approach based on a continual learning method called AR1*, using a latent replay layer for camera localisation. In addition, we built a frame search based on histogram colour and keyframes to identify the corresponding model.

3 METHODOLOGY

This section presents the methodology carried out to develop this work. We carry out our multi-model continual learning framework in 3 steps. Firstly, we generated the dataset with aerial images and GPS coordinates into mini-batches for the training. Afterwards, we use a colour histogram like an image descriptor to determine the corresponding keyframe. Thus, it will indicate which model corresponds with the image to perform the evaluation. Finally, we present the continual training with the latent replay strategy.

3.1 Mini-Batches generation

For the dataset generation, we collected aerial images with GPS information through the Robotic Operating System

http://www.imavs.org/

http://www.imavs.org/

(ROS) and our Ground Control Station (GCS), acquiring the frames from the video stream of the UAV. Then, we manually carried out 4 flight missions to obtain the dataset, splitting it into small sets of images with dimensions of 128×128 and converting the GPS to metres. Then, we generate the labels when we create a new class with a number of pictures defined with its labels in terms of indexes and create the mini-batches every two classes. Furthermore, since the network is an architecture based on a classification model, we define a representative mean flight coordinate for each class, merging the individual coordinates into one. This way, we store the mean flight coordinates in a file to represent the classes and UAV trajectory.

To better understand the mean flight coordinates stored in the file, we show the coordinates of one of the trajectories in Table 1, where the index represents the class belonging to a part of the path. Thus when the model gives us a result in terms of index, we look for in the text file to regress the mean flight coordinate and geolocalise the camera. Also, we show in Figure 2 the trajectories made in the flight missions with the UAV where we collected the data.

Table 1: Distribution of the classes associated with indexes representing the labels with discrete flight coordinates.

Index	Mean flight coordinates
0	x: -0.28260, y: 15.8941, z: 100.81
1	x: -14.3834, y: 50.8135, z: 100.94
2	x: -233.215, y: 228.076, z: 100.71
3	x: -351.116, y: 288.614, z: 100.34
4	x: -248.701, y: 341.021, z: 100.38
5	x: -125.802, y: 252.147, z: 100.52
6	x: 60.32710, y: 145.127, z: 100.50

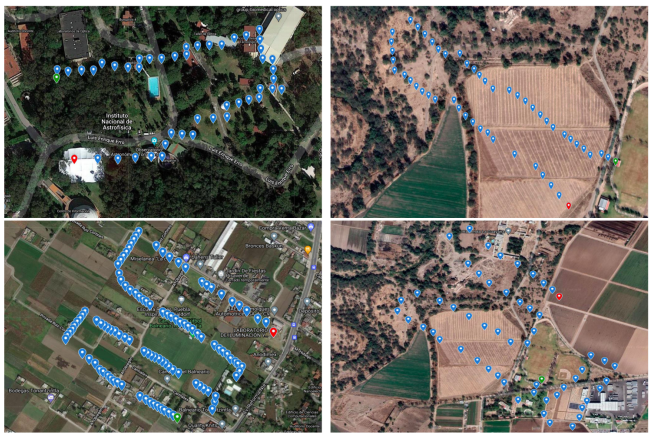

Figure 2: Manually performed flight trajectories. Each blue waypoint corresponds to the GPS coordinates; the red waypoint represents the beginning and the green the end of the trajectory.

3.2 Histogram image descriptor

To create the searcher that allows us to choose the model for the evaluation correctly, we propose using a simple descriptor based on the colour histogram of the image. This histogram applies a colour distribution in the pictures, causing them to have similar distributions or equal even in those that are not. Therefore, we perform the following steps to generate a robust descriptor to deal with this problem. The steps are described in a list to understand the procedure performed:

1. We convert the image colour space from BGR to HSV (Hue, Saturation, Value) to obtain robustness in the histogram.
2. We define the number of bins in our histogram with a dimension of 8 bins for the Hue, 12 bins for saturation, and 3 bins for the value channel. Total dimension of the feature vector: $8 \times 12 \times 3 = 288$.
3. We calculate a mask from the centre of the image and divide it into quadrants. Thus, we calculate the histogram in 5 regions of the image instead of a global one.
4. We update the features extracted from the five quadrants with values from 0 to 255.
5. Finally, the images are represented and quantified using a list of 288 floating-point numbers.

In this way, we calculate the colour histogram from a divided image into five different regions: 1) the top-left corner; 2) top-right; 3) bottom-right; 4) bottom-left; 5) and the centre of the image. We show this example in Figure 3. Once the histogram is obtained, we calculate the features for each keyframe that we will use to find the model. Thus, we extract the floating points from the histogram and save them in a list with their respective features. In addition, we have a feature vector from 5 regions with a dimensionality of 5 x 288 = 1440, where each image will be quantified and represented using 1440 numbers.

Finally, we compare the frame with the keyframe to find the match between their features. That is to say, we extract the histogram information of the current frame and compare it with the keyframe descriptor using the chi2 distance. Thus, we get histogram information from the five regions on the images, ordering the results from lowest to highest, where a value close to zero represents the similarity with the keyframe. Then, we load the model corresponding to this keyframe to start the evaluation. Figure 4 presents a diagram of this process to find the model through a distance metric, taking two feature vectors as inputs. It is worth mentioning that we create the keyframes just when we generate the mini-batches, so if the value of the histograms does not match, it will return a model that does not correspond to the current camera view.

Figure 3: Image divided in 5 regions.

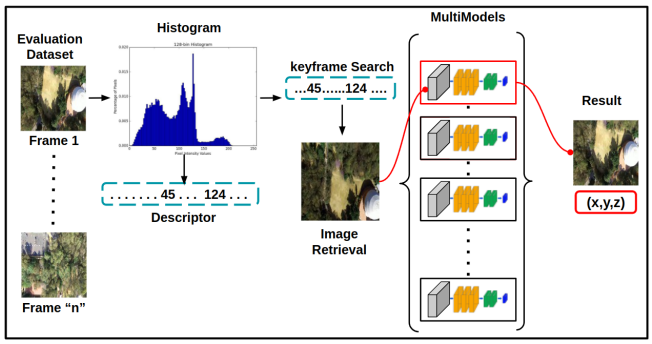

Figure 4: Overview to search for the keyframe with an evaluation frame using the descriptor vector to identify the corresponding model.

3.3 *Continual Training*

For multi-model continual training, we rely on the architecture presented in the work [10], showing a method of latent replay in the hidden layer *pooling6*. This process consists of training a dataset from mini-batches whose classes are spread in them, storing a portion of activation volumes belonging to each class. In the training process, the method slows down learning in the lower layers and leaves the upper layers free. That is to say, it slows down the learning at the layers below the pooling6 layer, saving a small fraction of patterns for each mini-batch. Thus, when new information arrives (a new mini-batch), the memory with previous data is merged and updated with the current data. In this way, the patterns of both mini-batches are kept constant by repeating the data in the latent layer. Finally, we perform this process until the last mini-batch generated will train.

This method, called Auto-Regressive Model (AR1*), is based on a rehearsal strategy which consists of the continuous repetition of previous data to avoid catastrophic forgetting of the learning. Each mini-batch represents new information generated from the dataset, training them one by one. We thus update a learning model when new data arrives, generating a single model with all the information learned. However, to develop our multi-model approach, we create a model for each mini-batch learned, arguing that each has information more accurate to the current data than the previous one. Finally, we have several models corresponding to the mini-batches for evaluation tests.

To carry out the continual training, we take the following parameters: 1) SGD Optimiser; 2) Batch size: 128; 3) Epochs: 4; 4) Cross-Entropy as loss function; 5) Learning Rate: 0.001; 6) Pool6 as latent layer; 7) 1500 patterns to storage in the external memory. These patterns are the information acquired from the previous and current data to update the learning model.

4 Experiments and Results

Hereafter, we present the experiments to compare the accuracy and processing speed results using a learning model and our multi-model approach. We should note that the search engine for a keyframe using a descriptor based on the colour histogram of the image is essential to establish the chosen model. Given this scenario, we perform the experiments on Ubuntu 20.04, Python 3.6, PyTorch 1.4.0 for the network training and evaluation, OpenCV 3 and ROS Noetic on a computer with an Nvidia Geforce 960 M with 8 GB of RAM.

4.1 *Histogram Results*

The first experiment compares the search result using three bins to generate the histogram. Thus, we calculate the histogram of each evaluation dataset image of trajectory one and search using the chi2 distance for the corresponding similarity to obtain the model. Table 2 shows the number of times the search acquired the correct model and a success rate indicating how effective the search is with the defined bins.

Table 2: Results of the searching model using a descriptor vector.

Bins	Correct	Success Rate
(10, 5, 3)	78	54.16 %
(12, 16, 3)	111	77.08 %
(8, 12, 3)	125	**86.90 %**

This result indicates that the third set bins achieve a better result w.r.t colour histogram, finding the correct keyframe using the descriptor to identify the corresponding model. However, we argue that a descriptor based on SIFT and ORB could improve this search result using visual and binary features corresponding to the image.

4.2 *Evaluation with a Model*

In the second experiment, we used a single learning model without the descriptor searching that determines the keyframe, training and updating the weights in the model

http://www.imavs.org/

when new mini-batches arrive in the network. Thus, the previous and current classes' patterns are replayed in the latent layer, updating the features and generalising the knowledge for each mini-batch trained. To illustrate the accuracy results in this experiment, we evaluate the CNN with the evaluation data of the 4 flight paths, obtaining an index corresponding to the class label with its mean flight position. Then, in Table 3, we present the results obtained, showing the number of frames present in the evaluation dataset, the number of correctly classified frames, the accuracy, and the processing speed per image expressed in fps.

Table 3: Accuracy results with the evaluation dataset using a single model.

Trajectory	Frames	Correct	Accuracy	Fps
1	144	89	0.6180	152.397
2	117	73	0.6239	153.044
3	84	70	0.8333	154.096
4	606	378	0.6237	151.564

We note that training presents an accuracy of around 0.62 in most of the paths whose traversed trajectory lengths are: 1.0 km for the first, 3.0 km in the second, 4.6 km for the third, and 6.4 km for the last. The best result was in the third flight achieving an accuracy of 0.83 with a high amount of frames evaluated correctly. Hence, we argue that the algorithm is limited in storing the volumes of information, causing catastrophic forgetting to occur the more classes it learns, even using a latent layer. Consequently, we expose that a multi-model approach for the camera localisation can have a better result using exclusively on the model corresponding to the camera view class.

4.3 Evaluation with a Multi-Model Approach

In the last experiment, we evaluated our multi-model approach with the keyframe searching to identify the corresponding model to the class and camera view. We used the bins of (8,12,3) analysed in the first experiment to calculate the histogram and create the feature vector descriptor. Afterwards, we compare our approach with the ORB-SLAM2 re-localisation module, presenting the accuracy and processing speed results. To compare us with ORB-SLAM2, we created a map with the training dataset and localised the evaluation images with the ORB-SLAM2 re-localisation module to find the coordinates w.r.t the Ground-Truth. Thus, we take the distance between frame and keyframe to determine whether the coordinate corresponds to a close keyframe. Finally, in Table 4, we present the results obtained from the four trajectories and the comparison with ORB-SLAM2, and in Table 5, we show the processing speed results.

These results show that ORB-SLAM2 presents a low localise on trajectory 1, indicating that the scenario is complex to obtain a localisation close to the training dataset. Furthermore, the ORB-SLAM2 re-localisation module does not find ORB descriptors that match the evaluation images, arguing that the scenario presents elements that confuse the system. Likewise, path 4 has an accuracy just above 0.5, meaning that half of the path cannot correctly locate the camera. Nonetheless, our multi-model system finds some coordinates of the evaluation images, localising thus the camera in these areas.

Table 4: Comparison accuracy results with the evaluation dataset using a multi-model and ORB-SLAM2.

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
ORB-SLAM2	0.034	0.913	0.869	0.584
Multi-Model	0.735	0.817	0.856	0.714

Table 5: Comparison processing speed results with our multi-model approach and ORB-SLAM2 (Fps).

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
ORB-SLAM2	85.47	83.33	92.59	89.28
Multi-Model	150.19	151.65	153.27	153.55

For the above, we argue that a multi-model approach using continual learning based on latent replay achieves a better result by identifying the corresponding model with the camera view. Besides, searching keyframes can recover the model created in that trajectory segment using only the descriptor vector without geometry calculations or depth images. Eventually, the processing speed tests show a superiority in terms of time to run the algorithm, which we can use during flight missions with a UAV in external scenarios.

To better understand the results obtained in the earlier experiments, we present in Table 6 the RMSE metric to observe how far the obtained values are from the ground truth. Thus, a minimum value represents a better fit to the accuracy measure with which the algorithm performs the evaluations. Note that our approach presents significant results in contrast to using a single model for camera localisation tasks. Finally, it should be noted that if the UAV enters an unknown area or outside the training area, the algorithm gives erroneous data. To deal with this, we plan to use a flag where we determine that the UAV has left the zone and thus extend the training with the following area data.

Table 6: Results of the RMSE metric using the 3 approaches performed in the experiments: with a single model, ORB-SLAM2, and our multi-model approach.

Approach	Traj. 1	Traj. 2	Traj. 3	Traj. 4
ORB-SLAM2	5.9389	0.4090	1.8563	8.2484
Single Model	2.1730	2.3056	4.4507	7.8642
Multi-Model	1.8671	1.6641	2.7080	7.0858

http://www.imavs.org/

5 CONCLUSIONS

We have presented a continual learning approach to localise a monocular camera using multi-models and a keyframe search engine to identify the representative model given the current camera view. Our goal has been to use the histogram colour of the images in 5 regions to search a close keyframe and choose the learned model, obtaining thus a label with a mean flight coordinate for camera localisation. Furthermore, it helps to extract and match similar features between the current camera view and keyframe, giving access to the corresponding model to evaluate the image instead of using a global model. In addition, the advantage of continual learning in a latent replay approach allows for learning data in small sets, maintaining previous information while updating the model with new data.

The proposed algorithm exhibits a suitable accuracy for camera localisation in combination with an image searching strategy, being efficient in small, medium and large trajectories. Furthermore, we have also demonstrated the applicability of a rehearsal strategy to improve the localisation using our scheme compared with SLAM systems. Overall, we achieve an accuracy result of 0.73 for short trajectories and 0.71 for long ones while maintaining a fast processing speed. Nonetheless, the feature descriptor is a topic to discuss because if the colour histogram returns a different keyframe, the model will give us another result and thus a wrong position.

Finally, we have observed that tests performed with continual learning based on mini-batches can be used to train new information in unseen areas while maintaining knowledge of the previous scenario. To this end, we propose extending the experiments to different environments to test the algorithm's effectiveness when the UAV visit new zones. In addition, we plan to use the method with a regression network to estimate the full 6 DoF camera position for future work.

http://www.imavs.org/

ACKNOWLEDGEMENTS

The first author is thankful for his scholarship funded by Consejo Nacional de Ciencia y Tecnología (CONACYT) under the grant 802791.

REFERENCES

[1] Peter Hansen, Peter Corke, and Wageeh Boles. Wide-angle visual feature matching for outdoor localization. *The International Journal of Robotics Research*, 29(2-3):267–297, 2010.

[2] David Wong, Daisuke Deguchi, Ichiro Ide, and Hiroshi Murase. Single camera vehicle localization using feature scale tracklets. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 100(2):702–713, 2017.

[3] Ramon Gonzalez, Francisco Rodriguez, Jose Luis Guzman, Cedric Pradalier, and Roland Siegwart. Combined visual odometry and visual compass for off-road mobile robots localization. *Robotica*, 30(6):865–878, 2012.

[4] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[5] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[6] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[7] Aldrich A Cabrera-Ponce and Jose Martinez-Carranza. Convolutional neural networks for geo-localisation with a single aerial image. *Journal of Real-Time Image Processing*, pages 1–11, 2022.

[8] Felix Wiewel and Bin Yang. Localizing catastrophic forgetting in neural networks. *arXiv preprint arXiv:1906.02568*, 2019.

[9] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Dıaz-Rodrıguez. Continual learning for robotics. *arXiv preprint arXiv:1907.00182*, pages 1–34, 2019.

[10] Lorenzo Pellegrini, Gabriele Graffieti, Vincenzo Lomonaco, and Davide Maltoni. Latent replay for real-time continual learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10203–10209. IEEE, 2020.

[11] Paul-Edouard Sarlin, Ajaykumar Unagar, Mans Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, et al. Back to the feature: Learning robust camera localization from pixels to pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3247–3257, 2021.

[12] Felix Ott, Tobias Feigl, Christoffer Loffler, and Christopher Mutschler. Vipr: visual-odometry-aided pose regression for 6dof camera localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 42–43, 2020.

[13] Shilang Chen, Junjun Wu, Qinghua Lu, Yanran Wang, and Zeqin Lin. Cross-scene loop-closure detection with continual learning for visual simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 18(5):17298814211050560, 2021.

[14] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in Neural Information Processing Systems*, 34, 2021.

[15] Keval Doshi and Yasin Yilmaz. Continual learning for anomaly detection in surveillance videos. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 254–255, 2020.

[16] Kishan Parshotam and Mert Kilickaya. Continual learning of object instances. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–225, 2020.

[17] Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR, 2017.

[18] Haifeng Li, Hao Jiang, Xin Gu, Jian Peng, Wenbo Li, Liang Hong, and Chao Tao. Clrs: Continual learning benchmark for remote sensing image scene classification. *Sensors*, 20(4):1226, 2020.

[19] Menachem Stern, Matthew B Pinson, and Arvind Murugan. Continual learning of multiple memories in mechanical networks. *Physical Review X*, 10(3):031044, 2020.

[20] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning. *arXiv preprint arXiv:2203.01578*, 2022.

[21] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8281–8290, 2021.

[22] Shuzhe Wang, Zakaria Laskar, Iaroslav Melekhov, Xiaotian Li, and Juho Kannala. Continual learning for image-based camera localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3252–3262, 2021.

[23] Aldrich A Cabrera-Ponce, Manuel Martin-Ortiz, and J Martinez-Carranza. Continual learning for multi-camera relocalisation. In *Mexican International Conference on Artificial Intelligence*, pages 289–302. Springer, 2021.

http://www.imavs.org/