TU/e | Bachelor Thesis
2023

Impact of the Buffer Sampling Method and Buffer Size on Rehearsal Based Continual SLAM

Full Name	Student ID
Matiss Šimanskis	1650076

Eindhoven, June 19, 2023

Contents

1 Abstract **1**

2 Introduction **2**

3 Related Work **3**

4 Experimental Setup **4**
4.1 Continual SLAM Overview 4
4.2 Datasets 4
4.3 Replay Buffer Size 4
4.4 Replay Buffer Sampling Strategy 4

5 Empirical Evaluation **7**

6 Conclusion **10**

7 References **11**

A Additional results **13**

1 | Abstract

Using Simultaneous Localization and Mapping (SLAM), entities such as robots or vehicles are able to map their environment around them while simultaneously identifying where they are located on this map. Self-supervised deep learning has allowed the problem of SLAM to be tackled in an competent manner. However, in the case this entity enters a new environment and starts adapting to it, catastrophic forgetting[8] happens where the entity forgets what it has learned about the previous environment. The authors of CL-SLAM[26] have done a great job of addressing this problem using a rehearsal, dual-network approach. Although, the rehearsal strategy that is used is not optimal as there is no limit to the amount of samples stored in the replay buffer. To address this gap I explored what the impact of having a fixed buffer size would have on CL-SLAM performance. Specifically I address the questions "What is the impact of the replay buffer's size on rehearsal based Continual SLAM?" and "What is the impact of the replay buffer's sampling method on rehearsal based Continual SLAM?". What was found is that using a fixed replay buffer size actually performs better than the default infinite one. Reservoir[25] sampling performed the best compared to two other sampling strategies and a buffer size of 1000 was found to be optimal when compared to six other replay buffer sizes. The code can be found at https://github.com/Gashpiii/CL-SLAM/tree/tue_thesis.

2 | Introduction

Simultaneous Localization and Mapping (SLAM) is the problem of an entity such as a robot or vehicle moving about a space and creating a global map of it's environment while also identifying where it is located with respect to this map. It has been big topic of research for many years and has recently began to be used in practise in fields such as autonomous driving due to the improvements in computer processing speeds. Before deep learning technology was advanced as it is now, classical SLAM algorithms such as GraphSLAM[24] were the main focus. They typically depend on handcrafted low-level features that have a tendency to fail under challenging conditions[4]. Deep learning based approaches such as ORB-SLAM2[18] and UnDeepVO[12] reduce such problems due to their ability to learn high-level features[16]. A problem still remains in the deep learning approach in the case when the entity visits an area that it has not been trained on, it is unable to generalise what it has learned previously and apply it. In this case, there still is the option to train the entity in the new environment, however as the robot learns it's new environment it forgets what it has learned about it's previous environments. This problem is known as catastrophic forgetting. CL-SLAM[26] address this problem by adapting continual learning techniques that have been used in classification.

Continual learning in classification, also known as class incremental learning or task adaptation, is the process of learning new tasks while also being able to perform old tasks with minimal performance drop. Despite the problem of not forgetting previously seen environments being one of domain adaptation, some of the methods in task adaptation are still applicable. There has been extensive research done in task adaptation and discovered methods include setting task boundaries[13, 11], knowledge distillation[10], replay mechanisms[21], and dual-networks[7]. CL-SLAM[26] use two of the task adaptation techniques mentioned, namely replay mechanisms and a dual-network architecture. Replay mechanisms are methods where previously seen data is stored in a buffer and replayed while the model is being trained on new data. This way the model adapts to new and old data at the same time. A dual-network architecture is one of two networks where one network is responsible for short term adaptation to the current data, while the other network retains knowledge of previous data. CL-SLAM makes use of these techniques and is able to produce good results. However, there is one flaw in the approach which is the fact that there is no size limit to the replay buffer. This is not very feasible since there often is limited memory storage in these vehicles or robots that are performing SLAM.

To address this gap, I will be investigating the impact that having a finite buffer size has on the performance on Continual SLAM. Since the buffer size is limited, there needs to be a sampling strategy employed to replace samples once the buffer is full. I formulate the following research questions. "What is the impact of the replay buffer's size on rehearsal based Continual SLAM?" and "What is the impact of the replay buffer's sampling method on rehearsal based Continual SLAM?"

3 | Related Work

Continual learning. The main challenge of continual learning is preventing the catastrophic forgetting of previous knowledge as new knowledge is acquired[8]. As mentioned in Section 2, there are different approaches to tackle this problem. One of which being identifying task boundaries and making modifications accordingly. Learning without forgetting (LwF)[13] makes predictions using new data using parameters that were trained on old data, then makes updates to the model using a modified loss function such that these predictions that were made are not modified too much. A parameter λ is used to determine how much the model is updated. This is known as the plasticity-stability trade-off[17] which is the dilemma of deciding to what degree should the model update. A stable model performs well on old tasks while a plastic model performs well on recent tasks. An optimal in-between is ideal. Elastic Weight Consolidation (EWC)[11] identifies weights that are important to an earlier task and penalizes updates to those weights. Another technique is knowledge distillation (KD) [10] which transfers knowledge from larger, complex networks to simpler networks that preserve the same output. In literature the original network is referred to as the "teacher" and the target network is the "student"[23]. CLS-ER[2] use KD for continual learning by considering the network that is learning the current task as the student and the network that retains knowledge of previous tasks as the teacher. This is also seen as a dual-network approach where one network serves as the long term memory (LTM) tasked with remembering previous tasks while another servers as the short term memory (STM) tasked with performing well on the current task[22]. These techniques work well in mitigating catastrophic forgetting, but a lot of the mentioned works also use rehearsal in parallel.

Rehearsal based approaches. Rehearsal or replay based approaches are ones that store previously seen data in a buffer and replay this data during training. The network gets to see both new and old data which would result in the mitigation catastrophic forgetting if the replay buffer is proper. Rehearsal was first used in 1992[14] with the goal to speed up learning and the buffer contained the most recent samples. Since then, rehearsal has been used in many continual learning approaches. iCaRL[20] maintains it's replay buffer by extracting features from images, then calculating the mean feature vector for each class. The exemplars are ranked based on how similar their feature vector is to the class mean and samples are added/removed based on this ranking. GEM[15] maintains a buffer of the most recent samples per class but mitigates catastrophic forgetting by constraining parameter updates by comparing the loss of the replay buffer and current sample. A-GEM[5] builds on the work of GEM by making it more computationally and memory efficient while maintaining similar performance. When previous tasks are rehearsed they are usually randomly sampled from the buffer, however ER-MIR[1] samples tasks from the buffer whose loss increase would be maximal with respect to the other tasks. DER[3] applies knowledge distillation and regularization on logits sampled during the optimization trajectory. Since DER doesn't rely on setting task boundaries, reservoir sampling[25] is used. MER[21] combines meta learning algorithms with rehearsal and uses reservoir sampling to populate the replay buffer. OCS[27] argues that some samples could be more informative than others and adds new samples to the replay buffer through minibatch similarity, sample diversity or coreset affinity using the gradients of these classes. These rehearsal sampling strategies will be considered in Section 4.

4 | Experimental Setup

Section 4.1 gives an overview of the CL-SLAM[26] framework to give some context to the results. I also describe the main parameters of the experiment and explain how their output can be interpreted with respect to the two research questions. Section 4.3 talks about the different replay buffer sizes that were considered and Section 4.4 talks about the final sampling strategies that were evaluated. Reasons why other sampling strategies were not employed is also discussed.

4.1 | Continual SLAM Overview

It is important to understand the underlying framework of the CL-SLAM[26] system so that there is valid context for the results. 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. Both networks are trained 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 and a replay buffer. VO estimates of the expert are used to construct a pose graph. To reduce drift, global loop closures are detected and are added to the graph, which is then optimized. Finally, a dense 3D map is created using the depth predicted by the expert and the optimized path.

4.2 | Datasets

The CL-SLAM method is employed on two datasets simulating a varied set of environments. In particular, CL-SLAM is initialized with network weights trained on Cityscapes[6] and online continual learning is performed on sequences from the KITTI odometry benchmark[9].

Cityscapes: The Cityscapes dataset[6] is a large-scale autonomous driving dataset that contains RGB images and vehicle metadata such as velocity. It was recorded in 50 cities in Germany, France, and Switzerland and contains 68300 frames.

KITTI: The KITTI Dataset[9] is a pioneering autonomous driving dataset that was recorded in Karlsruhe, Germany. For continual learning of new domains, the CL-SLAM authors use images and ground truth poses of multiple sequences from the odometry benchmark and combine them with the respective IMU (Inertial Measurement Unit) data from the raw dataset. The experiments are ran on sequences 9 and 10 which have 1590 and 1200 frames respectively.

4.3 | Replay Buffer Size

Identifying the impact of using a finite replay buffer size rather than an infinite is the core purpose of this paper. However, the size of the of the replay buffer also has a large impact in terms of both memory and performance. Therefore large, medium, and small samples need to be considered. Buffer sizes of 100, 250, 500, 1000, 2500, 5000, and 10000 were chosen to be tested.

4.4 | Replay Buffer Sampling Strategy

Since a finite replay buffer is being tested and the data that is being added is in the form of a stream, specific sampling strategies must be employed. Sampling from a data stream is a challenge since the entire distribution of data is unknown and future data of the stream is also unknown. Hence sampling strategies that would be used on a regular dataset such as random sampling or stratified sampling[19] cannot be used. Luckily, prior research has been done in continual learning and different sampling strategies have been explored (see Section 3). However, sampling strategies that work for class incremental learning do not necessarily work for domain incremental learning. The identified sampling strategies will be judged in whether or not they are applicable to the current problem.

4.4.1 | Infinite Buffer

The baseline naïve strategy that is used by default in CL-SLAM is having no limit on the replay buffer (R), hence every sample is added (Algorithm 1).

Algorithm 1 Infinite Buffer Algorithm

Input: Replay Buffer R, Number of seen examples N, Selected example x

$R[N] \leftarrow x$
return R

4.4.2 | Most Recent Sampling

Most recent sampling (Algorithm 2) takes the most recently seen samples and stores them in the replay buffer (R). It accomplishes this by adding the current sample (x) that is being streamed to the buffer and simultaneously removing the oldest sample. Since the new samples are being added to the end of the replay buffer, the oldest sample in the buffer would be at the start of the buffer. This strategy is used in GEM[15] and A-GEM[5].

Algorithm 2 Most Recent Algorithm

Input: Replay buffer R, Maximum buffer size B, Number of seen examples N, Selected example x

if $N \leq B$ **then**
    $R[N] \leftarrow x$
else
    $remove(M[1])$
    $R[N] \leftarrow x$
return R

4.4.3 | Reservoir Sampling

Reservoir[25] sampling (Algorithm 3) assigns equal probability to each sample for being represented in the replay buffer (R) and produces the same effect as uniformly sampling from a fixed distribution. It does this by assigning the probability of replacing a sample from the replay buffer with the current sample (x) to $\frac{B}{N}$ where B is the maximum replay buffer size and N is the number of seen examples. As the number of seen examples increases, the probability of replacement deminishes. Sampling and replacement are done at random and no priority is assigned to the samples being added or replaced from the replay buffer i.e. it doesn't matter if the sample being replaced has just been added or has been in the replay buffer from the start. Both DER[3] and MER[21] use reservoir sampling.

Algorithm 3 Reservoir Sampling Algorithm

Input: Replay buffer R, Maximum buffer size B, Number of seen examples N, Selected example x

if $N \leq B$ **then**
    $R[N] \leftarrow x$
else
    $v = randomInteger(min = 0,\ max = N)$
    if $v < B$ **then**
        $R[v] \leftarrow x$
return R

4.4.4 | Reservoir with Forgetting Sampling

Reservoir with forgetting sampling (Algorithm 4) is identical to reservoir sampling (Algorithm 3), however instead of the probability of replacing a sample from the buffer with the currently selected sample increasing as the number of seen examples increases, the probability remains at $\frac{B}{k}$ where B is the maximum buffer size and k is the forgetting coefficient. k can be set to any non-negative integer. If $k < N$ where N is the number of seen examples, then there would be more of a bias to add recent samples to the buffer and older samples would be more likely to be replaced/forgotten, whereas if $k > N$ the opposite would be true and the algorithm would be less inclined to add new samples to the replay buffer. k can be seen as a tunable hyperparameter where the plasticity and stability[17] of the model is modified. In the experiments, $k = 10000$ was tested.

Algorithm 4 Reservoir With Forgetting Sampling Algorithm

Input: Replay buffer R, Maximum buffer size B, Number of seen examples N, Forgetting coefficient k, Selected example x

if $N \leq k$ **then**
    $R[N] \leftarrow x$
else
    $v = randomInteger(min = 0,\ max = N)$
    if $v < k$ **then**
        $R[v] \leftarrow x$
return R

4.4.5 | Sampling Strategies not Considered

The two sampling strategies discussed in Section 3 that were not tested were the ones used by iCaRL[20] and OCS[27]. iCaRL's sampling strategy was attempted, however the attempt was unsuccessful. This is because the feature count of the extracted features in the Cityscapes images did not match the extracted feature count of the KITTI images. This could be due to the fact that the Cityscapes images have different properties compared KITTI images e.g. RGB values. iCaRL's sampling strategy could be used in the scenario where images between environments have the same properties i.e. the same camera is used in each environment. This would be the case in the scenario that a real robot/vehicle is deployed. Sampling strategies discussed in OCS were not used as selecting samples based on the gradient of a classification task can be useful as the differences between gradients of different classes are noticeable and informative, however the difference between gradients of different environments are arbitrary and ambiguous. In the end, only the feasible data stream sampling strategies were selected to be tested and evaluated.

5 | Empirical Evaluation

In this section I present extensive experimental results on the performance of combinations of different replay buffer sample sizes and sampling strategies. Each experiment has been ran 3 times with seeds 4, 42, and 218. The translation error t_{err} (in %) and the rotation error r_{err} (in $°/m$), proposed by the authors of KITTI[9], evaluate the error as a function of the trajectory length.

Baseline Infinite Buffer Size: Firstly, the baseline CL-SLAM[26] model was ran without modifying replay buffer size or replay buffer sampling. The original CL-SLAM results were re-produced so that they could be used as a baseline measure to see if the changes made were an improvement or not. The results are displayed in Table 5.1

Table 5.1: Mean translation and rotation errors for an infinite buffer size

	Error	$c \rightarrow k9$	$c \rightarrow k9 \rightarrow k10$
Infinite	t_{err}	2.88	7.68
	r_{err}	0.44	1.45

Sequence $c \rightarrow k9$: The current sequence's trajectory which is $k9$ is visualized in Figure 5.3. In this set of sequences, we have a diverse set of results ranging from 3.10 to 2.68 for translation error and for 0.46 to 0.38 for rotation error. When comparing the baseline results (Table 5.1) to the results of this sequence (Table 5.2), it is found that 13/21 or 61.9% of the translation errors are better than or equal to baseline and that 18/21 or 85.7% of the rotation errors are better than or equal to baseline. This suggests that overall using a fixed sized replay buffer in general produces results that are on the same level or even better than using an infinitely sized replay buffer.

Table 5.2: Mean translation and rotation errors with sequence $c \rightarrow k9$ for each sampling method. The smallest errors among the combinations are in bold. The smallest errors for each buffer sample size are underlined. The smallest errors for each sampling method are marked with an asterisk (*)

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Most recent	t_{err}	2.87	3.07	2.89	2.82*	2.95	2.87	3.10
	r_{err}	0.42	0.43	0.41	0.40*	0.41	0.41	0.45
Reservoir	t_{err}	**2.68***	2.92	2.79	2.87	3.04	2.80	2.96
	r_{err}	0.40	0.39	0.40	**0.38***	0.46	**0.38***	0.45
Reservoir forgetting	t_{err}	2.93	2.84	3.03	2.81	2.85	2.85	2.80*
	r_{err}	0.43	0.43	0.44	0.39*	0.41	0.40	**0.38**

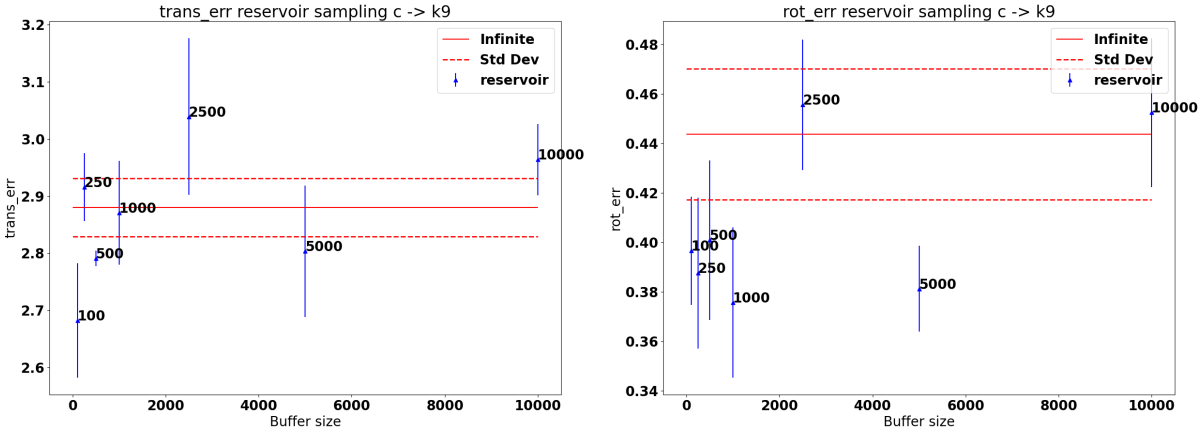

Figure 5.1: Reservoir sampling results with sequence $c \rightarrow k9$ visualized.

Sequence $c \rightarrow k9 \rightarrow k10$: The current sequence's trajectory which is $k10$ is visualized in Figure 5.3. In this set of sequences, we also have a diverse set of results ranging from 11.08 to 4.91 for translation error

and 1.83 to 1.13 for rotation error. When comparing the baseline results (Table 5.1) to the results of this sequence (Table 5.3), it is found that 15/21 or 71.4% of the translation errors are better than or equal to baseline and that 13/21 or 61.9% of the rotation errors are better than or equal to baseline. Similarly to the $c \rightarrow k9$ sequence analysis, we can see that overall using a fixed sized replay buffer in general produces results that are on the same level or even better than using an infinitely sized replay buffer.

Table 5.3: Mean translation and rotation errors with sequence $c \rightarrow k9 \rightarrow k10$ for each sampling method. The smallest errors among the metrics are in bold. The smallest errors for each buffer sample size are underlined. The smallest errors for each sampling method are marked with an asterisk (*)

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Most recent	t_{err}	6.87	7.07	5.30	5.01	**4.91***	6.36	8.39
	r_{err}	1.35	1.48	1.28*	1.39	1.62	1.75	1.46
Reservoir	t_{err}	10.28	8.27	7.03	7.56	6.72	11.08	5.56*
	r_{err}	1.41	1.34	1.34	1.72	**1.13***	1.44	1.39
Reservoir forgetting	t_{err}	9.02	6.95	9.42	5.74*	7.21	7.20	7.59
	r_{err}	1.32	1.27*	1.45	1.56	1.54	1.83	1.38

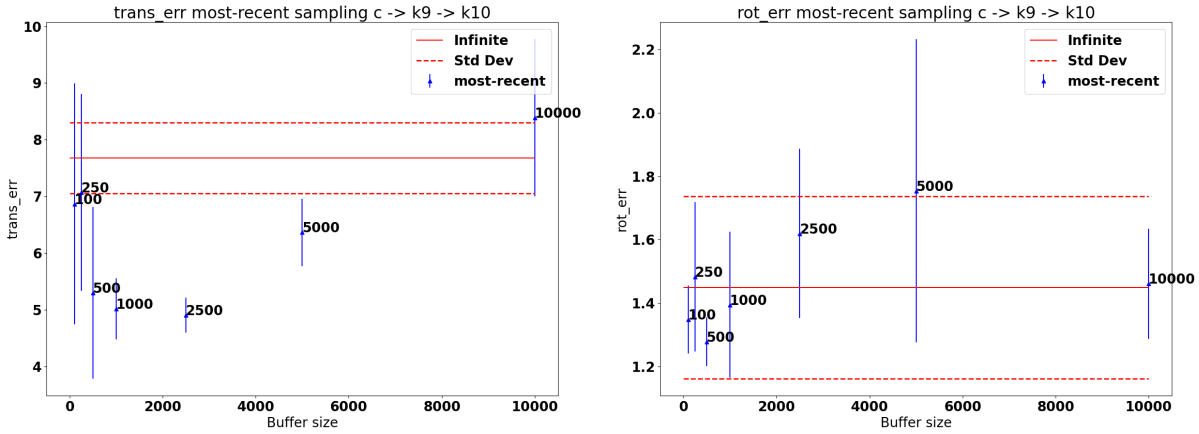

Figure 5.2: Most recent sampling results with sequence $c \rightarrow k9 \rightarrow k10$ visualized.

Replay Buffer Sampling Method: Now that we see that using a finite buffer size overall leads to similar or even better results than using an infinite buffer size, we can evaluate which sampling strategies perform the best among each other. Based on the results in Table 5.2 and Table 5.3, the number of underlined metrics which represent the smallest error for each buffer sample size can be used to determine the sampling method which performed the best. Although it is likely that the performance sampling method does not solely depend on the sampling method but rather a combination of both buffer sample size and sampling method, it is still insightful to see which sampling method performed the best overall. When counting up the underlined metrics for each sampling method, it is found that most recent sampling has 8 underlined metrics, reservoir sampling has 11 underlined metrics and reservoir with forgetting sampling has 10. Reservoir and reservoir with forgetting sampling perform very similarly but reservoir performs slightly better. Also, if looking at the metrics that are in bold which represent the the best results in Table 5.2, it is seen that reservoir sampling has a the lowest translation error and the lowest rotation error and also in Table 5.3 it has the lowest rotation error.

Replay Buffer Size: As seen in Figure5.1 and Figure 5.2 buffer sice can have a big impact on the results. Using the same logic that was used when evaluating the replay buffer sampling method, we can look at Table 5.2 and Table 5.3 and observe the number of metrics that have an asterisk (*) beside them and determine which replay buffer size gives the best results overall. It is found that buffer sizes of 100, 250, 500, and 5000 all have 1 asterisk. Buffer sizes of 2500 and 10000 both have two metrics that have an asterisk. The buffer size of 1000 has 5 metrics that have an asterisk. This suggests that a buffer sample size in the range of 1000 samples produces the most consistent results. When looking at the bold results in Table 5.2, there isn't any correlation with sample size and the best results, though Table 5.3 shows

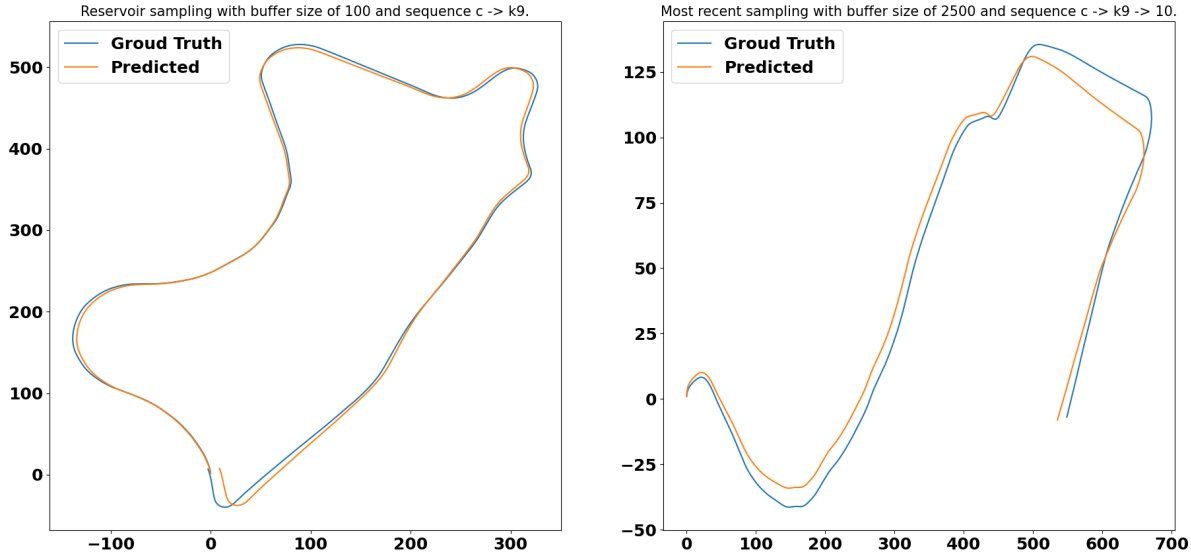

Figure 5.3: Trajectory of sequences $c \rightarrow k9$ and $c \rightarrow k9 \rightarrow k10$

that the best translation and rotation error are both sapling methods that have a buffer size of 2500. It appears that sequence $c \rightarrow k9$ performs the best with a buffer size of 1000 and sequence $c \rightarrow k9 \rightarrow k10$ performs the best with buffer size of 2500.

6 | Conclusion

In this paper I have addressed the gap of the infinite buffer size that was present in the CL-SLAM[26] framework. Based on the results it was shown that not only using a finite replay buffer is a valid strategy, but in many cases it can be an improvement. The two research questions asked have been answered and based on the evidence in Section 5 it seems that Reservoir[25] sampling is the most optimal sampling method for a finite buffer and that a sample size of 1000 or 2500 are the optimal buffer sampling sizes for this problem, however a buffer size of 1000 performed the best overall. The replay buffer sizes can be seen as tunable hyperparameters so if performance is key then the buffer size should be fine tuned as it has a big impact on the final result. This work can still be expanded by testing the iCaRL[20] extracted feature based sampling or any other method that extracts image features and samples based on the image characteristics. As mentioned in Section 4, this could be done in the case where datasets whose images have matching extracted feature sizes.

7 | References

[1] Rahaf Aljundi, Eugene Belilovsky, Tinne Tuytelaars, Laurent Charlin, Massimo Caccia, Min Lin, and Lucas Page-Caccia. Online continual learning with maximal interfered retrieval. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11849–11860. Curran Associates, Inc., 2019.

[2] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

[3] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[5] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[7] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022.

[8] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

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

[11] 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. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[12] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291, 2018.

[13] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

[14] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

[15] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[16] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1), 2022.

[17] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects, 2013.

[18] 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.

[19] Jerzy Neyman. *On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection.* Springer, 1992.

[20] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[21] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

[22] Mohammad Rostami, Soheil Kolouri, and Praveen K Pilly. Complementary learning for overcoming catastrophic forgetting using experience replay. *arXiv preprint arXiv:1903.04566*, 2019.

[23] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[24] Sebastian Thrun and Michael Montemerlo. The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research*, 25(5-6):403–429, 2006.

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

[26] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning. In *Robotics Research*, pages 19–35. Springer, 2023.

[27] Jaehong Yoon, Divyam Madaan, Eunho Yang, and Sung Ju Hwang. Online coreset selection for rehearsal-based continual learning. *arXiv preprint arXiv:2106.01085*, 2021.

A | Additional results

Absolute trajectory RMSE, relative pose (m) error, and relative pose (degree) error are additionally calculated.

Table A.1: Mean translation, rotation, absolute trajectory RMSE, relative pose (m), and relative pose (degree) errors for an infinite buffer size.

	Error	$c \rightarrow k9$	$c \rightarrow k9 \rightarrow k10$
Infinite	t_{err}	2.88	7.68
	r_{err}	0.44	1.45
	$traj_{RMSE}$	8.82	19.33
	$pose_{err}(m)$	0.04	0.05
	$pose_{err}(deg)$	0.05	0.12

Table A.2: Mean translation, rotation, absolute trajectory RMSE, relative pose (m), and relative pose (degree) errors for most recent sampling with sequence $c \rightarrow k9$

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Most recent	t_{err}	2.87	3.07	2.89	2.82	2.95	2.87	3.10
	r_{err}	0.42	0.43	0.41	0.40	0.41	0.41	0.45
	$traj_{RMSE}$	8.67	9.84	9.63	9.97	10.42	9.58	10.61
	$pose_{err}(m)$	0.04	0.04	0.04	0.04	0.04	0.04	0.04
	$pose_{err}(deg)$	0.05	0.05	0.05	0.05	0.06	0.05	0.06

Table A.3: Mean translation, rotation, absolute trajectory RMSE, relative pose (m), and relative pose (degree) errors for most recent sampling with sequence $c \rightarrow k9 \rightarrow k10$

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Most recent	t_{err}	6.87	7.07	5.30	5.01	4.91	6.36	8.39
	r_{err}	1.35	1.48	1.28	1.39	1.62	1.75	1.46
	$traj_{RMSE}$	16.85	17.35	13.36	13.43	12.83	18.06	18.08
	$pose_{err}(m)$	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	$pose_{err}(deg)$	0.11	0.11	0.11	0.11	0.12	0.12	0.12

Table A.4: Mean translation, rotation, absolute trajectory RMSE, relative pose (m), and relative pose (degree) errors for reservoir sampling with sequence $c \rightarrow k9$.

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Reservoir	t_{err}	2.68	2.92	2.79	2.87	3.04	2.80	2.96
	r_{err}	0.40	0.39	0.40	0.38	0.46	0.38	0.45
	$traj_{RMSE}$	8.86	9.76	9.71	9.79	9.87	9.70	9.75
	$pose_{err}(m)$	0.04	0.04	0.04	0.04	0.04	0.04	0.04
	$pose_{err}(deg)$	0.05	0.05	0.06	0.05	0.05	0.05	0.05

Table A.5: Mean translation, rotation, absolute trajectory RMSE, relative pose (m), and relative pose (degree) errors for reservoir sampling with sequence $c \rightarrow k9 \rightarrow k10$.

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Reservoir	t_{err}	10.28	8.27	7.03	7.56	6.72	11.08	5.56
	r_{err}	1.41	1.34	1.34	1.72	1.13	1.44	1.39
	$traj_{RMSE}$	22.61	18.42	18.09	20.39	17.18	23.82	15.97
	$pose_{err}(m)$	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	$pose_{err}(deg)$	0.12	0.11	0.12	0.13	0.12	0.12	0.12

Table A.6: Mean translation, rotation, absolute trajectory RMSE, relative pose (m), and relative pose (degree) errors for reservoir with forgetting sampling with sequence $c \rightarrow k9$

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Reservoir forgetting	t_{err}	2.93	2.84	3.03	2.81	2.85	2.85	2.80
	r_{err}	0.43	0.43	0.44	0.39	0.41	0.40	0.38
	$traj_{RMSE}$	9.81	8.71	10.35	9.02	9.99	8.99	9.17
	$pose_{err}(m)$	0.04	0.04	0.04	0.04	0.04	0.04	0.04
	$pose_{err}(deg)$	0.05	0.05	0.06	0.05	0.05	0.05	0.05

Table A.7: Mean translation, rotation, absolute trajectory RMSE, relative pose (m), and relative pose (degree) errors for reservoir with forgetting sampling with sequence $c \rightarrow k9 \rightarrow k10$.

	Error	**100**	**250**	**500**	**1000**	**2500**	**5000**	**10000**
Reservoir forgetting	t_{err}	9.02	6.95	9.42	5.74	7.21	7.20	7.59
	r_{err}	1.32	1.27	1.45	1.56	1.54	1.83	1.38
	$traj_{RMSE}$	20.52	17.31	21.72	16.26	17.52	18.41	19.58
	$pose_{err}(m)$	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	$pose_{err}(deg)$	0.11	0.12	0.12	0.12	0.12	0.12	0.12

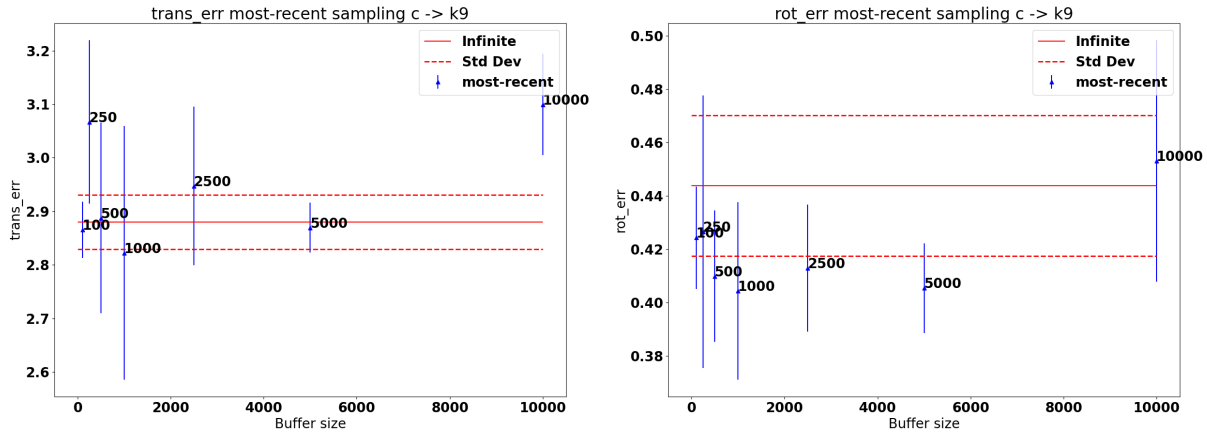

Figure A.1: Most recent sampling results with sequence $c \rightarrow k9$ visualized.

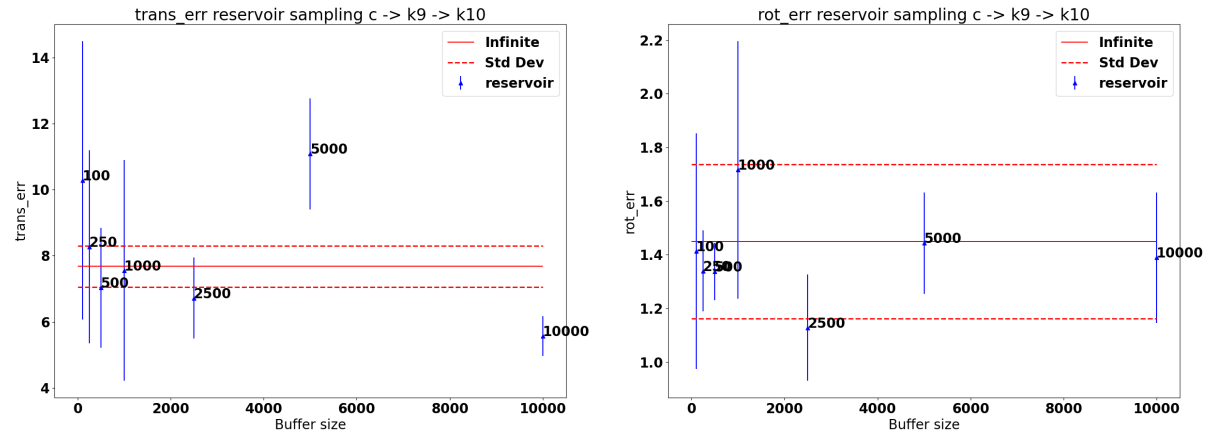

Figure A.2: Reservoir sampling results with sequence $c \rightarrow k9 \rightarrow k10$ visualized.

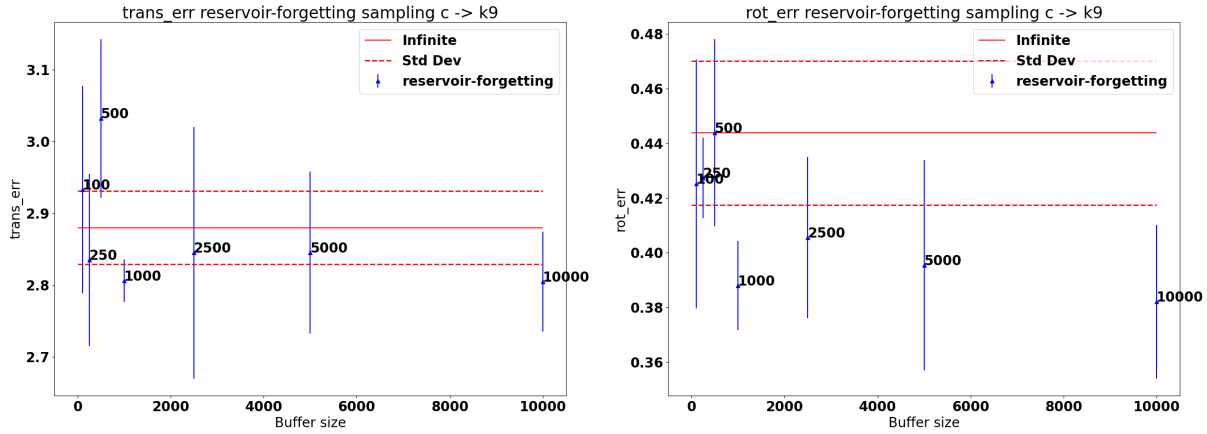

Figure A.3: Reservoir with forgetting sampling results with sequence $c \rightarrow k9$ visualized.

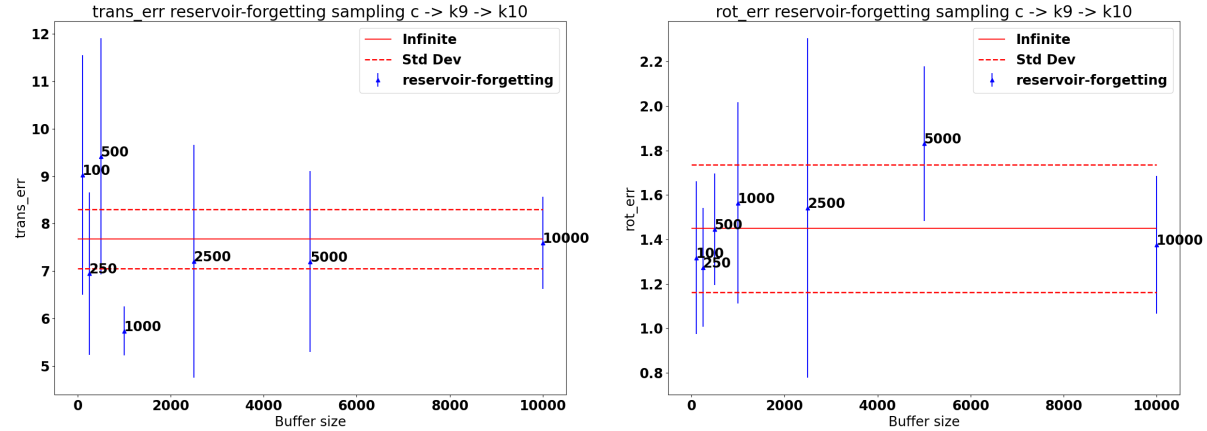

Figure A.4: Reservoir with forgetting sampling results with sequence $c \rightarrow k9 \rightarrow k10$ visualized.