Self-Improving Semantic Perception on a Construction Robot

Hermann Blum, Francesco Milano, René Zurbrügg, Roland Siegwart, Cesar Cadena, and Abel Gawel

***Abstract*— We propose a novel robotic system that can improve its semantic perception during deployment. Contrary to the established approach of learning semantics from large datasets and deploying fixed models, we propose a framework in which semantic models are continuously updated on the robot to adapt to the deployment environments. Our system therefore tightly couples multi-sensor perception and localisation to continuously learn from self-supervised pseudo labels. We study this system in the context of a construction robot registering LiDAR scans of cluttered environments against building models. Our experiments show how the robot's semantic perception improves during deployment and how this translates into improved 3D localisation by filtering the clutter out of the LiDAR scan, even across drastically different environments. We further study the risk of catastrophic forgetting that such a continuous learning setting poses. We find memory replay an effective measure to reduce forgetting and show how the robotic system can improve even when switching between different environments. On average, our system improves by 60% in segmentation and 10% in localisation compared to deployment of a fixed model, and it keeps this improvement up while adapting to further environments.**

I. Introduction

Mobile robots are expected to be deployed in increasingly unstructured environments. While they will have access to information about the environment, such as basic maps or models, advances in learning-based systems enable robots to partially understand the environment through, e.g., object detection or semantic classification [1], [2]. Such understanding is a key requirement to enable many complex, dynamic robotic applications such as autonomous driving or mobile manipulation [3], [4].

Anticipation of a wide variety of environmental conditions is required for safe operation, however difficult if not impossible, and robotic actors with a high degree of autonomy are required to adapt to unexpected and changing conditions for robust operation. Yet, deployment of learning-based systems typically means pre-training a model on a variety of data and then using this static model during deployment. In this work, we explore how learning can be used as a means to self-improve semantic perception during exploration of the environment. Enabling robots to adapt during the job poses three main challenges on robotic systems:

1) Models need to be efficiently (re)trained to incorporate new data (*incremental / continual learning*).
2) Acquired knowledge should be kept while adapting to new tasks and environments (avert *forgetting*).
3) Training signals of the environment are required during deployment, i.e. without manual human supervision in the loop (*self-supervision*).

Authors are with the Autonomous Systems Lab, ETH Zurich.
This work was partially supported by the HILTI group.

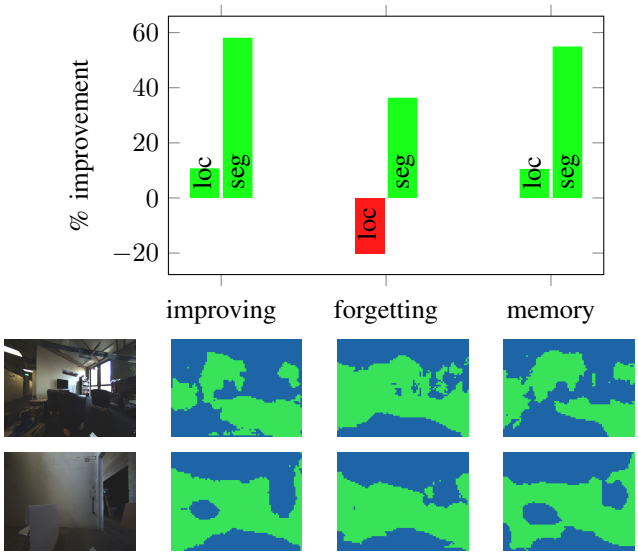


Fig. 1: Our system on average improves semantic segmentation by 60% and median localisation error by 10% during deployment. However, when moving to a different environment type, these improvements can be lost due to *catastrophic forgetting*. We find that memory replay is an effective method against forgetting that can retain large improvements on past and current environments. The graph shows the average relative improvement compared to a fixed pretrained segmentation model. Below are qualitative examples of the segmentation corresponding to the plot above where blue is *foreground* and green is *background*.

The first two points fit particularly with the field of *continual learning* [5], [6], also referred to as *incremental learning* [7], [8], and *lifelong learning* [9], [10], [11]. This learning paradigm deals with the problem of training neural networks in settings where tasks or classes are presented incrementally, or in which the domain – and more generally the data distribution – changes over time [5]. Furthermore, robotic systems generally rely on constrained resources and can only store a limited amount of information, a limitation that continual learning can tackle by reducing the need for big models and datasets fitting all possible deployment observations [6]. In order to continually learn during robot deployment, robots require to generate streams of training signals without humans in the loop. We refer to this as *self-supervised pseudolabeling*, where first approaches in this domain leverage multi-sensor [12] or multi-task systems [13] to transfer labels between modalities and tasks respectively.

We particularly study the use-case of deploying robots

in environments with existing 3D floorplan maps in which the robot is required to localize. This application case is prevalent in building construction and service robotics as increasingly digital 3D floorplan maps are available in these environments. In contrast to the map data that usually represents the static building structure, most relevant for robotic localization (we will refer to this as *background*), actual environments contain large amounts of un-modelled objects or clutter (we will refer to this as *foreground*), potentially obstructing localization performance. Such robotic applications are particularly interesting for self-supervised systems because there is often not much domain-specific data available [3].

Key to our proposed system is the combination of continual learning and self-supervision. A continually learned segmentation model serves as an input filter to the localisation, segmenting the scene into *foreground* and *background*. Based on the localisation in the 3D floorplan, we then harvest self-supervised pseudo-labels for the continual learning of the segmentation. This coupling generates a feedback loop. The robot localises to generate training data for background-foreground segmentation and segments to localise, resulting in improvement of both segmentation and localisation during deployment (see Figure 1). We thus enable online life-long self-supervised learning of semantic scene understanding.

II. Related Work

The idea of self-improving, learning robotic agents has been explored before. One framework in which agents are self-improving is reinforcement learning (RL). With RL, robots have been learning to walk [14], grasp objects [15], or fly [16]. All these systems indeed learn by self-improving over time, often failing in the beginning of the learning process. Usually these learned models are fixed once they acquired the necessary skills, instead of life-long learning. This is because they require supervision signals, e.g. from simulators that are not available in deployment. However, online adaptation of model-based RL has been shown for example in [17].
Outside of RL, [18] and [19] describe online parameter optimisations for model predictive control. The adaptive stereo vision of Tonioni2019-tj has been a particular inspiration for this work. Very related is also sofman2006improving, who learn a probabilistic model for terrain traversability in an online and self-supervised fashion.

A large range of works explored different variants of self-supervision to learn useful image features in convolutional neural networks (CNNs). These techniques include learning to (re)color images [20], to (un)rotate images [21], or to relatively position random crops [22]. However, supervision is always required to relate the learned features to any meaning. In mobile agents, egomotion was found to be a promising, cheaply available self-supervision signal for a range of tasks [23]. Photometric consistency between video frames is used to jointly learn camera calibration, visual odometry, depth estimation, and optical flow [13].
A different line of works produces pseudolabels for segmentation by leveraging models trained on more available data.Segmentation predictions were refined by optimizing over Mask R-CNN predictions, room layout estimation, and superpixels [24], by tracking and optical flow [25], or by aggregating predictions in 3D space and projecting them back onto images sun20203d.While there is no direct prior work for background-foreground-segmentation, our proposed method builds up on similar ideas to use observable characteristics of the environment to produce a learning signal for the target task, which in our case is image segmentation.

In recent years, an increasing number of works [26], [27], [28], [29], [5] investigated the problem of continually learning models while the domain or the classes in the training data vary [6]. The main objective of continual learning is to optimize for the performance on each task or domain with which the network is presented at any given time, while achieving positive knowledge transfer between the tasks, and preventing performance on the previous tasks from decreasing. This decrease on previous tasks is commonly referred to as *catastrophic* forgetting [30], [5]. Multiple techniques have been proposed to tackle catastrophic forgetting, ranging from architectural modifications [31], [29] to regularization techniques [26], [27] and methods based on memories [28], [32] or generative models [33], [34]. Evaluations of these techniques on complex robotic perception tasks often focus on class-incremental learning [35], [36]. In our work, we supplement the training data at each new environment through the use of a memory that keeps a limited number of samples from the previous environment, a method referred to as *replay buffers* [37], [38]. However, we also evaluate the use of regularization techniques often employed in the literature [26], and show the advantage of using memory-based approaches in our scenario, under different replay regimes.

III. Proposed System

We propose a self-improving perception system that interlinks localisation within a map and semantic segmentation of the scene. Importantly, we define the semantics of the scene not as arbitrary class labels, but by the observable affordance that some parts of the scene are mapped (*background*) and some are not (*foreground*). Therefore, we create pseudolabels based on the localisation in the map to train the semantic segmentation, whereas the segmentation into *foreground* and *background* informs the localisation. This creates a feedback loop that can yield improvements in both parts, as can be seen in Figure 2.

A. Semantically Informed Localisation

We localize the robot based on aligning 3D LiDAR scans with the given floorplan in the form of a 3D mesh. Given the building model mesh M, a pointcloud of the LiDAR scan P, and an initial alignment $T^{(t=0)}_{\text{mesh} \rightarrow \text{lidar}}$, we find subsequent robot poses as

$$T^{(t)}_{\text{mesh} \rightarrow \text{lidar}} = \text{ICP}(M, P^{(t)}, T^{(t-1)}_{\text{mesh} \rightarrow \text{lidar}})$$

We use point-to-plane ICP [39] and filter out points of large distance and other criteria. Specific parameters for our experiments are reported in Appendix F.

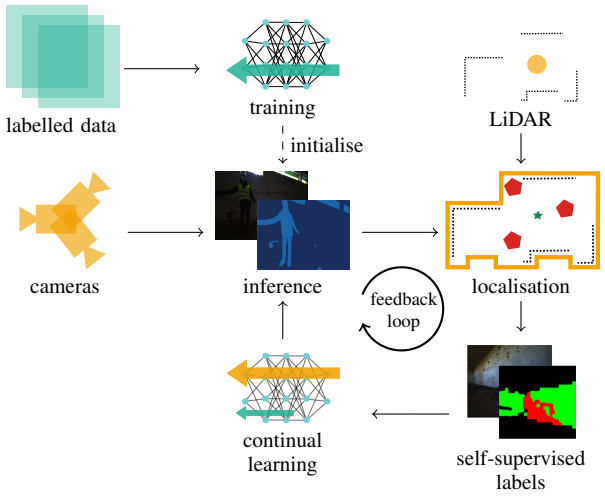
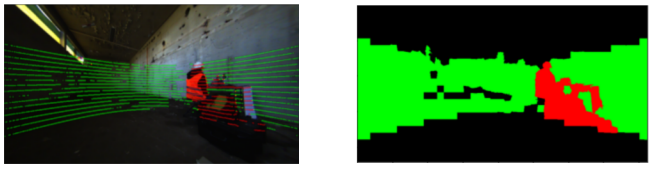


Fig. 2: Overview of the proposed self-improving system. While the perception still incorporates existing training data to form a good prior, it is not fixed during deployment, as would be the established approach. Instead, our semantic segmentation is updated during deployment based on continual learning methods. The necessary training signal is generated from self-supervised pseudolabels, which are therefore available during deployment without manual labelling. We mark signals from the deployment environment in orange and from the pretraining domain in green.



Projected Pointcloud Pseudo Labels

Fig. 3: The pseudo labeling approach. Left are the sparse projected labels after thresholding for a given distance threshold δ. Points colored in red are assigned the foreground class, whereas points colored in green are assigned the background class. The final Pseudo labels on the right are accumulated using superpixels and majority voting.

To further divide the scan P into *foreground* and *background* points, we use additional information from a camera system mounted on top of the LiDAR. Once camera images are semantically segmented, we filter $P_{\text{static}} \subseteq P$ as those points $p \in P$ whose reprojected pixel in image frame is segmented as *background* and localise with $\text{ICP}(M, P^{(t)}_{\text{static}}, T^{(t-1)}_{\text{mesh} \rightarrow \text{lidar}})$.

B. Pseudolabel Generation

We generate pseudo labels for each camera by labeling the captured LiDAR pointcloud leveraging a current pose estimate as well as an architectural mesh of the building. Our labels contain foreground, background and unknown classes and are created in two steps. First, for each point of the localised LiDAR scan, we calculate the distance to the closest plane of the mesh using fast intersection and distance computation [40]. We then check if the distance surpasses a given threshold δ. If so, the point is assigned the *foreground* class, otherwise the *background* class. In the second stage, we project each point onto the respective camera frames and refine the projection using superpixels created with the SLIC [41] algorithm, which utilizes k-means clustering. In particular, we first oversegment the image into a superpixel set S. A superpixel $s \in S$ is then assigned a class according to a majority voting of the contained projected labels. We further improve the segmentation by discarding superpixels whose depth variance surpasses a given threshold. An overview of the approach is depicted in Fig. 3.

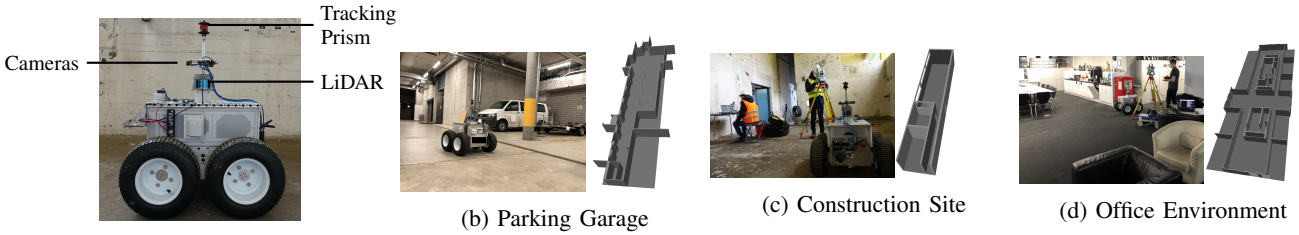
C. Domain Adaptation with Continual Learning

To solve the task of background-foreground segmentation, we incrementally train a neural network architecture on different data sources. To cater to the goal of online learning, we use a lightweight architecture based on Fast-SCNN [42]. We pre-train the network on the NYU-Depth v2 dataset [43], which contains 1449 images extracted from video sequences of indoor scenes, each with per-pixel semantic annotations. We map the classes *wall*, *ceiling*, and *floor* to background and regard everything else as foreground. We perform this initial (pre-)training step to allow the model to acquire prior knowledge.The network is then fine-tuned with self-supervision through the pseudolabels generated on the real-world scene in which the robot is deployed. With reference to the nomenclature often used in continual learning [6], we consider each new environment a *task*, and we assume *task boundaries* to be known. Every time the robot is moved to a new environment, the same scheme as above is applied, i.e., the network trained on the previous environments is provided as new training data the pseudolabels generated from the current environment.

To tackle catastrophic forgetting of information learned on the previous tasks, we adopt a method based on memory replay buffers. When adapting to a new environment, each training batch is filled with the frames collected in the current environment, along with a small fraction of images collected in the previous environments. Therefore, in each training step, the model has to jointly optimize over current and previous environments. However, storing all observations from past environments in memory would come at huge costs. Instead, a memory buffer for each previous environments only contains a random subset of all image and (pseudo-)label pairs. Training batches are then filled from the memory buffers of previous environments alongside self-supervised labels of the current environment. We set the replay fraction to 10%. We evaluate other replay approaches and approaches based on regularisation in Section B.

IV. Experimental Evaluation

We test and verify the applicability of the proposed framework in different steps of increasing complexity. We first validate that our robot can self-improve by deploying it into different unknown environments and measure the gained improvement. We then evaluate the effects of forgetting



(a) Wheeled Robotic Platform (b) Parking Garage (c) Construction Site (d) Office Environment

Fig. 4: Overview of our experimental setup.

and knowledge transfer when switching deployment between different environments. Finally, we conduct an experiment in which the robot learns online during the mission.

For each experiment, we measure the localisation error in the x-y plane[1] in mean, median and standard deviation. We also measure the segmentation quality in mean Intersection over Union (mIoU). The detailed experimental setup is described in Section A.

A. Deployment in a new environment

To test the effectiveness of the pseudolabel training and the localisation based on filtered pointclouds, we deploy the robot in a single new environment. There, the robot collects information in the form of pseudolabels that are used to train the segmentation. Afterwards, we deploy the robot over a different trajectory in the same environment and measure the performance of both image segmentation and localisation, comparing to unfiltered ICP and filtering with a network trained solely on the available labelled NYU data. The results are shown in Table I.We note that segmentation in general is important for good localisation and can prevent failure, as on the construction site. The results also confirm the effectiveness of the pseudolabels, as training on these yields improvements in both segmentation quality and localisation error in all metrics.

B. Transfer into a second environment

We then evaluate the ability of our system to retain knowledge and still adapt to new domains when moved from a first environment to a second one. In particular, we first train on a *source* environment and afterwards train the same model on a *target* environment. For each of these source-to-target transfers, we evaluate the extent of forgetting on both the pretraining data (NYU) and the source environment, comparing our adopted method based on a replay buffer with simple fine-tuning. As shown in Table II and summarised in Figure 1, using replay buffers improves the segmentation performance on the previous tasks w.r.t. the case in which no replay is adopted. This is even more prominent when measured on the pseudolabels and measuring forgetting on the NYU data, which we analyse in further detail in Table V in Appendix D. We further observe that training without replay can cause localisation failure on the source environment, as observed for Construction→Garage and Construction→Office. Table II also highlights cases in which deployment in two consecutive environments in general appears advantageous compared to the experiments in Section IV-A, both with and without memory replay. This indicates a general effect of positive knowledge transfer between our evaluation environments. For a self-improving robotic system this is a promising finding, as it suggests that the robot generally can become better at its task with every new deployment.

[1]The ground truth from the total station only provides translation measurements and does not enable evaluation of correctly estimated orientation.

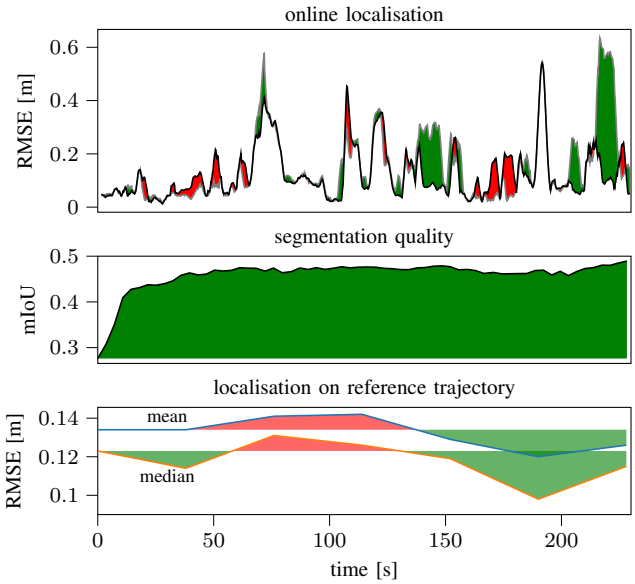


Fig. 5: Online Learning on the construction site. The first row shows the localisation error while the robot drives and learns. The second row shows the evolution of the segmentation quality measured on the ground-truth of the same environment for snapshots of the model at the given time. The third row shows the corresponding localisation error using these snapshots on a different trajectory. Areas in green/red show changes (better/worse) with respect to no online learning. We observe that segmentation quality increases over time and localisation error decreases.

C. Online Learning

We now evaluate how the system can learn online during a mission by learning from the pseudolabels directly as they are generated. For each forward pass, we therefore compile batches of (i) the current camera images that need to be segmented to perform localisation, (ii) 10 images randomly

environment	mean/median/std translation error [mm]			segmentation quality [% mIoU]	
	no segmentation	trained on NYU	self-improving	trained on NYU	self-improving
Garage	50 / 41 / 37	58 / 41 / 73	**43 / 35 / 31**	33.9	**62.8**
Construction	488 / 183 / 999*	126 / 78 / 129	**104 / 68 / 105**	27.6	**48.2**
Office	167 / 168 / 88	196 / 145 / 202	**150 / 138 / 81**	46.5	**53.9**

TABLE I: Comparison of localisation and segmentation quality when using no segmentation, deploying a fixed segmentation model trained on the NYU dataset, and our self-improving approach (including replay). Localisation errors marked with * contain a major ICP failure.

environment source → target	method	mean/median/std translation error [mm] source	 target	segmentation [% mIoU] source	 target
Garage→ Construction	replay	41 / 33 / 30	98 / 66 / 98	**60.8**	48.6
	finetuning	40 / 33 / 29	87 / 67 / 77	55.1	49.4
Garage→ Office	replay	39 / 31 / 31	168 / 137 / 109	**62.6**	47.2
	finetuning	37 / 30 / 33	196 / 118 / 267	61.0	47.4
Construction→ Garage	replay	**105 / 68 / 108**	40 / 33 / 29	**49.3**	62.2
	finetuning	549 / 84 / 1500*	40 / 31 / 29	42.3	62.0
Construction→ Office	replay	**125 / 72 / 128**	158 / 137 / 93	**50.3**	47.6
	finetuning	514 / 191 / 914*	146 / 123 / 93	45.4	49.4
Office → Garage	replay	**153 / 131 / 95**	44 / 34 / 36	**47.8**	62.1
	finetuning	182 / 151 / 114	38 / 32 / 27	40.2	61.0
Office → Construction	replay	171 / 161 / **86**	91 / 66 / 85	**47.5**	49.9
	finetuning	**168 / 159** / 88	121 / 70 / 128	33.3	49.1

TABLE II: Evaluation of forgetting and knowledge transfer when switching between deployment environments. The perception system is first trained on pseudolabels of the source environment, then on the target, and then evaluated on both. Bold marks cases where one method reduces forgetting compared to the other method. Underlined metrics are better than single-environment deployment from Table I. Finetuning leads to two cases where forgetting causes ICP failures, which are marked with star.

sampled from a buffer A where we already have associated pseudolabels, and (iii) 1 image from a buffer B of NYU images for memory replay. After the forward pass, we extract the prediction of (i) and backpropagate based on the loss from (ii) and (iii). After localising the current scan, we generate pseudolabels for (i) and fill them into buffer A. Our evaluation of online learning is shown in Figure 5. We observe that over time segmentation quality increases and localisation error decreases. Notably, already a few seconds of online learning increase the segmentation quality significantly. However, it takes longer until we measure a notable effect on the localisation. Due to the limited time until the end of the trajectory, we are therefore unable to see if there is a feedback effect where the improved localisation would create better pseudolabels that can in turn increase the segmentation quality further.

V. Conclusion & Outlook

In this work we propose a framework for self-improving semantic perception by combining continual learning with self-supervision. We study this on a robotic system that localises in 3D floorplan meshes. Our experiments validate the gains of the self-improving systems in diverse environments. In particular, we analyse the effects of knowledge transfer and forgetting when switchting between environments. We find that memory replay is an effective solution that can mitigate the effects of forgetting, and observe that exposure to multiple environments is sometimes even beneficial for overall performance.

The concept of self-improving, continual, online learning robots opens up exciting questions for future research. The related self-supervision approaches that we describe may facilitate the transfer of our proposed framework to other robotic applications. Moreover, our evaluations of knowledge transfer and forgetting show potential for effective combinations of self-supervision and continual learning. Finally, we identify long-term deployment of online learning systems as an important future research direction for self-improving perception.

References

[1] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric Object-Level SLAM," in *Intl. Conf. on 3D Vision (3DV)*. IEEE, 2018, pp. 32–41.

[2] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1689–1696.

[3] L. Kunze, N. Hawes, T. Duckett, M. Hanheide, and T. Krajník, "Artificial intelligence for Long-Term Robot Autonomy: A Survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4023–4030, 2018.

[4] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[5] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2020.

[6] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Díaz-Rodríguez, "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges," *Information Fusion*, vol. 58, pp. 52–68, 2020.

[7] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta, "Incremental Robot Learning of New Objects with Fixed Update Time," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2017.

[8] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental Learning of Object Detectors without Catastrophic Forgetting," in *Intl. Conf. on Computer Vision (ICCV)*, 2017.
[9] Z. Chen and B. Liu, "Lifelong Machine Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.
[10] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and Autonomous Systems*, vol. 15, pp. 25–46, 1995.
[11] D. L. Silver, Q. Yang, and L. Li, "Lifelong Machine Learning Systems: Beyond Learning Algorithms," in *AAAI Conf. on Artificial Intelligence (AAAI)*, 2013.
[12] W. Sun, J. Zhang, and N. Barnes, "3D Guided Weakly Supervised Semantic Segmentation," *CoRR*, vol. abs/2012.00242, 2020.
[13] Y. Chen, C. Schmid, and C. Sminchisescu, "Self-Supervised Learning With Geometric Constraints in Monocular Video: Connecting Flow, Depth, and Camera," in *Intl. Conf. on Computer Vision (ICCV)*, 2019, pp. 7063–7072.
[14] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to Walk via Deep Reinforcement Learning," *arXiv preprint arXiv:1812.11103*, 2018.
[15] M. Q. Mohammed, K. L. Chung, and C. S. Chyi, "Review of Deep Reinforcement Learning-Based Object Grasping: Techniques, Open Challenges, and Recommendations," *IEEE Access*, vol. 8, pp. 178 450–178 481, 2020.
[16] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a Quadrotor With Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.
[17] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4019–4026.
[18] M. Lorenzen, M. Cannon, and F. Allgöwer, "Robust MPC with recursive model update," *Automatica*, vol. 103, pp. 461–471, 2019.
[19] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-Based Model Predictive Control for Autonomous Racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.
[20] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," in *European Conf. on Computer Vision (ECCV)*. Springer, 2016, pp. 649–666.
[21] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *Intl. Conf. on Learning Representations (ICLR)*, 2018.
[22] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised Visual Representation Learning by Context Prediction," in *Intl. Conf. on Computer Vision (ICCV)*, 2015, pp. 1422–1430.
[23] P. Agrawal, J. Carreira, and J. Malik, "Learning to See by Moving," in *Intl. Conf. on Computer Vision (ICCV)*, 2015, pp. 37–45.
[24] M. A. Reza, A. U. Naik, K. Chen, and D. J. Crandall, "Automatic Annotation for Semantic Segmentation in Indoor Scenes," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4970–4976.
[25] L. Porzi, M. Hofinger, I. Ruiz, J. Serrat, S. R. Bulo, and P. Kontschieder, "Learning Multi-Object Tracking and Segmentation From Automatic Annotations," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[26] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
[27] Z. Li and D. Hoiem, "Learning without Forgetting," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 40, no. 12, pp. 2935–2947, 2018.
[28] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental Classifier and Representation Learning," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[29] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & Compress: A scalable framework for continual learning," in *Intl. Conf. on Machine Learning (ICML)*, 2018.
[30] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.
[31] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," *arXiv preprint arXiv:1606.04671*, 2016.
[32] D. Lopez-Paz and M. Ranzato, "Gradient Episodic Memory for Continual Learning," in *Conf. on Neural Information Processing Systems (NIPS)*, 2017.
[33] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual Learning with Deep Generative Replay," in *Conf. on Neural Information Processing Systems (NIPS)*, 2017.
[34] C. Wu, L. Herranz, X. Liu, Y. Wang, J. v. d. Weijer, and B. Raducanu, "Memory Replay GANs: Learning to Generate Images from New Categories without Forgetting," in *Conf. on Neural Information Processing Systems (NIPS)*, 2018.
[35] A. Douillard, Y. Chen, A. Dapogny, and M. Cord, "PLOP: Learning without Forgetting for Continual Semantic Segmentation," *CoRR abs/2011.11390*, 2020.
[36] L. Yu, X. Liu, and J. van de Weijer, "Self-Training for Class-Incremental Semantic Segmentation," *CoRR abs/2012.03362*, 2020.
[37] T. L. Hayes, N. D. Cahill, and C. Kanan, "Memory Efficient Experience Replay for Streaming Learning," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019.
[38] S. Zhang and R. S. Sutton, "A Deeper Look at Experience Replay," in *Conf. on Neural Information Processing Systems (NIPS) - Deep Reinforcement Learning Symposium*, 2017.
[39] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.
[40] P. Alliez, S. Tayeb, and C. Wormser, "3D fast intersection and distance computation," in *CGAL User and Reference Manual*, 5.2 ed. CGAL Editorial Board, 2020. [Online]. Available: https://doc.cgal.org/5.2/Manual/packages.html#PkgAABBTree
[41] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 34, no. 11, pp. 2274–2282, 2012.
[42] R. P. K. Poudel, S. Liwicki, and R. Cipolla, "Fast-SCNN: Fast Semantic Segmentation Network," in *British Machine Vision Conf. (BMVC)*, 2019.
[43] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *European Conf. on Computer Vision (ECCV)*, 2012.
[44] U. Michieli and P. Zanuttigh, "Incremental Learning Techniques for Semantic Segmentation," in *Intl. Conf. on Computer Vision (ICCV), Workshop on Transferring and Adapting Source Knowledge in Computer Vision (TASK-CV)*, 2019.
[45] ——, "Knowledge distillation for incremental learning in semantic segmentation," *Computer Vision and Image Understanding*, vol. 205, p. 103167, 2021.
[46] Y. Wu and K. He, "Group Normalization," in *European Conf. on Computer Vision (ECCV)*, 2018.
[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Intl. Conf. on Machine Learning (ICML)*, 2015.
[48] R. Giraud, V. Ta, and N. Papadakis, "SCALP: superpixels with contour adherence using linear path," *CoRR*, vol. abs/1903.07149, 2019. [Online]. Available: http://arxiv.org/abs/1903.07149

APPENDIX

A. Experimental Setup

1) Robotic System: We conduct all our experiments on an open-source wheeled robotic ground vehicle[2] shown in Figure 4a. To calibrate the sensor system, we calibrate all cameras to an IMU that we attach to the sensor system and then align trajectories from visual-inertial odometry and LiDAR to find the extrinsic calibration between the LiDAR and the cameras. The tracking prism is used to gather ground-truth with a total station for our evaluation and is mounted on the sensor system to be aligned with the optical center of the LiDAR. To improve time synchronisation between the different sensors, we use hardware trigger boards to trigger all cameras simultaneously. From images captured at 20 Hz

[2]https://github.com/ethz-asl/eth_supermegabot

we then take those closest to the timestamp of the LiDAR scans, which are captured at 5Hz. All sensor drivers run on the same computer where sensors are synchronised to system time. While we also ran time-synchronisation to the same computer from the external total station tracking which supplies ground-truth poses in our evaluations, we found that this synchronisation did not work sufficiently accurate. We therefore manually correct time-offsets based on trajectory alignment between tracked prism and localised robot.

2) Evaluation Environments: We deploy our proposed system into three different environments: a construction site (`Construction`), a parking garage (`Garage`), and an office floor (`Office`). Architectural researchers provided us 3D meshes, constructed from dense 3D scans (`Construction`) and existing 2D floorplans supplemented with additional measurements (`Garage` and `Office`). Figure 4 shows these meshes together with the experimental setup. In each environment, we steer the robot through multiple independent trajectories of 2-3 min while tracking the robot with a total station. The coordinate system of the total station is always initialised to the origin of the building mesh, which we therefore set at corners visible to the total station. To evaluate the learned segmentation models, we sample images from each environment and manually annotate them with segmentation masks. We labelled 30 images from the garage that have in total 65% background pixels, 26 images from the construction site that have 76% background pixels and 31 images from the office that have 40% background pixels. When not specified differently, we evaluate only the region of the images that can also be reprojected to the LiDAR, applying a static field-of-view (FoV) mask.[3]

3) Learning Setup: We use a lightweight architecture based on Fast-SCNN [42] that we train for background-foreground segmentation. We resize input images from both the pre-training dataset and from the cameras to a common size (480×640 pixels). We first train the model on the NYU dataset. Then, when we deploy the robot into a new environment (cf. Sec. IV-A), we fill a replay buffer with samples from NYU in addition to training on the pseudolabels from the current environment. When we evaluate the transfer from a first environment to a second one (cf. Sec. IV-B), we replay images from both NYU and the first environment. We set the replay fraction to 10%, but we evaluate different replay regimes and strategies in Section B. Before feeding images to the network, we augment them with left-right flipping and random perturbation of brightness and hue. See Appendix C for all training parameters.

In each experiment, we hold out 10% of the training samples for validation and train our model using Adam optimizer; we set the learning rate to 10^{-4} for the pre-training on NYU and to 10^{-5} for the remaining experiments, and adaptively decrease it when the validation loss reaches a plateau. We optimize the cross-entropy loss on the binary foreground-background labels. When training on pseudo-labels produced through the method detailed in Sec. III-B, we only apply the loss to those pixels that contain one of the two classes.

[3]The FoV mask has two reasons: Because we are primarily interested in good localisation, comparing segmentation quality in the region that can be reprojected to the LiDAR relates better to localisation performance. Additionally, pseudolabels are also only available in image regions where the LiDAR provides information, therefore we expect the segmentation to learn mostly the semantics of the scene visible in these regions.

B. Ablation Studies

We investigate two details in ablation studies to assess how the use of different continual learning methods – and in particular different replay schemes – impacts the segmentation accuracy in our domain adaptation setting.

We first evaluate two different strategies for memory replay. In the first strategy, which is the one that we adopt in the main experiments, on each source-to-target experiment (e.g., NYU$\rightarrow$`Garage`), we fill the replay buffer with a fraction of samples from the source dataset(s) (NYU in the example), which we select randomly. We then fill training batches from the replay buffer and target dataset according to their relative sizes. In the second strategy, we fill the replay buffer with the full source dataset but fill training batches with a pre-defined target-source ratio. For instance, a ratio `Garage` : NYU $= 4 : 1$ with a batch size of 10 indicates that batches on average contain 8 images from Garage and 2 images from NYU. As shown in Table III, milder replay regimes (larger target-source ratios, or smaller replay fractions) achieve higher performance on the target domain, but cause the amount of information retained from the source domain to drop. This forgetting phenomenon is particularly evident in the adaptation tasks in which the semantic gap between source and domain is larger. Indeed, for instance, in the NYU$\rightarrow$`Garage` experiment we observe a drop of 31.9% in mIoU on the NYU labels between a replay strategy with a fraction of 10% and simple fine-tuning, while the same decrease in performance when the target domain is `Office` – semantically closer to the indoor dataset NYU – is 14.8%. At the same time, the segmentation quality on the pseudo-labels of the target dataset follows an inverse trend, generally increasing for smaller amounts of replay. This highlights the trade-off between the accuracy on the target and on the source domain.

We further compare regularization techniques from the continual-learning literature. In particular, we investigate the use of a distillation approach, in which a weighted regularization term $\mathcal{L}_d$ is added to the cross entropy loss $\mathcal{L}_{ce}$, to encourage the network to retain the knowledge from previous tasks: $\mathcal{L} = \mathcal{L}_{ce} + \lambda\mathcal{L}_d$. Similarly to [44], we experiment this distillation either on the output logits produced by the network (*Output distillation*) or on the intermediate features extracted from the model architecture before the final classification module (*Feature distillation*). Furthermore, we evaluate Elastic Weight Consolidation (EWC) [26]. The loss optimized in EWC is of a similar form: $\mathcal{L} = \mathcal{L}_{ce} + \lambda\mathcal{L}_{ewc}$, where $\mathcal{L}_{ewc}$ penalizes deviations of the network parameters across tasks. We present further details on the methods and parameters in Appendix E.

As shown in Table III, in our experiments replay buffers prove to be the most effective among the examined methods

in minimizing the amount of forgetting on the NYU dataset, and generally allow attaining a good trade-off with the segmentation quality on the pseudo-labels from the target domain. We also note that, with limited exceptions, both regularization approaches fail to maintain a good performance on the source dataset NYU. We believe that for distillation methods this can be ascribed to the fact that, as opposed to related works that explored similar techniques in a class-incremental setting [44], [45], we conduct domain adaptation. More importantly, unlike [44], [45] our supervision signal does not consist of accurate ground-truth annotations available at all pixels, but of noisy pseudo-labels that often cover a limited region of the image. Finally, while similar considerations also apply for EWC, we believe that the limited effectiveness of this technique in our setting is also related to the method being designed for classification as opposed to image segmentation.

C. Details on the Network Architecture

In all our experiments we use a batch size of 10 and train the network for up to 100 epochs, using early stopping with a patience of 20 epochs based on the validation loss. Our network architecture, based on Fast-SCNN [42], has a total of 1,775,110 trainable parameters. We use group normalization [46] in all layers; we conducted a preliminary ablation study (cf. Table IV) comparing this design choice with the alternative batch normalization [47]. In accordance with [46], we found group normalization to be more indicated for our transfer-learning tasks, in which the statistics of the *source* training data, used by batch normalization to fit per-layer parameters [47], do not match in general those of the *target* domain. This is reflected in the models trained with group normalization performing consistently better or comparably to those trained with batch normalization, as soon as a non-negligible amount of replay is used.

D. Details on Cross-Domain Forgetting

We present a detailed analysis of forgetting in terms of segmentation in Table V as supplementary information to the main results presented in Table II. With no exception, memory replay performs better on source environments than finetuning. We note that the effect of forgetting is even stronger on the NYU data than in the deployment environments.

E. Details on the Continual-Learning Ablation Study

For both distillation and EWC, we use the same learning parameters as the experiments with replay buffers. In the following, we denote with $\mathbf{X}$ and $\mathbf{M}$ respectively an image and the corresponding mask from the training dataset $\mathcal{D}$. When $\mathbf{X}$ is a pseudo-label image, a pixel in $\mathbf{M}$ is *masked* if the corresponding pixel in $\mathbf{X}$ has an associated pseudo-label (background/foreground) and *not masked* if the corresponding pixel has unknown label; if $\mathbf{X}$ is an image replayed from NYU, all pixels in $\mathbf{X}$ are masked. For a given stage-1 experiment (i.e., in which we deploy the model pretrained on NYU in a new environment, cf., e.g., Tab. V), we denote the output prediction of the model pretrained on NYU as $\mathbf{y}_0(\mathbf{X})$ and the output prediction of the current stage-1 model as $\mathbf{y}(\mathbf{X})$; to indicate the predicted score associated to each class $c \in \{b, f\}$ (b = background, f = foreground) we write $\mathbf{y}_0(\mathbf{X})[c]$ and $\mathbf{y}(\mathbf{X})[c]$. Finally, we denote with $M(\mathbf{X}, \mathbf{M})$ a function that maps an input image $\mathbf{X}$ and its corresponding mask $\mathbf{M}$ to a vectorized version of $\mathbf{X}$ that contains only the pixels that are masked in $\mathbf{M}$.

The generic distillation loss reads as follows:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda \mathcal{L}_d, \tag{1}$$

where λ is a hyper-parameter and $\mathcal{L}_{ce}$ is the cross-entropy loss (cf. Sec. B).

For output distillation, the regularization loss $\mathcal{L}_d$ is a cross-entropy loss between the prediction of the previous and the current model, masked by the input mask of each image, i.e.,

$$\mathcal{L}_d = -\sum_{(\mathbf{X},\mathbf{M})\in\mathcal{D}} \sum_{c\in\{b,f\}} \frac{M(\mathbf{y}_0(\mathbf{X}),\mathbf{M})[c] \cdot \log(M(\mathbf{y}(\mathbf{X}),\mathbf{M}))[c]}{|\mathcal{D}|}. \tag{2}$$

For feature distillation, similarly to [44] we consider the features outputted by the network at a selected layer and minimize the ℓ_2 norm between these as returned by the pre-trained model and by the current model. In particular, we consider the layer that precedes the final classification module in the Fast-SCNN architecture [42] and denote its output as $\mathbf{l}_0(\mathbf{X})$ and $\mathbf{l}(\mathbf{X})$, respectively for the pre-trained and for the current model. The regularization loss can therefore be expressed as:

$$\mathcal{L}_d = \frac{\|\mathbf{l}_0(\mathbf{X}) - \mathbf{l}(\mathbf{X})\|_2^2}{|\mathcal{D}|}. \tag{3}$$

For Elastic Weight Consolidation (EWC), we adopt the original loss introduced in [26], which is of the form:

$$\mathcal{L} = \mathcal{L}_{\text{main}} + \lambda \sum_i F_i(\theta_i - \theta_{i,0})^2, \tag{4}$$

where the sum is computed over the trainable parameters θ_i and $\theta_{i,0}$ respectively of the current and of the pre-trained model, and F_i is the element on the diagonal of the Fisher information matrix associated with the i-th parameters. $\mathcal{L}_{\text{main}}$ represents the main loss optimized in the given task, which in our case is the background-foreground cross-entropy loss $\mathcal{L}_{ce}$.

F. Localisation Parameters

In general, we run point-to-plane ICP with 3 nearest neighbors and initialise on the previously solved pose. We apply multiple filters to the input scan, even after the semantic filtering:

- We require the scan to have at minimum 500 points (ie rejecting scans where the segmentation classified nearly everything as foreground).
- We subsample the scan to a maximum density of 10000 pts/m^3.
- After nearest neighbor association, we reject the 20% points that are furthers away from the map.

		NYU→Garage			NYU→Construction			NYU→Office			
		NYU	Garage		NYU	Construction		NYU	Office		
			Pseudo	GT		Pseudo	GT		Pseudo	GT	
Finetuning		36.4	96.3	61.8	36.6	79.5	48.9	66.2	70.9	51.2	
Replay buffer with ratio target : NYU	1 : 1	**87.2**	86.5	61.5	**82.9**	66.6	46.1	**83.5**	57.8	49.9	memory replay
	3 : 1	81.1	91.7	60.7	79.8	73.1	47.8	81.4	68.1	**52.2**	
	4 : 1	79.8	92.4	**62.3**	78.7	73.1	**48.8**	82.0	65.8	51.7	
	10 : 1	73.4	94.7	61.3	75.3	75.6	47.6	77.7	**71.2**	52.1	
	20 : 1	67.5	95.3	62.0	72.4	76.3	48.3	76.0	69.1	52.0	
	200 : 1	53.9	**96.1**	61.6	53.2	**77.2**	48.7	74.8	68.7	50.9	
Replay buffer with fraction replay	10%	**68.3**	95.4	62.8	**78.6**	**77.0**	48.2	**81.0**	**69.7**	**53.9**	
	5%	65.0	**95.9**	62.0	76.2	76.9	**48.5**	79.9	69.3	51.5	
Feature distillation	$\lambda = 0.5$	35.4	**96.1**	**61.9**	33.2	**77.1**	50.3	**65.3**	71.1	**50.7**	regularisation methods
	$\lambda = 1$	34.8	95.9	61.2	30.6	76.9	**50.8**	58.6	**78.9**	49.1	
	$\lambda = 10$	**37.4**	94.2	61.6	**33.6**	72.1	48.3	48.7	72.2	48.0	
	$\lambda = 50$	33.6	92.7	61.1	32.6	63.1	46.5	44.0	64.5	46.8	
Output distillation	$\lambda = 0.5$	33.1	**94.4**	63.3	34.0	**76.5**	**47.8**	**62.9**	**68.4**	**49.9**	
	$\lambda = 1$	32.4	85.3	**64.4**	**38.2**	59.4	46.6	53.0	60.4	44.3	
	$\lambda = 10$	37.8	40.8	47.5	37.9	32.9	37.1	45.3	35.8	36.1	
	$\lambda = 50$	**39.0**	48.9	53.0	31.7	28.7	31.1	46.3	30.5	35.9	
EWC [26]	$\lambda = 0.5$	36.1	**96.4**	**61.5**	34.4	**76.5**	47.9	65.7	69.2	51.6	
	$\lambda = 1$	36.2	96.3	61.4	**37.0**	76.4	**48.0**	66.2	**74.0**	50.8	
	$\lambda = 10$	**37.9**	96.2	61.1	35.4	76.2	**48.0**	**70.6**	69.1	**51.8**	
	$\lambda = 50$	**37.9**	95.9	**61.5**	35.0	75.6	47.9	65.5	73.2	51.0	

TABLE III: Ablation study over different continual learning methods. After adapting from the source domain (NYU) to the different deployment environments, we measure segmentation quality [% mIoU] on the source data as well as the self-supervised pseudolabels (Pseudo) and our ground-truth annotations (GT). We compare the finetuning baselines with memory replay and regularisation methods. We observe that memory replay is more successful than regularisation at preventing catastrophic forgetting in our application.

		NYU		Garage (Pseudo)	
		BN	GN	BN	GN
Ratio target : NYU	1 : 1	84.9	**87.2**	**87.7**	86.5
	3 : 1	77.8	**81.1**	90.6	**91.7**
	4 : 1	76.4	**79.8**	92.0	**92.4**
	10 : 1	70.3	**73.4**	93.6	**94.7**
	20 : 1	66.7	**67.5**	94.5	**95.3**
	200 : 1	**54.6**	53.9	95.3	**96.1**
Fraction replay NYU	10%	67.6	**68.3**	94.0	**95.4**
	5%	63.6	**65.0**	94.9	**95.9**
	0% (fine-tuning)	**37.3**	36.4	95.5	**96.3**

TABLE IV: Comparison of segmentation quality [% mIoU] on NYU→`Garage` between models trained with batch normalization (BN) and models trained with group normalization (GN), under different replay regimes.

- We reject associations where the estimated surface normals (estimated based on the 10 nearest neighbors) have a larger angle deviation than 1.5 rad.

As mentioned, in order to localise without segmentation and generate pseudolabels in the very cluttered office environment, we enforce additional filters:

- We only localise in 4 degrees of freedom (x, y, z, yaw).
- We estimate normal directions based on 30 nearest neighbors and only associate points to the map if the angle between the normals is below 0.8 rad.

G. Pseudolabel Parameters

We empirically set the distance threshold to $\delta = 0.1$m and discard superpixels with a depth variance that surpasses 0.5m. We smooth the images with a Gaussian kernel ($\sigma = 0.2$) and oversegment them into approximately 400 superpixels with SLIC parameter `compactness` $= 10^4$. On the data captured from the garage, we use a different superpixel algorithm (SCALP [48]) that we later discard because of long runtimes. We do not notice qualitative differences between the created superpixels. In the office environment, we increase the standard deviation threshold to 1m due to large amounts of clutter.

To get an estimate of the quality of the pseudolabels themselves, we match frames where we have both manual ground-truth annotations and pseudolabels. Unfortunately, we could not recover pseudolabels for the images that were used to generate ground-truth in the office environment. When evaluating the pseudolabels, we also ignore all pixels that are not labelled (due to high variance or no reprojected LiDAR points in that superpixel). Therefore, the evaluation is strongly biased in favor of the pseudolabels. We measure 68.4% mIoU on the garage pseudolabels. For the same pixels (only those where pseudolabels are not ignored), our trained models get 64.3% mIoU. In the construction site environment, we measure 49.5% mIoU for the pseudolabels and 54.3% mIoU

[4]This procedure is suggested by the skimage implementation that we use.

Stage	Source → target	Segmentation quality [% mIoU]													
		NYU		Garage				Construction				Office			
		GT		Pseudo		GT		Pseudo		GT		Pseudo		GT	
		RB	FT	RB	FT	RB	FT	RB	FT	RB	FT	RB	FT	RB	FT
0	Pretraining on NYU	–	86.4	–	(22.5)	–	(33.9)	–	(22.7)	–	(27.6)	–	(39.6)	–	(46.5)
1	NYU → Garage	**68.3**	36.4	95.4	96.3	62.8	61.8	–	–	–	–	–	–	–	–
1	NYU → Construction	**78.6**	36.6	–	–	–	–	77.0	79.5	48.2	48.9	–	–	–	–
1	NYU → Office	**81.0**	66.2	–	–	–	–	–	–	–	–	69.7	70.9	53.9	51.2
2	Garage→ Construction	**70.3**	30.7	**91.8**	77.1	60.8	55.1	77.4	78.5	48.6	49.4	–	–	–	–
2	Garage→ Office	**70.9**	42.7	**92.8**	71.7	62.6	61.0	–	–	–	–	69.9	72.2	47.2	47.4
2	Construction→ Office	**78.6**	48.9	–	–	–	–	**71.3**	55.9	50.3	45.4	70.3	72.2	47.6	49.4
2	Construction→ Garage	**70.5**	36.7	94.4	95.6	62.2	62.0	**61.4**	43.3	49.3	42.3	–	–	–	–
2	Office → Garage	**68.7**	36.4	95.3	96.4	62.1	61.0	–	–	–	–	**61.2**	46.9	47.8	40.2
2	Office → Construction	**77.7**	38.8	–	–	–	–	73.1	73.0	49.9	49.1	**63.4**	44.7	47.5	33.3

TABLE V: Bold shows how the replay buffer (RB) prevents degradation of performance on the datasets on which the model has previously been trained, as opposed to simple fine-tuning (FT).

for our trained model.

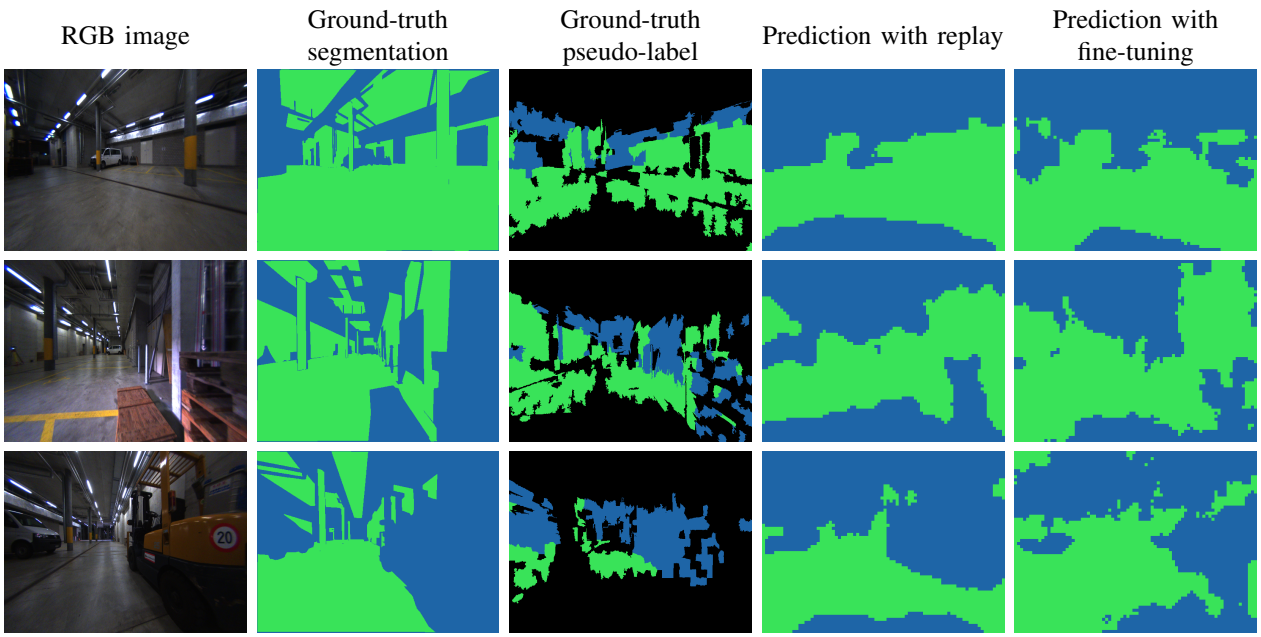
H. Example of segmentation predictions

Figures 6, 7, and 8 show examples of segmentation masks produced by the network on the source environment in the experiments with transfer from a first to a second environment. We report a selection of frames for which we have available ground-truth segmentation and show the predictions obtained both with a model trained with simple finetuning and with one trained with replay from the source and the pre-training datasets.

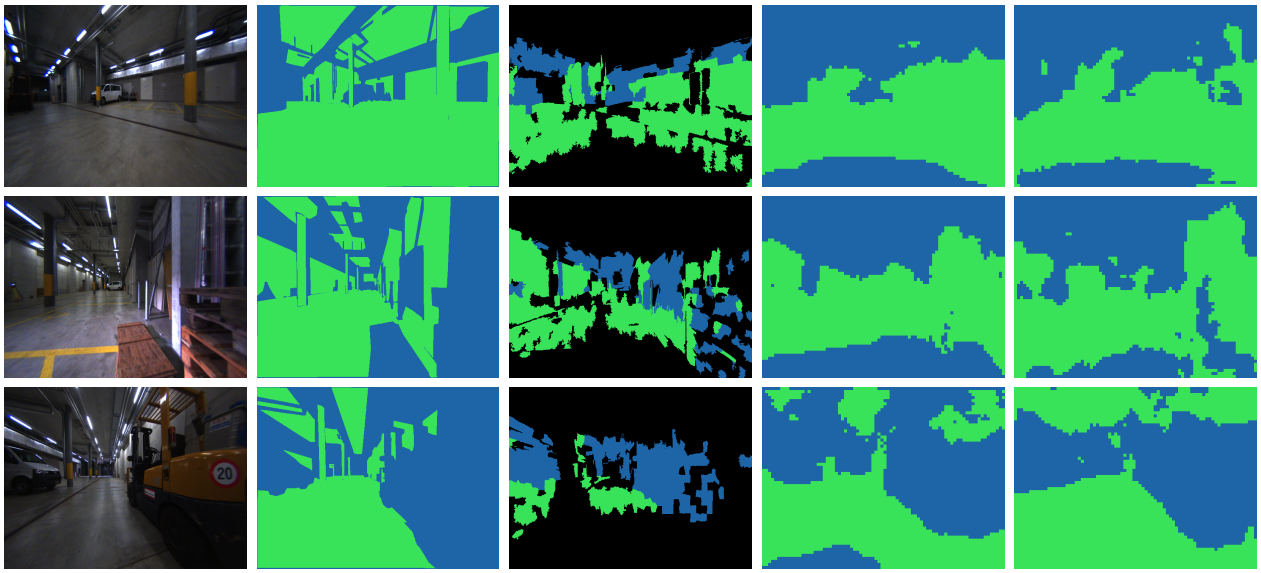
In the qualitative outputs, we observe that the models learn biases towards regions that are generally unlabeled at training time. In particular, areas in the upper and lower part of the image are commonly classified as foreground, and show a curvature that roughly reflects the regions in the training pseudo-labels where information is missing due to the reprojection of the LiDAR measurements into the camera view. This is in line with our discussion of the FoV mask, as supervision through pseudolabels is missing in those parts of the image; indeed, the learned biases in these unobserved regions often do not match the ground-truth class in these areas (cf., e.g., Fig. 6a, columns `Ground-truth segmentation` and `Prediction with replay`), and the evaluation would reflect this negatively if these areas were considered. We stress that the maksed FoV region is most relevant for our application, as it represents the overlap of camera and LiDAR scans that we aim to filter and improve localization with. However, we also provide numbers when evaluating whole camera images instead of FoV masks in Table VI. As expected, the results outside of the LiDAR FoV are more noisy. From the qualitative examples and comparison with the FoV evaluation we know that this is due to wrong biases in image regions where no pseudolabels are available.

Stage	Source → target	Segmentation quality [% mIoU]													
		NYU		Garage				Construction				Office			
		GT (no mask)		Pseudo		GT (no mask)		Pseudo		GT (no mask)		Pseudo		GT (no mask)	
		RB	FT	RB	FT	RB	FT	RB	FT	RB	FT	RB	FT	RB	FT
0	Pretraining on NYU	–	86.4	–	(22.5)	–	(40.3)	–	(22.7)	–	(29.4)	–	(39.6)	–	(46.3)
1	NYU → Garage	**68.3**	36.4	95.4	96.3	44.5	43.3	–	–	–	–	–	–	–	–
1	NYU → Construction	**78.6**	36.6	–	–	–	–	77.0	79.5	32.7	32.7	–	–	–	–
1	NYU → Office	**81.0**	66.2	–	–	–	–	–	–	–	–	69.7	70.9	53.2	51.7
2	Garage→ Construction	**70.3**	30.7	**91.8**	77.1	43.8	46.0	77.4	78.5	34.7	34.6	–	–	–	–
2	Garage→ Office	**70.9**	42.7	**92.8**	71.7	45.3	48.0	–	–	–	–	69.9	72.2	52.1	50.3
2	Construction→ Office	**78.6**	48.9	–	–	–	–	**71.3**	55.9	34.7	36.4	70.3	72.2	46.6	47.5
2	Construction→ Garage	**70.5**	36.7	94.4	95.6	43.7	44.2	**61.4**	43.3	33.1	31.0	–	–	–	–
2	Office → Garage	**68.7**	36.4	95.3	96.4	43.3	42.9	–	–	–	–	**61.2**	46.9	46.8	42.7
2	Office → Construction	**77.7**	38.8	–	–	–	–	73.1	73.0	34.1	33.7	**63.4**	44.7	46.6	36.7

TABLE VI: While we in general evalaute segmentation quality only in the overlapping field of view of cameras and LiDAR, this table serves as a comparison as to how Table V would look when evaluating the whole camera images, including regions where the segmentation never has training signals because pseudolabels cannot be generated. We observe similar trends also in this table, while the results are more noisy.

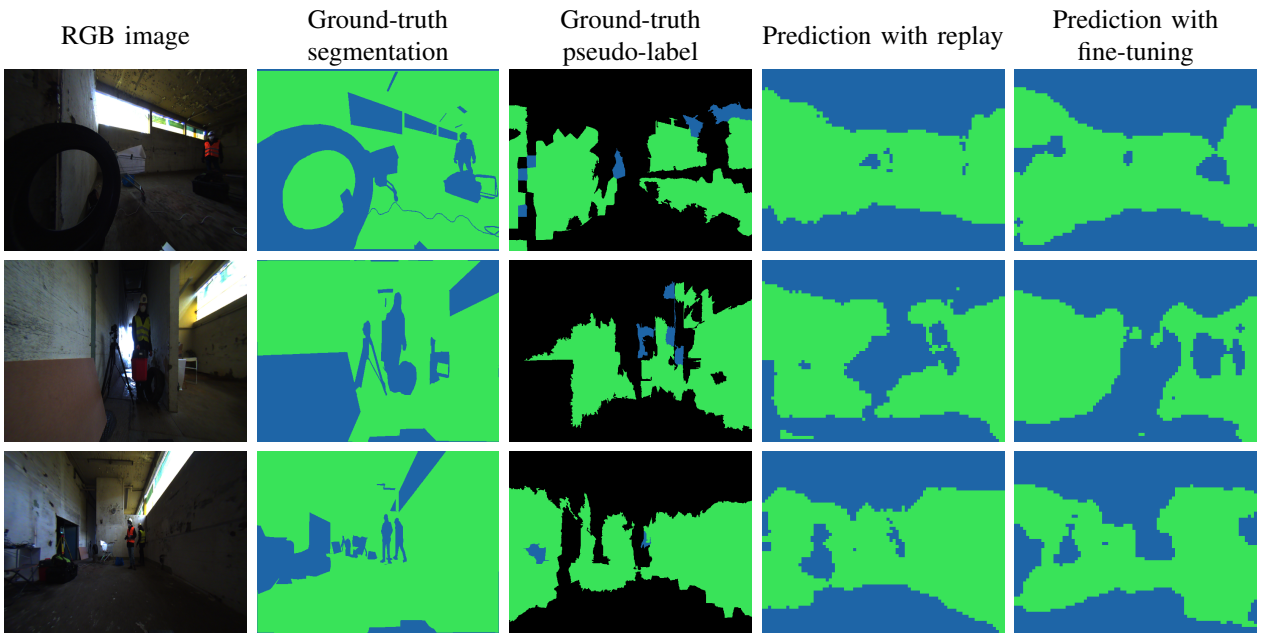


(a) `Garage→Construction`

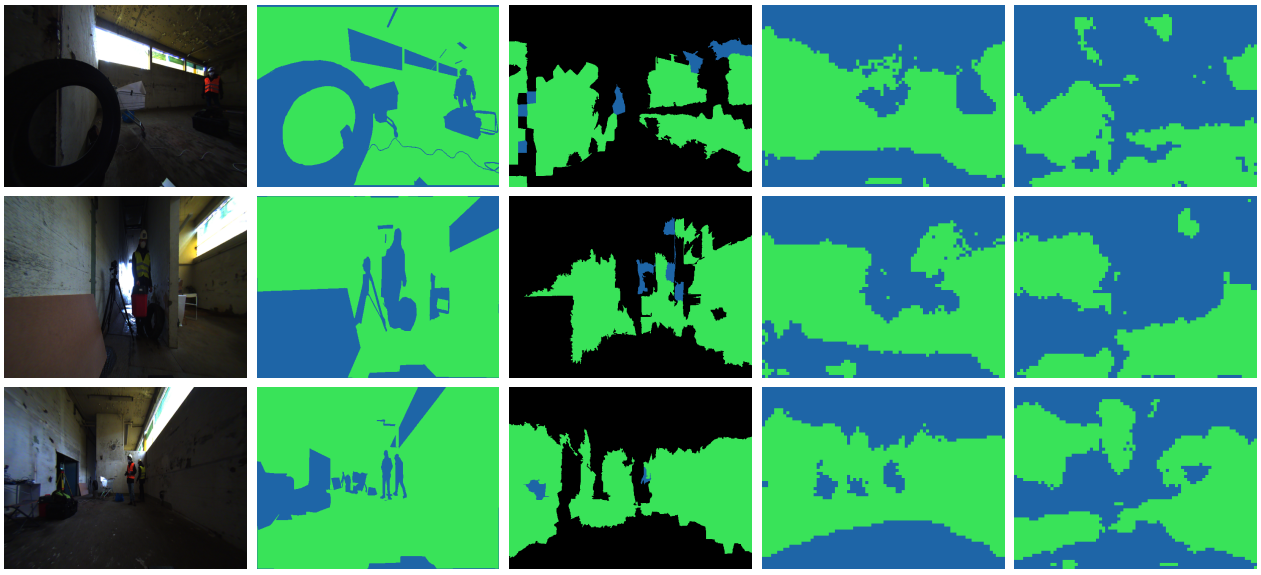


(b) `Garage→Office`

Fig. 6: Illustrations of (prevention of) forgetting for the parking garage as source environment. Green is *background*, blue is *foreground* and black pseudolabels are ignored in training.

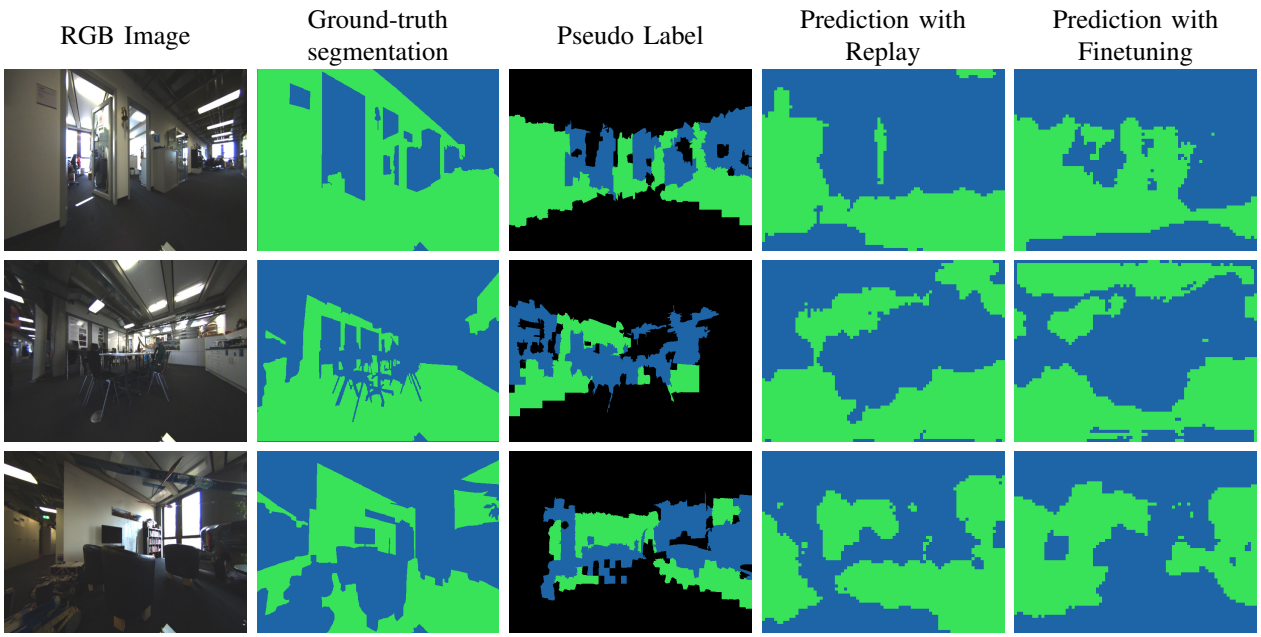


(a) `Construction→Garage`

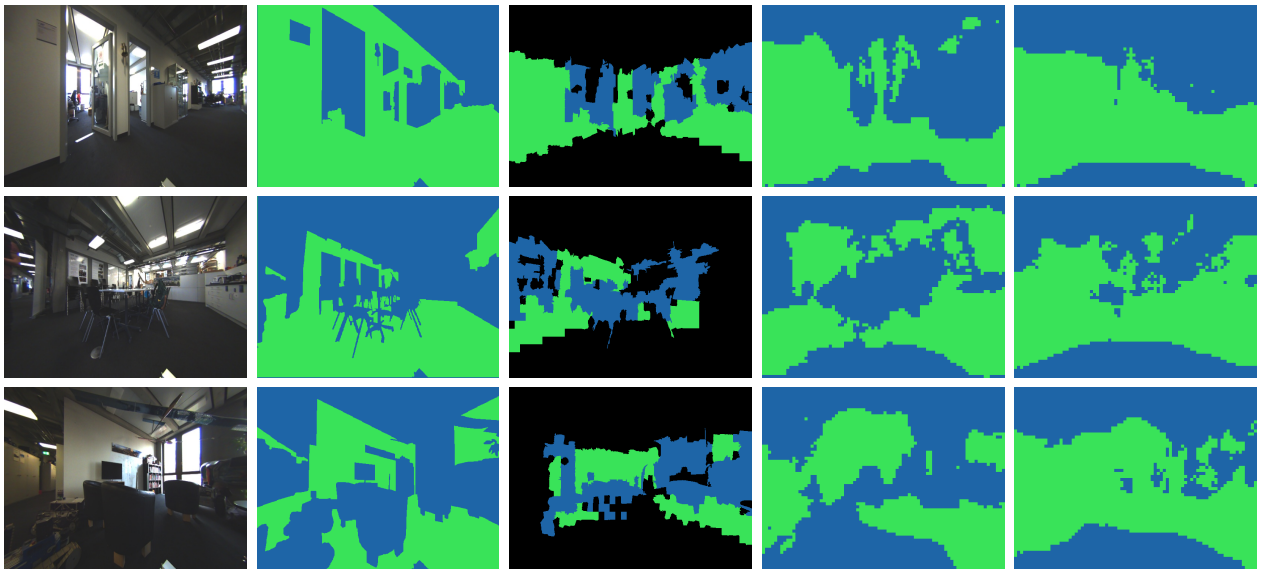


(b) `Construction→Office`

Fig. 7: Illustrations of (prevention of) forgetting for the construction site as source environment. Green is *background*, blue is *foreground* and black pseudolabels are ignored in training.

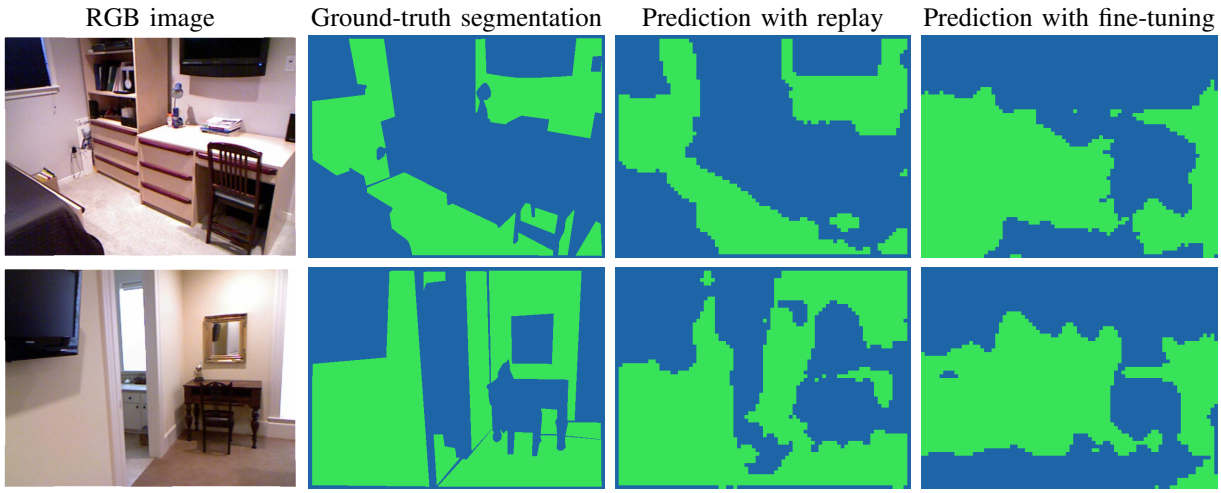


(a) `Office→Garage`

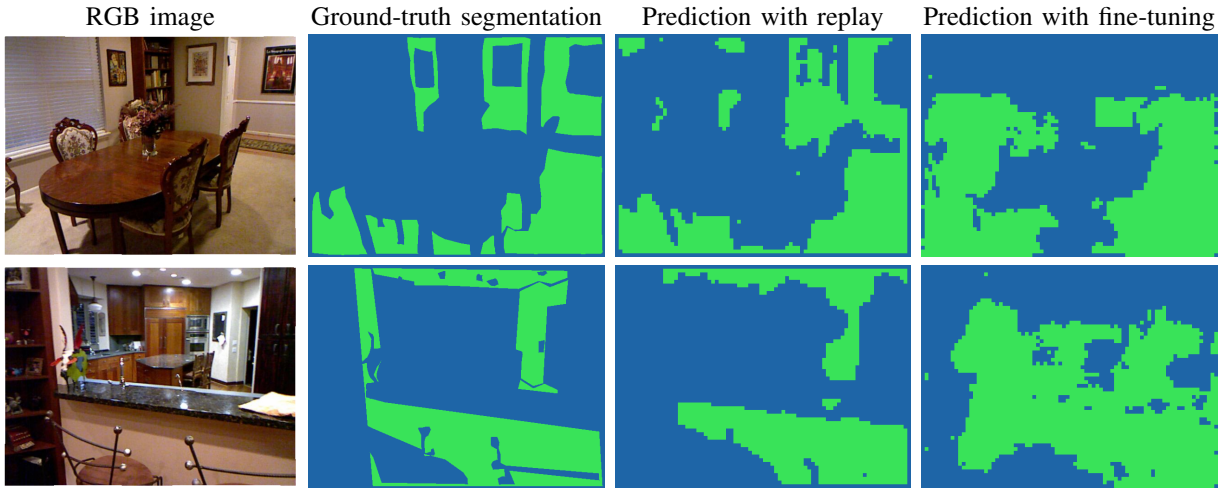


(b) `Office→Construction`

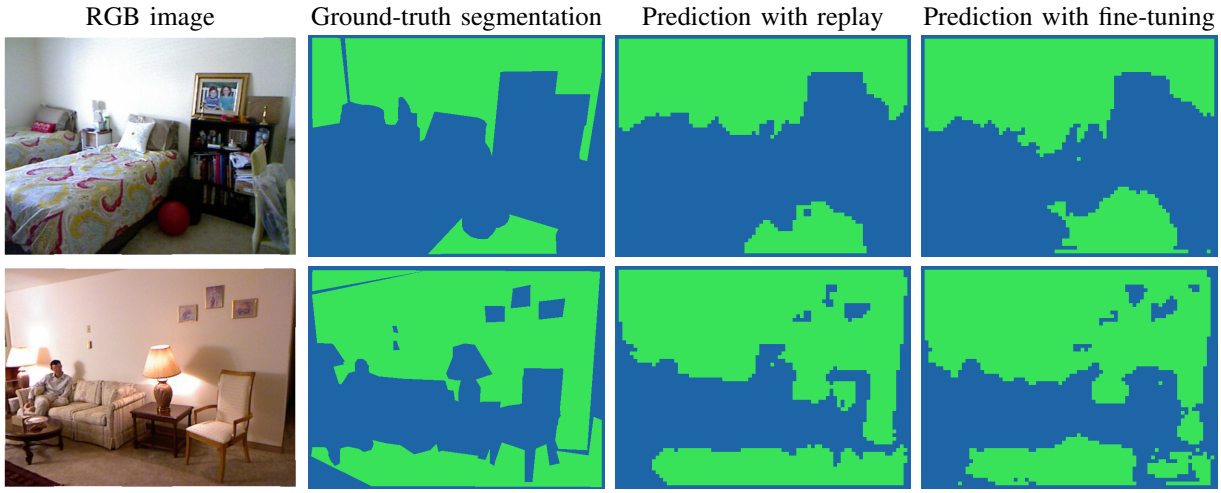
Fig. 8: Illustrations of (prevention of) forgetting for the office as source environment. Green is *background*, blue is *foreground* and black pseudolabels are ignored in training.



(a) NYU→`Garage`



(b) NYU→`Construction`



(c) NYU→Office

Fig. 9: Illustrations of (prevention of) forgetting for the NYU dataset as source environment. Green is *background*, blue is *foreground*.