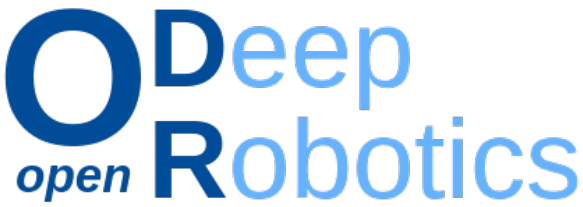


OpenDR — Open Deep Learning Toolkit for Robotics

Project Start Date: 01.01.2020
Duration: 48 months
Lead contractor: Aristotle University of Thessaloniki

Deliverable D4.4: Final report on deep environment active perception and cognition

Date of delivery: 29 September 2023

Contributing Partners: AU, AUTH, ALU-FR, TAU, AGI
Version: v2.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449.

Title	**D4.4: Final report on deep environment active perception and cognition**
Project	**OpenDR** (ICT-10-2019-2020 RIA)
Nature	Report
Dissemination Level:	**PU**blic
Authors	Niclas Vödisch (ALU-FR), Ahmet Selim Çanakçı (ALU-FR), Abhinav Valada (ALU-FR), Illia Oleksiienko (AU), Alexandros Iosifidis (AU), Avramelou Loukia (AUTH), Kakaletsis Efstratios (AUTH), Passalis Nikolaos (AUTH), Kirtas Emmanouil (AUTH), Babis Emmanouil (AUTH), Nousi Paraskevi (AUTH), Tzelepi Maria (AUTH), Symeonidis Charalampos (AUTH), Spanos Dimitrios (AUTH), Tosidis Pavlos-Apostolos (AUTH), Tsampazis Konstantinos (AUTH), Tefas Anastasios (AUTH), Nikolaidis Nikolaos (AUTH), Anton Muravev (TAU), Moncef Gabbouj (TAU), Alea Scovill (AGI), Rasmus Nyholm Jørgensen (AGI)
Lead Beneficiary	AU (Aarhus University)
WP	4
Doc ID:	OPENDR_D4.4.pdf

Document History

Version	Date	Reason of change
v0.1	9/5/2023	Deliverable template ready
v1.0	15/9/2023	Initial version of deliverable submitted for review
v1.1	22/9/2023	Reviewer comments sent and minor comments added in the document
v2.0	29/9/2023	Review comments addressed and deliverable revised

Contents

Executive Summary

This document presents the final update of the work performed for **WP4–Deep environment active perception and cognition** during the final year of the project. This work package consists of four main tasks, which are *Task 4.1–Object detection/recognition and semantic scene segmentation and understanding*, *Task 4.2–2D/3D object localization and tracking*, *Task 4.3–Deep SLAM and 3D scene reconstruction*, and *Task 4.4–Sensor information fusion*. After a general introduction that provides an overview of the individual chapters with a link to the main objectives of the project, the document dedicates a chapter to each of the tasks. Each chapter *(i)* provides an overview on the state of the art for the individual topics and *(ii)* details the partners' work in each task with performance results for the individual tasks. Finally, a concluding chapter provides a final overview of the work for each individual task.

1 Introduction

This document describes the work progress of the last year of the project in the four major tasks of *WP4–Deep environment active perception and cognition*, namely *Task 4.1–Object detection/recognition and semantic scene segmentation and understanding*, *Task 4.2–2D/3D object localization and tracking*, *Task 4.3–Deep SLAM and 3D scene reconstruction* and *Task 4.4–Sensor information fusion*. In this section, a brief description of the work conducted by the consortium in these tasks is provided, along with a short description on how this work contributes to the Objectives of the project. A more detailed description of the conducted work is provided in the following sections.

1.1 Object detection/recognition and semantic scene segmentation and understanding (T4.1)

ALU-FR contributed to this task by proposing a method for online continual learning of joint vision-based depth estimation and panoptic segmentation (Section 2.1). The method, called CoDEPS, allows continuous adaptation to previously unseen domains. AUTH continued working on methods to tackle the shortcomings associated with the one-hot encoding by introducing a framework for learning soft label embeddings (Section 2.2), by incorporating both similarity relationships at a class-level and an instance-level, demonstrating improvements in additional evaluation setups. Furthermore, AUTH also worked on developing lightweight object detection models for agricultural applications, showing that adequate performance can be obtained in many cases (Section 2.3).

1.2 2D/3D object localization and tracking (T4.2)

AU worked on incorporating uncertainty estimation into 3D perception by creating Variational Neural Networks versions of Voxel Pseudo Image Tracking for 3D Single Object Tracking (Section 3.1) and TANet for 3D Object Detection in combination with Uncertainty-Aware AB3DMOT for 3D Multiple Object Tracking (Section 3.2). Variational 3D perception models provide an improvement in the accuracy of perception and the valuable uncertainty information that can be further used in decision-based methods.

1.3 Deep SLAM and 3D scene reconstruction (T4.3)

ALU-FR continued their work reported in D4.3 towards continual learning for vision-based odometry (Section 4.1). In particular, they proposed a novel method called CoVIO that provides several improvements with respect to the simplicity of the network design and overall performance. AGI continued their work with the plant row guidance and mapping (Section 4.2). A DL-based method was analysed at 3 levels and refined. The method shows that it is possible to navigate and map the crop rows with the purposed method.

1.4 Sensor information fusion (T4.4)

TAU worked on the evaluation of the sensor fusion framework via openly available multimodal benchmarks, as well as the toolkit integration.

1.5 Connection to Project Objectives

The work carried out by the consortium in the context of WP4, as summarized in the previous subsections, is directly related to the overarching project goals. In particular, the work described here progressed the state-of-the-art towards meeting the project goals O1 and O2, as will be presented in detail below.

O1 *To provide a modular, open and non-proprietary toolkit for core robotic functionalities enabled by lightweight deep learning*

O1a *To enhance the robotic autonomy exploiting lightweight deep learning for on-board deployment*

AU improved the quality of perception of embedded 3D single and multiple object tracking methods by estimating and utilizing uncertainty with the use of Variational Neural Networks (Sections 3.1 and 3.2).

O1b *To provide real-time deep learning tools for robotics visual perception on high-resolution data*

AUTH worked on developing lightweight object detection algorithms and models for agricultural applications (Section 2.3).

O2 *To leverage AI and Cognition in robotics: from perception to action*

O2a *To propose, design, train and deploy models that go beyond static computer perception, towards active robot perception*

ALU-FR contributed to this task by proposing several methods for online continual learning for the tasks of both panoptic segmentation and visual odometry (Sections 2.1 and 4.1). These methods enable active adaptation to previously unseen domains.

O2c *To provide tools for enhanced robot navigation, action and manipulation capabilities that can exploit if needed deep environment active robot perception*

AUTH continued working on methods to tackle the shortcomings associated with the one-hot encoding by introducing a framework for learning soft label embeddings that can provide enhanced visual perception for various applications (Section 2.2). Additionally, ALU-FR continued their work towards online adaptation of visual odometry to previously unseen domains (Section 4.1). AGI continued their work with the plant row guidance system, which enables the agricultural field robot to navigate and map based on actual placement of the plant rows instead of RTK GNSS (Section 4.2). TAU worked towards objective O2c by experimentally validating the performance and robustness of the multimodal feature fusion framework (Section 5), with integration to follow.

2 Object detection/recognition and semantic scene segmentation and understanding

2.1 CoDEPS: Continual Learning for Depth Estimation and Panoptic Segmentation

2.1.1 Introduction and objectives

Operating robots in unfamiliar environments requires adaptability. Ideally, robots can autonomously adjust, e.g., adapting their perception to changing lighting. Deploying robots, like autonomous cars in cities, demands a comprehensive understanding, including semantics, instances, and depth perception. This enables vision-based methods to create a 3D semantic scene for tasks like localization and planning. However, deep learning approaches may struggle when moving to new domains with different environmental conditions [93] or sensor parameters [3, 10]. This domain gap poses a challenge for robots in unfamiliar environments. Unlike the source domain, the target domain usually lacks ground truth annotations, making classical domain adaptation unsuitable. Unsupervised domain adaptation addresses these limitations.

Most prior approaches focus on sim-to-real domain adaptation, often offline, with knowledge transfer but without addressing catastrophic forgetting or considering abundant target annotations [28, 56]. Additionally, these methods may not account for robotic platform constraints, like limited computational resources and storage capacity [41, 97].

In this work, we employ online continual learning for depth estimation and panoptic segmentation in a multi-task setting. Using images from an onboard camera as illustrated in Fig. 1, we enhance performance during inference. Our method, CoDEPS, mitigates forgetting by using experience replay of both source and previous target images. We incorporate a replay buffer and introduce cross-domain mixing for unlabeled target data. We address hardware constraints by using a single GPU and maintaining a fixed-size replay buffer. CoDEPS effectively improves performance in new target domains without sacrificing performance in previous domains.

The main contributions of this work are as follows:

1. We introduce CoDEPS, the first online continual learning approach for joint monocular depth estimation and panoptic segmentation.

2. We propose a novel cross-domain mixing strategy to adapt panoptic segmentation to unlabeled target data.

3. To address the storage restrictions of robotic platforms, we leverage a fixed-size replay buffer based on rare class sampling and image diversity.

4. We extensively evaluate CoDEPS and compare it to other methods in challenging real-to-real settings.

A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.1:

- [100] N. Vödisch, K. Petek, W. Burgard, and A. Valada, "CoDEPS: Online Continual Learning for Depth Estimation and Panoptic Segmentation", *Robotics: Science and Systems (RSS)*, 2023.

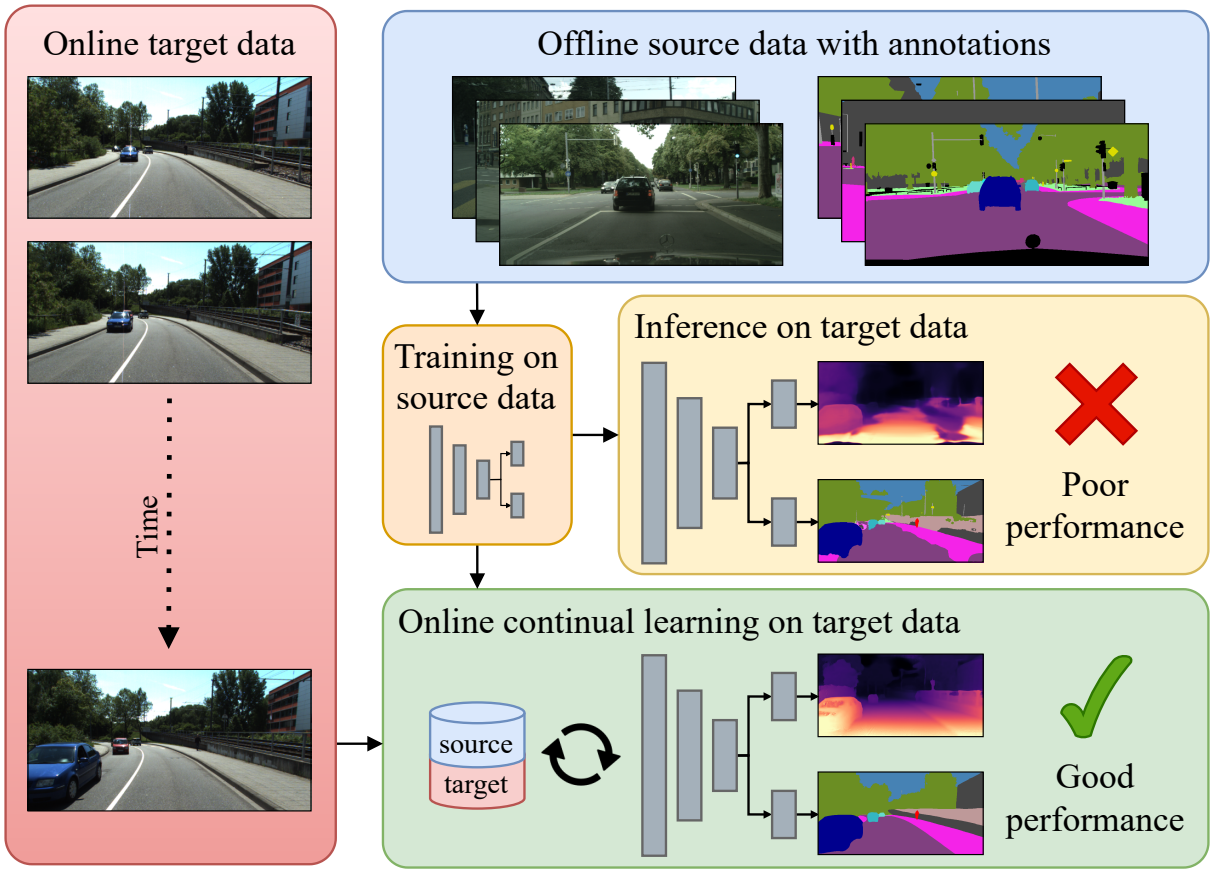


Figure 1: Neural networks can struggle in new domains. To tackle this, we propose continuous adaptation with real-time target images. We use a fixed-size replay buffer to mitigate forgetting and improve generalization by revisiting source and target data.

2.1.2 Summary of state of the art

Monocular depth estimation predicts a depth map from a single RGB image. Supervised methods use range sensors for network supervision [79], while unsupervised methods rely on temporal geometric cues [25, 114]. Panoptic segmentation merges semantic and instance segmentation, classifying pixels into semantics and "thing" classes. Networks use joint encoders and separate decoders, with approaches including bottom-up [9, 66] and top-down [26, 65] methods. Domain adaptation bridges the gap between a source $\mathcal{S}$ used for training and a target $\mathcal{T}$ for inference. This is related to continual learning (CL) [55], adapting to changing tasks while avoiding catastrophic forgetting, ideally with positive forward transfer. Real-world scenarios often lack target annotations, requiring unsupervised domain adaptation (UDA). Offline UDA assumes target data availability, while online UDA enables continuous robot operation in new domains without previous target data collection. Offline UDA combines annotated source data and abundant unlabeled target data to adapt networks from $\mathcal{S}$ to $\mathcal{T}$. For depth estimation, DESC [56] adapts by source-to-target style transfer and consistency loss between depth predictions from RGB and semantic maps. GUDA [28] adapts semantic segmentation using shared encoders for depth and semantic segmentation, improving both tasks. Cross-domain sampling in DACS [90] and ConfMix [62] enhances adaptation through pixel-level copying and mixing strategies. Huang et al. [34] propose a UDA method for panoptic segmentation by regularizing complementary features. Online UDA faces challenges due to consecutive

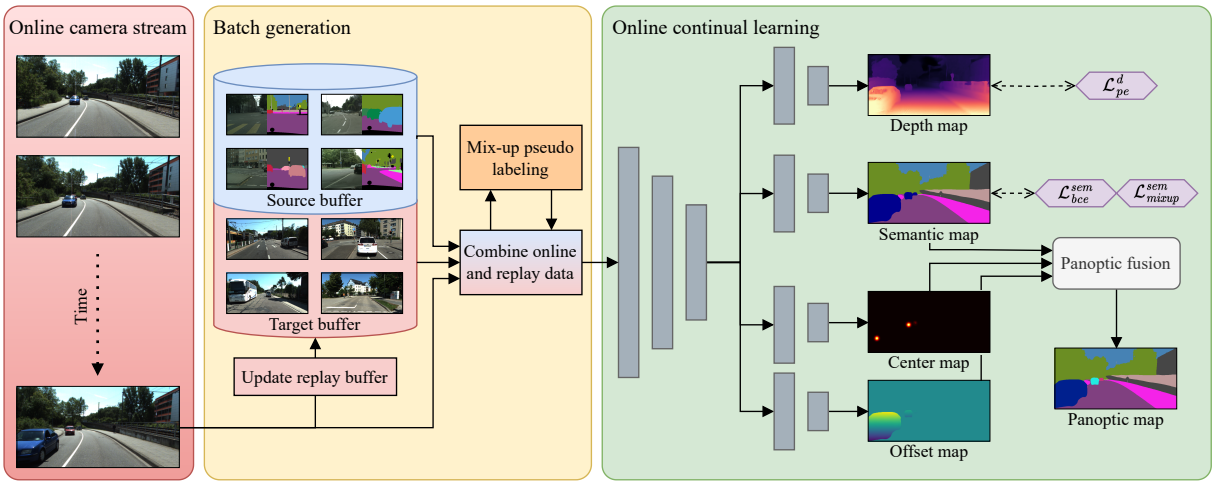


Figure 2: Overview of our proposed CoDEPS

sampling from $\mathcal{T}$, resembling a camera's image stream. These challenges include reduced diversity and model overfitting to the scene [111]. Continual SLAM [97] improves visual SLAM using unsupervised depth estimation as a proxy task and prevents forgetting by incorporating samples from $\mathcal{S}$ and previous target domains $\mathcal{T}_i$. For semantic segmentation, CBNA [38] updates batch normalization layers by mixing statistics from $\mathcal{S}$ and $\mathcal{T}$. CoTTA [105] adapts the network without source data, using self-supervision and techniques to mitigate error accumulation.

Our approach is the first for online continual UDA, addressing joint depth estimation and panoptic segmentation.

2.1.3 Work description

The setting investigated in this work consists of two steps. First, we train a neural network on the source domain $\mathcal{S}$ partly using ground truth supervision. Second, to close the gap between domains, we continuously adapt the network during inference time on the target domain $\mathcal{T}$ using a replay buffer and unsupervised training strategies.

We adopt a standard multi-task network design, employing a single ResNet-101 [30] as the shared encoder for depth prediction, semantic segmentation, and instance segmentation tasks. Our network architecture, outlined in Fig. 2, consists of a depth head inspired by Monodepth2 [24], featuring five consecutive convolutional layers with skip connections to the encoder. Additionally, we include a separate PoseNet consisting of a ResNet-18 encoder and a four-layer CNN to estimate the camera motion between two image frames. For panoptic segmentation, we follow the bottom-up method Panoptic-Deeplab [9], leveraging separate heads for semantic segmentation and instance segmentation, and slightly modify the semantic head [28]

Source Domain Pretraining: During the initial training phase on the source domain, we assume to have access to image sequences as well as ground truth panoptic segmentation annotations. In the following, we briefly describe the respective loss functions that we employ for training the three task-specific heads.

We train the depth estimation head using the common methodology of unsupervised training based on the photometric error [24].

$$\mathcal{L}^d_{pe} = \lambda_{pr}\mathcal{L}^d_{pr} + \lambda_{sm}\mathcal{L}^d_{sm}. \quad (1)$$

We train the semantic segmentation head in a supervised manner using the bootstrapped cross-entropy loss with hard pixel mining $\mathcal{L}_{bce}^{sem}$ following Panoptic-Deeplab [9].

For training the instance segmentation head, we adopt the MSE loss $\mathcal{L}_{center}^{ins}$ for the center head and the L1 loss $\mathcal{L}_{offset}^{ins}$ for the offset head. The total loss to supervise instance segmentation is then computed as a weighted sum:

$$\mathcal{L}_{co}^{ins} = \lambda_{center}\mathcal{L}_{center}^{ins} + \lambda_{offset}\mathcal{L}_{offset}^{ins}. \tag{2}$$

Online Adaptation: After the described network has been trained on the source domain $\mathcal{S}$ using the aforementioned losses, we aim to adapt it to the target domain $\mathcal{T}$ in a continuous manner. That is, unlike other works, data from the target domain is revealed frame by frame resembling the online stream of an onboard camera. As depicted in Fig. 2, every adaptation iteration consists of the following steps:

1. Construct an update batch by combining online and replay data.
2. Generate pseudo-labels using the proposed cross-domain mixing strategy.
3. Perform backpropagation to update the network weights.
4. Update the replay buffer.

Replay Buffer and Batch Generation: Upon receiving a new image taken by the robot's onboard camera, we construct a batch that is used to perform backpropagation on the network weights. In detail, a batch consists of the current online image, previously received target images, and fully annotated source samples. By revisiting target images from the past, we increase the diversity in the loss signal on the target domain and hence mitigate overfitting to the current scene. This further accounts for situations in which the current online image suffers from visual artifacts, e.g., overexposure. Similarly, the problem of catastrophic forgetting by ensuring that previously acquired knowledge can be preserved. Additionally, the annotations of the source samples enable pseudo-supervision on the target domain by exploiting cross-domain mixing strategies. For both the target and the source replay, we randomly draw multiple samples from the respective replay buffer and apply augmentation to stabilize the loss. In particular, we perform RGB histogram matching of the source images to the online target image, and all available source samples have to be selected once before repetition is allowed to ensure diverse source supervision.

Depth Adaptation: To adapt the monocular depth estimation head along with the PoseNet, we exploit the fact that the photometric error loss does not require ground truth annotations. Hence, we can directly transfer it to the implemented continual adaptation.

Panoptic Adaptation: Panoptic segmentation is the fused output of a semantic head and an instance head. We observe that the decrease in performance on samples from unseen domains can mostly be attributed to the semantic head, while instance predictions remain stable. In CoDEPS, we bootstrap annotated source samples and high-confident target predictions to artificially generate pseudo-labels for the target samples in an online fashion to supervise the semantic head using $\mathcal{L}_{bce}^{sem}$. The instance head remains frozen.

2.1.4 Performance evaluation

To simulate data from a variety of domains, we employ our method on three datasets, namely Cityscapes [11], KITTI-360 [51], and SemKITTI-DVPS [1]. In particular, we utilize Cityscapes for pre-training and sequences of both KITTI-360 and SemKITTI-DVPS for adaptation. In the supplementary video of the published work, we further provide qualitative results on in-house data recorded with our robotic platform.

Table 1: Adaptation Performance

Method	Seq.	Protocol 1						Protocol 2					
		mIoU	PQ	SQ	RQ	RMSE	Abs Rel	mIoU	PQ	SQ	RQ	RMSE	Abs Rel
Only source	00	51.61	39.10	72.72	50.48	6.54	0.36	49.94	35.29	72.14	45.50	6.08	0.34
CoDEPS		**53.76**	**40.72**	**72.90**	**52.51**	**5.09**	**0.19**	**52.08**	**36.08**	**72.58**	**46.08**	**4.34**	**0.15**
Only source	02	45.97	31.83	67.62	41.08	6.26	0.35	46.55	30.13	65.03	39.30	6.06	0.36
CoDEPS		**46.62**	**32.11**	**67.74**	**41.62**	**4.31**	**0.16**	**47.48**	**30.33**	**65.35**	**39.46**	**3.76**	**0.13**
Only source	03	46.63	28.15	57.41	35.23	**8.20**	0.34	**52.10**	28.20	56.67	35.77	7.34	0.29
CoDEPS		**47.94**	**29.05**	**58.07**	**36.10**	8.26	**0.33**	52.00	**31.13**	**61.51**	**39.65**	**6.98**	**0.18**
Only source	04	45.02	29.34	65.48	38.15	6.70	0.37	45.53	30.13	**70.85**	38.89	6.61	0.38
CoDEPS		**45.40**	**29.78**	**65.89**	**38.84**	**5.00**	**0.19**	**45.68**	**30.63**	66.18	**39.89**	**4.33**	**0.17**
Only source	05	48.94	32.19	66.80	41.37	6.76	0.37	**44.52**	**27.34**	**60.72**	**35.58**	5.93	0.43
CoDEPS		**49.26**	**32.96**	**66.98**	**42.40**	**5.25**	**0.21**	43.79	26.48	60.33	34.88	**4.68**	**0.25**
Only source	06	46.03	29.88	66.58	38.42	6.09	0.39	46.28	31.79	70.47	41.40	6.12	0.37
CoDEPS		**46.53**	**30.45**	**66.66**	**39.20**	**4.97**	**0.22**	**47.27**	**31.99**	**70.74**	**41.71**	**4.23**	**0.18**
Only source	07	40.54	28.48	66.52	34.42	7.83	0.34	59.07	27.62	45.88	35.41	9.64	0.38
CoDEPS		**41.46**	**29.30**	**67.64**	**35.58**	**6.50**	**0.22**	**60.57**	**30.91**	**50.25**	**39.79**	**6.48**	**0.20**
Only source	09	50.59	37.26	74.06	47.38	6.03	0.36	50.78	36.57	72.22	46.75	5.60	0.35
CoDEPS		**52.29**	**38.02**	**74.88**	**48.21**	**4.74**	**0.19**	**51.53**	**37.56**	**72.87**	**47.99**	**4.56**	**0.16**
Only source	10	51.94	32.60	71.27	32.60	8.06	0.35	45.74	30.62	69.56	39.49	7.90	0.33
CoDEPS		**53.02**	**33.50**	**71.53**	**33.50**	**7.19**	**0.22**	**49.91**	**31.91**	**70.68**	**40.95**	**5.57**	**0.15**

Comparison between our CoDEPS and the performance of the same architecture without performing online continual learning on the respective sequence of the KITTI-360 dataset. Thus, "only source" refers to the model weights after pretraining on Cityscapes. The listed metrics are mean intersection over union (mIoU) for semantic segmentation; panoptic quality (PQ), segmentation quality (SQ), and recognition quality (RQ) for panoptic segmentation; root mean squared error (RMSE) and absolute relative error (Abs Rel) for monocular depth estimation. Bold values denote the best result on each sequence.

In Tab. 1, we assess the performance of CoDEPS on all sequences of the KITTI-360 dataset and compare it with the baseline method "only source", which is also pretrained on Cityscapes but does not perform further adaptation to the target domain $\mathcal{T}$. This approach should be interpreted as a lower performance bound that must be improved. We demonstrate the key performance metrics of both protocols 1 and 2. For more results and the definition of these protocols, please refer to the original publication.

2.2 Deep Label Embedding Learning for Classification

2.2.1 Introduction and objectives

The popular one-hot encoding method, typically used in classification problems in conjunction with the cross-entropy loss function, fails to capture the inherent uncertainty in data labeling processes. Among the sources of uncertainty is the similarity between

classes in a dataset, e.g., a class of objects can resemble another class or be very dissimilar from others. Furthermore, certain instances may be ambiguous, even for human annotators. One-hot encoding, due to its crisp nature, can also lead to overfitting more easily. As an example, consider a binary classification example, where a difficult sample may be correctly classified with a predicted probability of 0.8. Despite the prediction being correct, most recent approaches to this problem would further modify the network weights such that the predicted probability lies closer to 1. In aiming to learn this crisp probability, the network loses some of its generalization ability.

AUTH continued working on methods to tackle the shortcomings associated with the one-hot encoding by introducing a framework for learning soft label embeddings. Similarity relationships both at a class-level and an instance-level were taken into into the proposed label embedding methods. Two variants were proposed: first, a learnable general-class embedding which aims to capture information regarding interclass similarities, and second, a neural architecture which can be added to any neural classifier and aims to learn inter-instance similarities. The inherent uncertainty in data labels is thus somewhat alleviated, allowing the network to focus on incorrectly classified samples, instead of difficult but correctly classified ones. The concept of this framework is similar to real-life learning procedures, where uncertainty rules over many subjects.

A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.2:

- P. Nousi and A. Tefas, "Deep Label Embedding Learning for Classification", *technical report, under review.*

2.2.2 Summary of state of the art

Soft labels and label embedding have been studied in the past, in the context of simpler classifiers like the k-Nearest Neighbor one [14], and recently scientific research in this task has resurfaced. In [16], a method for modeling subjectivity in emotion recognition was proposed. An ensemble method, where different networks are trained with different annotators, was compared against a single network trained on soft labels, generated by averaging the labels from different annotators. In [99], soft labels were learned in a meta learning fashion, by treating them as learnable parameters, modeling both class-level and instance-level similarities. Soft labels have also recently been linked to knowledge distillation methods [92, 91].

2.2.3 Work description

A summary of the proposed method follows. Using soft label embeddings, the learning task is formulated as:

$$CE(\hat{\mathbf{y}}, \mathbf{p}) = -\sum_{k=1}^{K} \hat{y}_k \cdot \log(p_k), \tag{3}$$

where $\hat{\mathbf{y}}$, i.e., the soft labels, are given by a differentiable function $g(\cdot)$, the parameters of which can be learned in conjunction with the parameters of the classifier during training, and CE denotes the cross-entropy loss function.

General class similarities We formulate the function $g(\cdot)$ for the case of general class similarities as a simple embedding of the form $\hat{\mathbf{y}} = g(\mathbf{y}) = \mathbf{W} \cdot \mathbf{y}$ where $\mathbf{W} \in \mathbb{R}^{K \times K}$ is a learnable weights matrix. This formulation allows $\mathbf{W}$ to take on the form of an identity matrix in the case of very crisp classes. Otherwise we expect the weights of this matrix to capture similarities and correlations between the classes present in a dataset.

Instance level similarities In the instance-specific case, the soft labels for each instance $\mathbf{x}_i$ are a function of the instance itself, that is:

$$\hat{\mathbf{y}}_i = h(\mathbf{x}_i). \tag{4}$$

The criterion from Eq. (3) still applies in this case and the function $h(\cdot)$ can take the form of a network which takes $\mathbf{x}_i$ as input and outputs $\hat{\mathbf{y}}_i$. The proposed method introduced the use of an Autoencoder as the function $h(\cdot)$, whose hidden dimension is set to match the number of classes in the dataset and which is trained with a double objective: first, the latent representation should retain enough information to reconstruct the original input samples, and second, the activations of the neurons in the latent representation should be higher for the corresponding classes, which is imposed with an additional cross-entropy loss at the hidden representation level.

Combined general class and instance level similarities Finally, the two variants can be combined in a single end-to-end trainable architecture to capture both general class and instance level similarities at the same time without additional pretraining steps. In this case, the classification network is trained to learn the soft labels from the first and second methods combined.

2.2.4 Performance evaluation

AUTH performed additional experiments on larger datasets to highlight the proposed method's ability to capture similarities between classes.

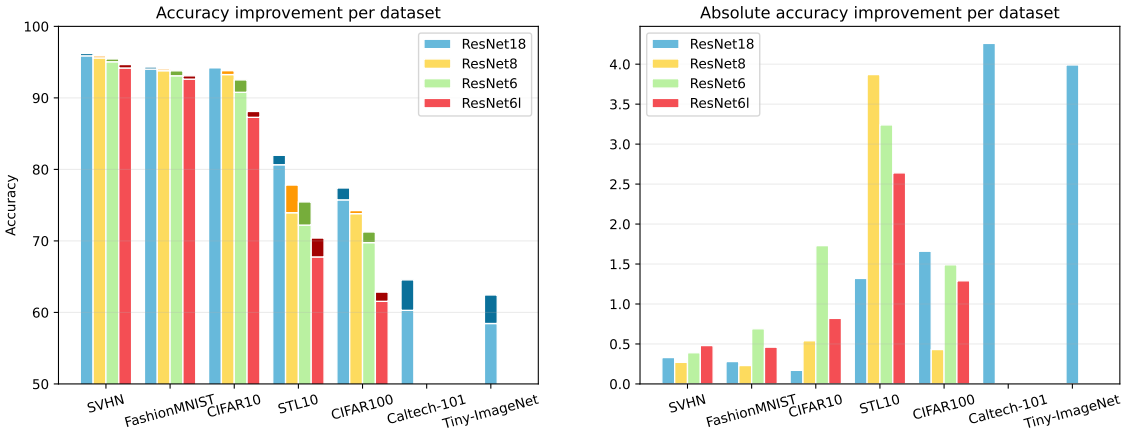
Using a ResNet18 model [30], we evaluated the proposed methods on the larger and more challenging Caltech-101 [47] and Tiny-Imagenet datasets [44]. All of the proposed variants improve the performance of the model on all datasets. Notice that the combined (GC+IS) method performs better than the general class (GC) and instance-specific (IS) methods for four out of seven datasets for this model. The performance gain is greater in the CIFAR10, CIFAR100, STL10, Caltech-101 and Tiny-ImageNet datasets, and smaller for SVHN and FashionMNIST, although it is consistent over the baseline training in all datasets and networks.

The results are also visualized in Figure 3, where the accuracy improvement for different networks is shown, and the datasets have been arranged in order of increasing difficulty. The improvement that the proposed method offers is larger for the more complex datasets.

Finally, Figure 4 is a visualization of the GC targets pre-softmax for 30 classes of the Caltech-101 dataset. The classes shown are sorted by their contribution to the "*Face*" class, resulting in this specific pattern. Note that the method is able to capture the similarity between intuitively similar classes, with the most notable example being the classes "*Faces*" and "*Faces_easy*". Intuitively, a classifier confusing these two classes for

Table 2: ResNet-18 results on all datasets in terms of classification accuracy.

Dataset	Baseline	GC	IS	GC+IS
SVHN	95.88 ± 0.08	**96.30** ± 0.13	**96.24** ± 0.07	**96.21** ± 0.06
FashionMNIST	94.03 ± 0.18	**94.07** ± 0.15	**94.42** ± 0.10	**94.31** ± 0.12
CIFAR10	94.23 ± 0.10	**94.38** ± 0.07	**94.30** ± 0.07	**94.40** ± 0.03
STL10	80.65 ± 0.27	**82.01** ± 0.21	**81.88** ± 0.33	**81.97** ± 0.48
CIFAR100	75.74 ± 0.30	**77.15** ± 0.18	**77.27** ± 0.32	**77.40** ± 0.20
Caltech-101	60.29 ± 0.55	**61.53** ± 1.32	**64.19** ± 1.12	**64.55** ± 1.35
Tiny-ImageNet	58.45 ± 0.18	**60.58** ± 0.28	**62.26** ± 0.26	**62.44** ± 0.10



(a) (b)

Figure 3: (a) Baseline and GC+IS accuracy, and (b) absolute accuracy improvement, for baseline and GC+IS methods.

each other should be punished less than it would for confusing them with other completely unrelated classes, an intuition that is captured by the proposed framework.

2.3 Lightweight High Resolution Weed Detection

2.3.1 Introduction and objectives

In the ever-evolving landscape of computer vision and object detection, the demand for real time and accurate algorithms has never been more pressing. Furthermore, high resolution images, with their wealth of detail, hold the potential to revolutionize industries. High-resolution inputs, often containing millions of pixels, offer the promise of detection of very small objects, even when the camera is relatively far away from the observed objects. Yet, with great resolution comes great computational demands that most embedded devices are not capable to overcome. With clever use of modern module architectures, optimization and training schemes we enabling the rapid identification of objects within vast image landscapes.

To this end, we used an agricultural dataset, the RoboWeedMap dataset which consist of 2054×2456 pixels images and has annotations of very small objects. For these reasons

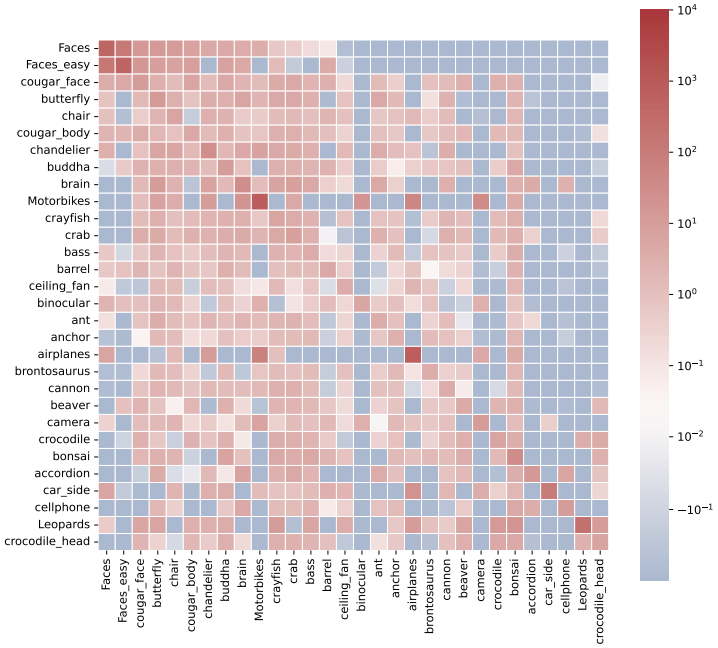
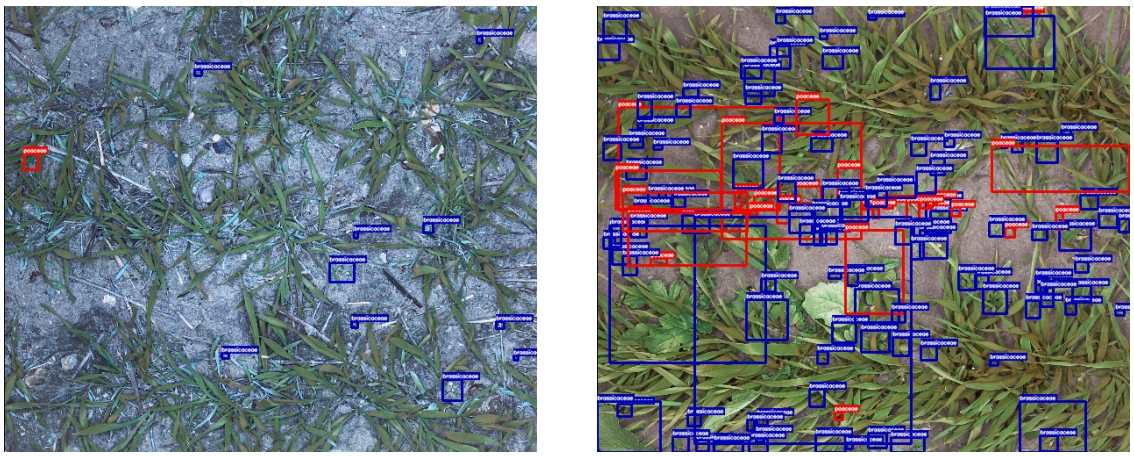


Figure 4: Caltech101 generic class similarities as captured by the proposed GC method.

object detection task in RoboWeedMap proven to be a difficult task, especially for small annotations. Samples of this dataset are shown in Figure 3.



(a) (b)

Figure 5: Examples of images in RoboWeedMap dataset.

2.3.2 Summary of state of the art

Developing fast object detection algorithms is a major research topic and we have witnessed remarkable advancements in terms of both speed and accuracy. In [82], the Region Proposal Networks (RPN) have paved the way for faster inference while maintaining competitive accuracy. From studies like [80], [53] the Fully Convolutional Networks have been proven to be by far the fastest, while new training strategies, like teacher networks [77],

[76] and different losses functions [83], [52], [50] are provide better over all accuracy. Furthermore, new network architecture modules have been studied [88], [13], [29], [33] to provide better information extraction without compromise on computational cost.

2.3.3 Work description

We provide a comprehensive overview of the work undertaken in developing our novel fully convolutional network model, which draws inspiration from the Nanodet neural network family. The proposed model employs a) a simple VGG backbone feature extractor, capable of analyzing high resolution input b) Gost Pan Feature Pyramid Network [29], which can best use our feature extractor output, and c) the Nanodet plus head with Generalized Focal Loss [50]. The objective of this work is to create an efficient yet highly capable object detection model tailored for high-resolution inputs, with a focus on the RoboWeedMap Dataset. Below, we detail the key components and innovations that constitute our model.

VGG Backbone Feature Extractor: Our model begins with a VGG-inspired backbone feature extractor. This component is responsible for analyzing high-resolution input efficiently. Specifically, the VGG backbone consists of 7 layers, each comprising a Convolution - Batch Normalization - LeakyReLU stack. This structure enables the extraction of rich features from the high resolution input data. The operation of each layer can be represented as:

$$\mathbf{X}_i = LeakyReLU(\mathbf{W}_i\mathbf{X}_{i-1} + \mathbf{b}_i), for \quad i = 1, 2, ...7, \tag{5}$$

where $\mathbf{X}_i$ represents the output feature map at layer i, $\mathbf{W}_i$ denote the convolutional weights, and $\mathbf{b}_i$ represents the biases.

Ghost Pan Feature Pyramid Network (FPN): Our model incorporates a Ghost Pan FPN, which plays a crucial role in efficiently utilizing the feature extractor's output. The Ghost Pan FPN comprises fast DepthWise-PointWise Convolution Modules and two Ghost Convolution Modules (GCM). GCMs are designed to aggregate multi-scale information effectively. In this work we implement a two stream pyramid that each input passes from a fast DepthWise - PointWise Convolution Modules and then we use the feature rich, small feature map along side:

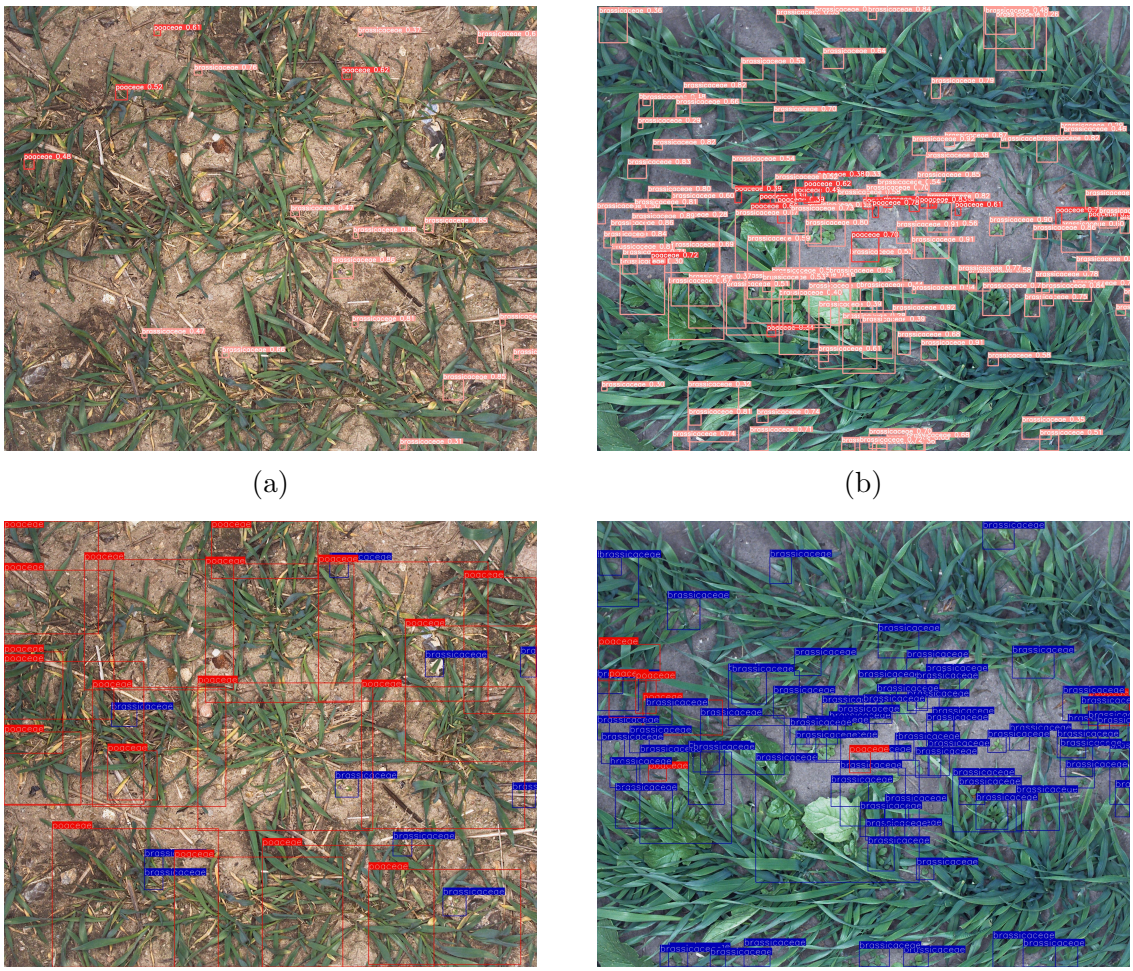
$$\mathbf{O}_1 = GhostModule(Concant(DWPW(\mathbf{X}_6), Upsample(DWPW(\mathbf{X}_7))), \tag{6}$$

and

$$\mathbf{O}_2 = GhostModule(Concant(DWPW(\mathbf{O}_1), DWPW(\mathbf{X}_7)), \tag{7}$$

where $\mathbf{X}_6$, $\mathbf{X}_7$ are the two last feature maps of our backbone, $DWPW$ is a fast DepthWise-PointWise Convolution Module, $Upsample$ is a simple up-sample interpolation module and $GhostModule$ suggests the Ghost Module in [29] that keeps the feature number integrity with a simpler computationally process.

Nanodet Plus Head with Generalized Focal Loss At the head of our model, we implement a small Nanodet Plus head consisting of two branches that leverage fast DepthWise-PointWise Convolution Modules with a single 2DConvolution at the end. The



(a) (b)

(c) (d)

Figure 6: Detection examples in RoboWeedMap dataset: (a), (b) represents the output from YoloV5S model and (c), (d) represents the output from Fast Nanodet model.

output of the convolution is flattened and all branches are concatenated into one final output feature map. This head is responsible for producing object detection predictions, including bounding box regressions and class probabilities. The number of output channels from this head is formulated as:

$$OutChannels = (4 \cdot \text{regression outputs}) + \text{number of classes}. \tag{8}$$

This head is designed to trained with Generalized Focal Loss [50] which instead to push the model to output bounding boxes coordinates, it pushes it to learn the underline General distribution of given bounding box from the regression outputs. In Figure 7, we depict the Fast Nanodet Model with feature maps sizes when an image of RoboWeedMap dataset is fed to the model as input.

2.3.4 Performance evaluation

Our first step was to test our model inference time in embedded devises and then downscale the input of YoloV5S to run at the same speed as our implementation. In Table 3, we can see that Nanodet has real time performance in Jetson TX2 when optimized with

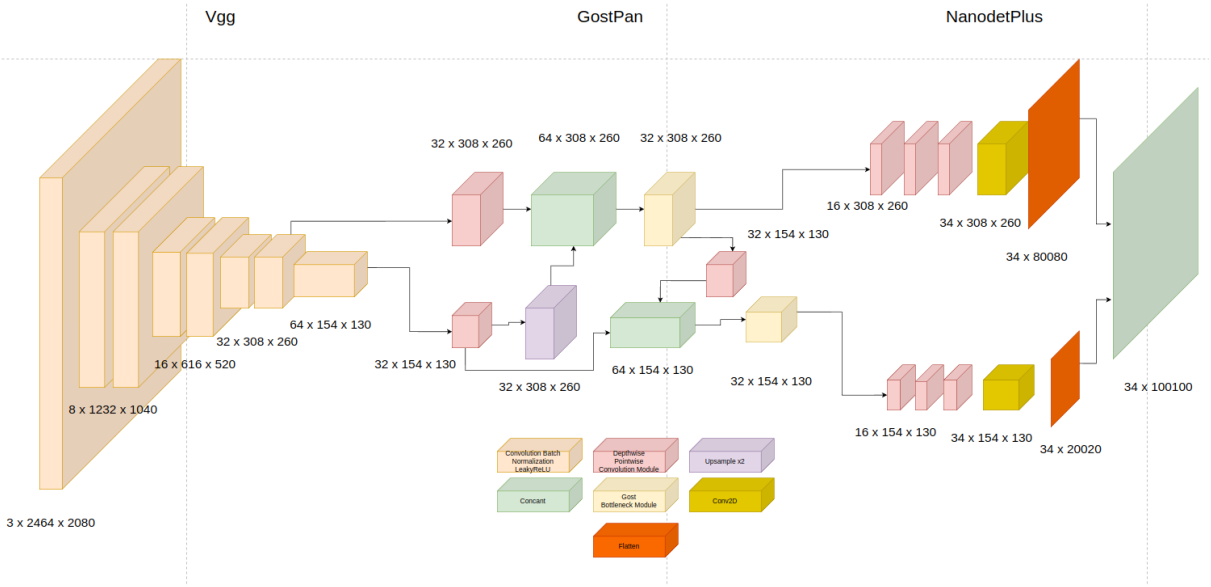


Figure 7: Fast Nanodet architecture

TensorRT with half precision weights and it is almost 5 times faster than YoloV5S in the same resolution images in our test bench PC environment. To compensate for that we downscale 2.3 times the images that are feed in YoloV5S so the two models have similar inference speeds.

Dataset implementations For the effectiveness of our models we performed a simple experiment. For our dataset, we divide the 1147 images of RoboWeedMap dataset into 3 sets, 689 images for training, 228 images for validation and 230 images for testing. To form our dataset each of the 2054×2456 pixels original images was cropped into several 640×640 pixels images. For our baseline, YoloV5S, these cropped images was down-scaled 2.3 times before the training scheme was started but our Nanodet Model no down-scaled was performed before training.

Training Scheme We trained our YoloV5S model with default hyperparameters. For our model we implement a semi-supervised training that a bigger, teacher, model was trained along side the original model to provide a better dynamic soft label assigning inspired from YoloX. Furthermore, we implement an Exponential Moving Average scheme of weights for updating the weights of our model, in order to increase training stability. The results of the evaluation are provided in Table 4.

Table 3: Nanodet and YoloV5S inference speed in FPS.

Model	Nanodet	Nanodet	YoloV5S	YoloV5S
Platform	Jetson TX2	Nvidia RTX3070	Nvidia RTX3070	Nvidia RTX3070
Input	2464×2080	2464 × 2080	2464×2080	1088×896 (/2.3)
FPS	30	547	109	486

Table 4: Evaluation results in test set for our proposed method and baseline. We report mean average precision at both 50 and 50-95 IoU levels. * indicates that the models where used with down-scaled inputs.

Class	all		poaceae		brassicaceae	
Measured mAP at IoU level	50	50-95	50	50-95	50	50-95
YoloV5S	0.726	0.474	0.653	0.388	0.8	0.561
YoloV5S*	0.656	0.371	0.573	0.292	0.739	0.45
YoloV5S*, not pretrained	0.653	0.386	0.566	0.298	0.741	0.474
Proposed	0.514	0.287	0.396	0.188	0.633	0.386

3 2D/3D object localization and tracking

3.1 Variational Voxel Pseudo Image Tracking

3.1.1 Introduction and objectives

3D single object tracking task (3DSOT) aims to track a single object of interest through time and combines challenges from both 3D object detection and 3D multiple object tracking. From the detection perspective, 3DSOT requires finding an accurate object position and orientation for each frame, and from the multiple object tracking perspective, the association process still has to be performed as the method should distinguish from the similarly-looking objects and the object of interest, especially when their trajectories lie close to each other.

There are different sensors that can be used for this task, including single or dual camera images, Radars and Lidars. Camera images are the cheapest, but they lack depth information which is critical for 3D perception and, while dual camera setups are trying to mimic human eyesight, this approach leads to worse results than the Lidar-based methods. Lidars usually operate at 10-20 Hz and create a point cloud for each frame, which consists of a set of 3D points, perceived my shooting a laser beam and waiting for a possible reflection to come back. The explicit 3D nature of Lidars makes it a default choice for 3DSOT, producing both high accuracy and speed of the models. Single object tracking can be performed by using correlation filters [8, 32], direct offset prediction with deep learning techniques [31] or with a Siamese approach which looks for a position with the highest similarity score between it and the template [15, 2, 46, 45, 69].

By estimating uncertainty in neural networks, one can produce better statistical models, use the estimated uncertainty to improve the model's perception quality and change the post-model behavior by analyzing the predicted uncertainty. Applications of uncertainty estimation include 3D Object Detection [17, 63, 64], 3D Object Tracking [113, 103], 3D Human Pose Tracking [12], and Steering Angle Prediction [57]. These methods improve the perception and control quality by incorporating uncertainty estimation, but most of them use either a single deterministic network or the Monte Carlo Dropout (MCD) [19] methods for estimating uncertainty, which provide much poorer quality of uncertainty compared to other methods [75].

There exist two main approaches to utilize uncertainty in neural networks, a statistical approach which improves the quality of the model, and a control approach which changes the decision of the system that uses the network's outputs to control the system. The statistical approach is easier to evaluate, as any dataset used for a desired task can be used in the same way to test the statistically better uncertainty-based method, but the control approach requires either a simulation environment, or a real controlled environment where the new control system can be tested on top of the improved perception method.

A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.3:

- [70] Oleksiienko, I., Nousi, P., Passalis, N., Tefas, A., and Iosifidis, A., "Variational Voxel Pseudo Image Tracking", *arxiv:2302.05914*, 2023.

3.1.2 Summary of the state of the art

Gawlikowski et al. [20] define four types of uncertainty estimation methods. Single Deterministic Networks [86, 113] use a single network and try to estimate its uncertainty by either adding branches that learn to output the model's uncertainty, or by analyzing the model's performance with external tools. Bayesian Neural Networks (BNNs) [7, 60] consider a distribution over weights of a model and sample multiple networks for each run, resulting in a set of predictions, the variance of which represents the uncertainty of the model. Ensemble Methods [74, 96] can also be viewed as special types of BNNs, which use a categorical distribution over weights and usually train each sample independently, resulting in a predefined set of networks. Test-Time Data Augmentation methods [102, 101, 37] change the input by applying different augmentations and analyze the variance in outputs of a single network, applied to these slightly different inputs. Variational Neural Networks [73, 72] are similar to BNNs, but instead of considering a distribution over weights, they place a Gaussian distribution over the outputs of each layer and estimate its mean and variance values by the corresponding sub-layers.

The most used approach for 3D SOT task is to deploy a Siamese network, which considers a pair of target and search regions and applies the same transformation to both these regions. The transformed regions are then compared by a cross-correlation module, which outputs the most likely position of the target region inside the search region, allowing to track the object of interest when applied for each frame. P2B [78], BAT [112], Point-Track-Transformer (PTT) [87, 35] and 3D-SiamRPN [15] use point-based Siamese networks since the small sizes of regions of interest allow processing points directly. 3D Siam-2D [109] uses one Siamese network in a 2D Birds-Eye-View (BEV) space to create fast object proposals and another Siamese network in 3D space to select the true object proposal and regress the bounding box. Voxel Pseudo Image Tracking (VPIT) [69] avoids processing points directly and follows the widely used in 3D detection approach of voxelizing the space to create 2D pseudo images, but instead of voxelizing the whole scene, only the search region is considered for processing, which makes it possible to do real-time tracking.

Utilization of uncertainty for tracking is still in an early stage of development. Bayesian YOLO [39] combines MCD and a single deterministic network to estimate epistemic and aleatoric uncertainties for 2D object detection. Feng et al. [17] use a Lidar-based 3D object detection method, and, similarly to Bayesian YOLO, deploy a partially MCD model to estimate the epistemic uncertainty and use a single deterministic network for aleatoric uncertainty. LazerNet [63] utilizes the predicted bounding box uncertainties during a non-maximum suppression process. This method is further improved by estimation of ground-truth uncertainties for training bounding boxes by calculating an Intersection over Union (IoU) of the bounding box and a convex hull of the corresponding point cloud.

Zhong et al. [113] perform 3D Multiple Object Tracking (MOT) by using a single deterministic network for estimating detection uncertainty and then providing these values to a Kalman filter [36] of the 3D tracking method. Uncertainty-Aware Siamese Tracking (UAST) [110] performs 2D single object tracking by using a single deterministic network and quantizing over the specific range of values and predicting the softmax score for each quantized value to estimate a distribution over outputs. The expectation of the estimated distribution is used as a final regression value.

To the best of our knowledge, there are no methods that utilize uncertainty for 3D

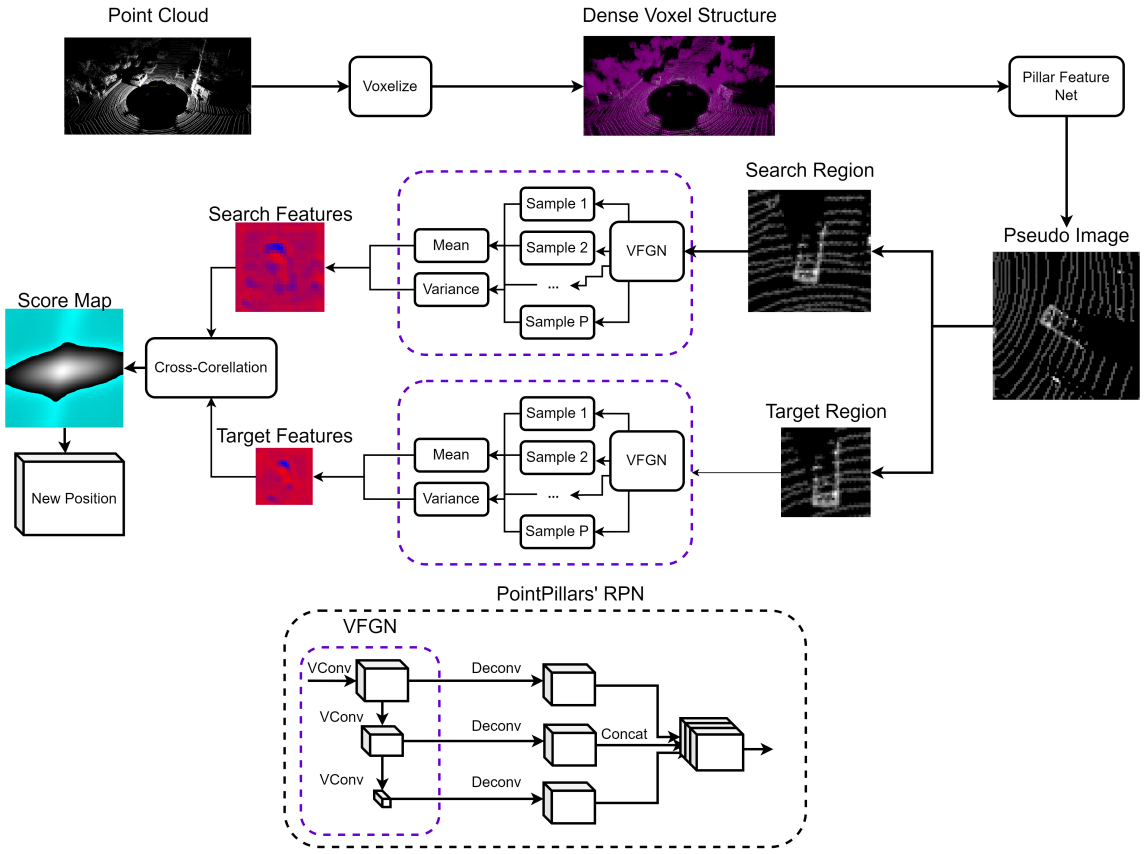


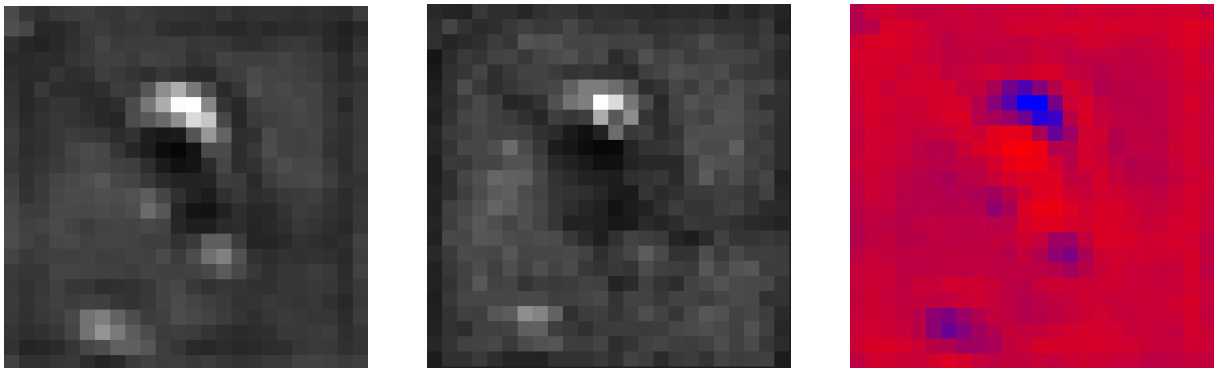
Figure 8: Computational structure of Variational Voxel Pseudo Image Tracking.

Single Object Tracking. Moreover, the estimation of uncertainty for related tasks is performed with statistically worse tools [75], such as single deterministic networks or MCD.

3.1.3 Work description

We select Voxel Pseudo Image Tracking (VPIT) [69] as a base model as it achieves the highest inference speed on the popular KITTI dataset [21] and suffers less speed loss when deployed on embedded devices, compared to other real-time methods [78, 87], which is caused by a specific architecture of CPU-GPU communication on the tested embedded devices, which is better utilized by VPIT. Voxel Pseudo Image Tracking uses PointPillars [43] as a backbone to create a pillar pseudo image and to process it with a Feature Generation Network (FGN), which is a convolutional subnetwork of the PointPillars' Region Proposal Network. After processing search and target regions with FGN, the generated features are compared with a convolutional cross-correlation module to create a pixel-level similarity map, where the highest value corresponds to the position of the target region inside the search region.

We create a Variational VPIT (VVPIT) model by replacing VPIT's FGN with a Variational Neural Network (VNN) [71, 72] version of it, i.e., VFGN. The structure of the proposed Variational VPIT model is present in Fig. 8. The input point cloud is voxelized and processed by a small Pillar Feature Network that generates features for each voxel and creates a 2D pseudo image where each pixel represents a corresponding



(a) Mean (b) Variance (c) Mean & certainty

Figure 9: An example of (a) mean, (b) variance and (c) mean and certainty features of a search region corresponding to a car point cloud. Darker colors in mean and variance images correspond to lower feature values. In the mean and certainty feature map, the blue color channel represents feature intensity, and the red color channel represents the corresponding certainty.

voxel. Pseudo images of target and search regions are processed with the same VFGN module that uses Variational Convolutional layers (VConv) to generate outputs of multiple network samples, the mean and variance of which are computed as final outputs. Different number of samples can be used during each step of training and inference. For each of the target and search regions, VFGN produces a set of outputs in the form $Y = \{y_i, i \in [1, \ldots, P]\}$ which correspond to the outputs of P sampled VFGN models, with $Y^s = \{y_i^s, i \in [1, \ldots, P]\}$ corresponding to the search region output set and $Y^t = \{y_i^t, i \in [1, \ldots, P]\}$ to the target region output set. For target and search regions, the number of samples P can be different, but for simplicity we use the same values. The means and variances of the outputs are computed as follows:

$$\begin{aligned} y_m^s &= \frac{1}{P}\sum_i^P y_i^s, \;\; y_v^s = \text{diag}\left(\frac{1}{P}\sum_i^P (y_i^s - y_m^s)(y_i^s - y_m^s)^T\right), \\ y_m^t &= \frac{1}{P}\sum_i^P y_i^t, \;\; y_v^t = \text{diag}\left(\frac{1}{P}\sum_i^P (y_i^t - y_m^t)(y_i^t - y_m^t)^T\right), \end{aligned} \tag{9}$$

where y_m^s, y_v^s and y_m^t, y_v^t are the mean and variance values of search and target output feature sets, respectively, and $\text{diag}(\cdot)$ is a function that returns the main diagonal of a matrix. Fig. 9 shows the mean and variance values of an VFGN output for one input example. The combined image has high certainty on red pixels, which are mostly background pixels. This is explained by the fact that it is easy for the model to find the background, and since there is no need to distinguish between different types of background, all samples agree on these features.

The proposed VVPIT can use the estimated uncertainty in different ways. By following the simplest way, we can ignore the uncertainties and process only mean values. In this case, it is still expected to achieve a higher model quality as the statistical qualities of the model are improved. The regular cross-correlation function $g(a, b)$ is used, which is defined as a 2D convolution $\text{conv2D}_{\omega=b}(a)$ with ω being the kernel weights. Since most 3D SOT methods compare features in a similarity-based way, we focus on similarity approaches to

utilize uncertainty instead of distance-based methods. We propose a double similarity-based process which treats mean and variance values as separate sets of features and uses the regular similarity function $g(a, b)$ on both sets separately. The final similarity value $\hat{g}_{\text{double}}$ is computed as a liner combination of two similarity scores as follows:

$$\hat{g}_{\text{double}}(y_m^s, y_m^t, y_v^s, y_v^t) = g(y_m^s, y_m^t) + \lambda g(y_v^s, y_v^t), \tag{10}$$

where λ is a variance weight hyperparameter. This approach is based on the idea that pixels with similar uncertainty vectors are likely to represent the same object.

Furthermore, we developed a different approach which tries to focus on more certain positions and penalize pixels which are uncertain. This is achieved by dividing each mean feature value during the convolutional process by the corresponding normalized variance score, as follows:

$$\begin{aligned} &\forall c,\ \ v_n^c(v) = (\rho - 1)\frac{v^c - \min(v^c)}{\max(v^c) - \min(v^c)} + 1, \\ &\forall p_x, \forall p_y,\ \ \hat{g}_{\text{pen}}(y_m^s, y_m^t, y_v^s, y_v^t)^{p_x,p_y} = \frac{2\tilde{y_m^s}^{\,p_x,p_y}\tilde{y_m^t}^{\,p_x,p_y}}{v_n(\tilde{y_v^s})^{p_x,p_y} + v_n(\tilde{y_v^t})^{p_x,p_y}}, \end{aligned} \tag{11}$$

where the $v_n(v)$ function is used to normalize the variance predictions by the channel-wise minimum and maximum values to be in $[1, \rho]$ range, with a hyperparameter ρ that defines how much the uncertain predictions are penalized, $v_n^c(v)$ implements the normalization procedure for a single channel c. For an input j, the notation $\tilde{j}$ denotes the tensor with convolutional patches of j, and j^{p_x,p_y} corresponds to the values of j at position (p_x, p_y).

Training of VVPIT is performed with the same protocol as the base model. We use a pre-trained variational PointPillars for 3D object detection to initialize VVPIT. After the initialization, the network is trained with a Binary Cross-Entropy (BCE) loss between the ground truth and the predicted score maps.

3.1.4 Performance evaluation

Following the standard protocol, we use a training set of the KITTI [21] tracking dataset for both training and testing. The tracks $[0, \ldots, 18]$ are used for training and validation, and tracks 19 and 20 are used to test the trained models. Performance of the model is computed with Precision and Success [107] metrics, which are based on the predicted and ground truth objects' center difference and 3D Intersection Over Union, respectively. The model is trained for $64,000$ steps with different number of training VFGN samples per step in $[1, \ldots, 20]$ range. For the double similarity uncertainty utilization method, we test the variance weight hyperparameter λ in the range of $\{0.1, \ldots, 1\}$ with step 0.1 and for the uncertainty penalization approach, the ρ value is tested for the range of $\{2, \ldots, 5\}$ with step 1. The best models are reported for each uncertainty estimation method.

Table 5 compares a regular VPIT models with different uncertainty utilization models of VVPIT with 20 samples. As expected, by discarding uncertainty, we achieve a better model than the original one, but by actually utilizing the estimated uncertainty, we can improve the performance further.

Table 5: Precision and Success evaluation of KITTI single object tracking experiments for VPIT and Variational VPIT (VVPIT) models.

Method	Uncertainty	Success	Precision
VPIT	-	50.49	64.53
VVPIT	averaging	51.97	66.69
VVPIT	double similarity	52.62	66.56
VVPIT	uncertainty penalization	**53.30**	**67.79**

3.2 Uncertainty-Aware AB3DMOT by Variational 3D Object Detection

3.2.1 Introduction and objectives

3D object detection (3D OD) task aims to detect objects in 3D world, providing relative 3D position and size of the objects of interest. 3D multiple object tracking (3D MOT) is often built upon 3D object detection by adding an association mechanism which assigns unique IDs to each detected objects. These ID values have to be consistent through different frames, meaning that the same object on different frames should be marked with the same ID and different objects have to be marked with different ID values.

Similarly to VVPIT presented in Section 3.1, this work is aimed to improve the quality of 3D multiple object tracking by utilizing high-quality uncertainty for a critical perception task.

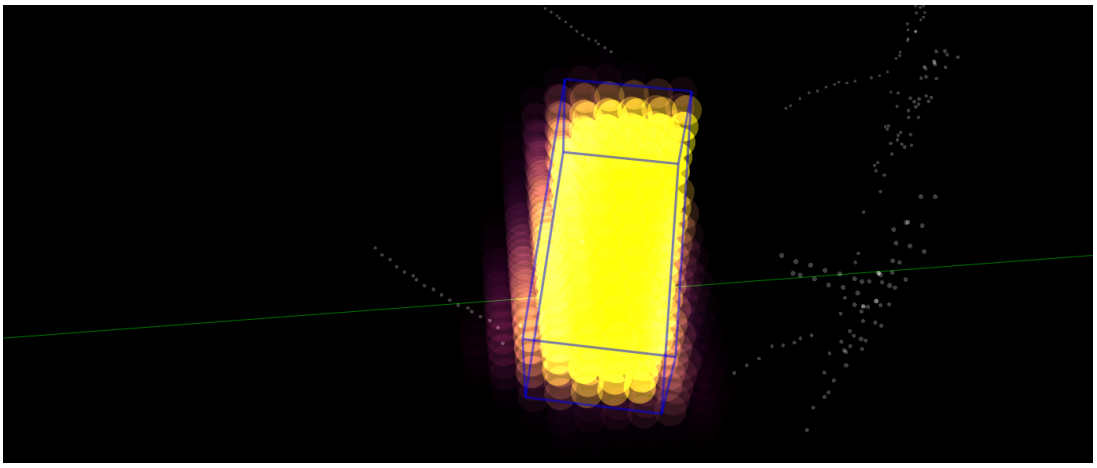
A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.4:

- [68] Oleksiienko, I. and Iosifidis, A., "Uncertainty-Aware AB3DMOT by Variational 3D Object Detection", *arxiv:2302.05923*, 2023.

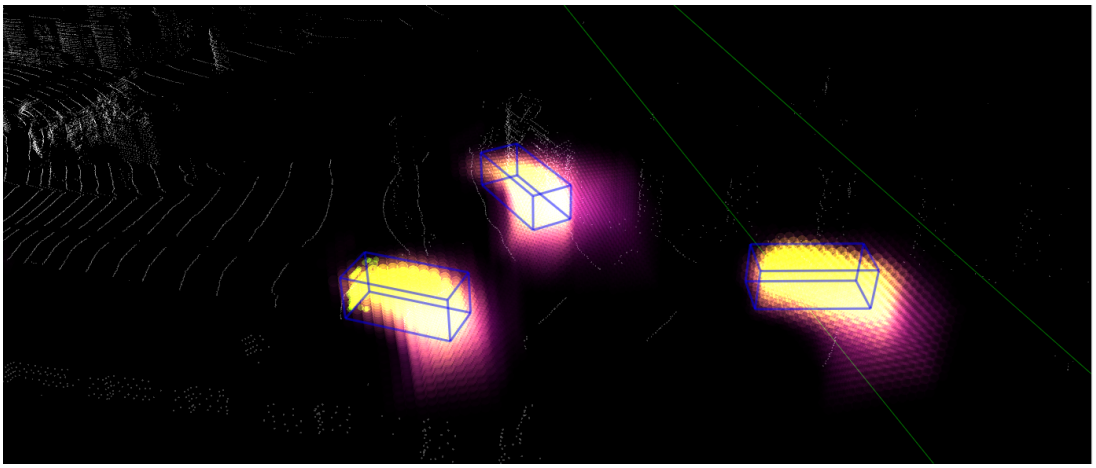
3.2.2 Summary of the state of the art

The best methods for 3D OD in combination of accuracy and inference speed use Lidar point clouds as inputs, but these point clouds cannot be processed by powerful CNNs, which leads to different structuring approaches, with voxelization being the most popular way to structure point clouds for processing with CNNs. VoxelNet [113] proposed to split a sub-scene of interest into an even 3D grid of cuboids called voxels. These voxels are processed by a small neural network to create voxel-level features, which are later processed by 3D and 2D convolutional subnetworks. PointPillars [43] improves this approach by considering pillars, which are voxels that take out the whole vertical space, resulting in a 2D grid, instead of 3D grid. The generated 2D pseudo image is processed by a faster 2D convolutional network, leading to lower inference times, compared to VoxelNet. TANet [54] uses a Triple Attention module to produce a better pillar pseudo image than TANet and then processes this image with a Coarse-to-Fine regression network. This method provides a trade-off between PointPillars' speed and the improved detection accuracy.

AB3DMOT [106] is a 3D version of SORT [5] for multiple object tracking, which performs object association based on already detected objects. Both these methods use a



(a) A detected 3D bounding box of a car and a corresponding uncertainty from a Variational 3D object detection model.



(b) 3D bounding boxes with modified uncertainties provided to the Kalman filter of the uncertainty-aware AB3DMOT tracker.

Figure 10: 3D bounding boxes with raw uncertainties obtained by the proposed Variational TANet (top), and the uncertainties provided to the Kalman filter of the proposed Uncertainty-aware AB3DMOT (bottom). The yellow color represents higher object probability, while the purple color represents lower probability of an object being present at that point. Blue boxes correspond to the average of the predictions.

2D or 3D Kalman Filter [36] to represent the inner state of the tracked objects and use a Hungarian algorithm [40] to associate between new detections and already tracked objects. In [113], the use of uncertainty is for 3D MOT is performed by training a voxel-based SECOND [108] model with an additional uncertainty regression branch. The predicted uncertainty is fed to a Kalman Filter of SORT to improve the tracking accuracy.

Estimation of uncertainty in neural networks can be performed in a single deterministic network approach [86, 113], in a Bayesian Neural Networks (BNNs) approach [7, 60, 74, 96], or in a test-time augmentation approach [102, 101, 37]. Variational Neural Networks [71, 72] are similar to BNNs but instead of considering a distribution over the model's weights, they consider a distribution over the outputs of the model, parametrized during run-time by the corresponding sub-layers.

3.2.3 Work description

Variational TANet TANet [54] is an anchor-based method [81], and therefore the outputs of the Coarse-to-Fine network have to be further post-processed to predict the final boxes. Each point of the predicted feature map represents a position in space and describes the possibility of having an object there, with a description of its relative size and position. To filter out the overlapping descriptions of the same objects from neighboring pixels, a Non-maximum suppression mechanism is applied.

We train a Variational TANet (VTANet) model by replacing its fully convolutional Coarse-to-Fine network with Variational Convolutional layers. The training is performed with the same loss function and training protocol, but we also define a number of samples used for each run of the network during training and inference.

We define two ways of estimating uncertainty for the inference of the Variational TANet model. The internal variance approach is a straightforward way to estimate the model's uncertainty by computing the variance in the outputs of the variational part of TANet and then propagating this variance through the post-processing steps of the network. This is achieved by treating the outputs of the Variational Coarse-to-Fine network as Gaussian random variables and computing the distribution after the post-processing operations. The voxel-wise outputs of the network are in the form of (x, y, z, w, l, h, r) vectors based on the corresponding anchor of the same form, where (x, y, z) is the relative position vector, (w, l, h) is the relative size vector and r is the relative rotation angle. Given a set of predictions $(x_i, y_i, z_i, w_i, l_i, h_i, r_i), i \in [1, S]$, where S is the number of samples, the mean and variance values $(m_x, m_y, m_z, m_w, m_l, m_h, m_r)$ and $(v_x, v_y, v_z, v_w, v_l, v_h, v_r)$, respectively, are computed and passed to the decoding algorithm. The decoding process takes an anchor $(x_a, y_a, z_a, w_a, l_a, h_a, r_a)$ and produces the decoded prediction $(\hat{x}, \hat{y}, \hat{z}, \hat{w}, \hat{l}, \hat{h}, \hat{r})$ as follows:

$$
\begin{aligned}
&d = \sqrt{l_a^2 + w_a^2}, \\
&(\hat{x}, \hat{y}) = (m_x, m_y)d + (x_a, y_a), \\
&(\hat{w}, \hat{l}, \hat{h}) = (e^{m_w} w_a, e^{m_l} l_a, e^{m_h} h_a), \\
&\hat{r} = m_r + r_a, \\
&\hat{z} = m_z h_a + z_a + h_a/2 - \hat{h}/2.
\end{aligned}
\tag{12}
$$

The corresponding variances $(v_{\hat{x}}, v_{\hat{y}}, v_{\hat{z}}, v_{\hat{w}}, v_{\hat{l}}, v_{\hat{h}}, v_{\hat{r}})$ are calculated as follows:

$$\begin{aligned}
&d = \sqrt{l_a^2 + w_a^2},\\
&\mathrm{v}_{\exp}(m, v) = e^{2m+2v} - e^{2m+v},\\
&(v_{\hat{x}}, v_{\hat{y}}) = (v_x, v_y)d^2,\\
&(v_{\hat{w}}, v_{\hat{l}}, v_{\hat{h}}) =\\
&\quad (\mathrm{v}_{\exp}(\hat{w}, v_w)w_a^2, \mathrm{v}_{\exp}(\hat{l}, v_l)l_a^2, \mathrm{v}_{\exp}(\hat{h}, v_h)h_a^2)\\
&\hat{r} = v_r,\\
&v_{\hat{z}} = v_z h_a^2,
\end{aligned} \tag{13}$$

where $\mathrm{v}_{\exp}(m, v)$ is the variance of the exponent of the Gaussian random variable.

The external covariance approach treats the model as a black box and only uses the post-processed outputs to estimate the uncertainty. This method is less intrusive, but requires some understanding of the nature of the outputs. The model is inferred S times to generate S sets of 3D bounding boxes, which are then combined into one set by association, similarly to a Hungarian algorithm. Consider a set of predictions $P^S = \{p_i \mid i \in [1, \ldots, S]\}$, where each element is a list of 3D bounding boxes with size K_i, predicted by the model, $p_i = \{b_k \mid k \in [1, \ldots, K_i]\}$. In most cases, K_i will be equal across i, but if there are boxes with high uncertainty, they may appear in some samples and disappear in other samples, leading to fluctuations in the final list size. In order to combine the lists, we define an association set $A(P)$ as a set of box groups $\{g_q\}$ where each group contains 3D bounding boxes from different model samples with the closest distances to each other, but no more than 1 meter. By iterating through all boxes for each sample and finding the smallest distance to the average position of each group, we can populate these groups sequentially. If the closest distance is higher than 1 meter, a new group will be assigned to this object:

$$\begin{aligned}
&\forall i \in [1..S]\, \forall k \in [1..K_i]\\
&\hat{G}(b_k) = \underset{g_q}{argmin}\ |\mathrm{avg}(\{b^{\mathrm{x,y,z}} \in g_q\}) - b_k^{\mathrm{x,y,z}}|\\
&G(b_k) = \begin{cases} \hat{G}(b_k), & \text{if } |\mathrm{avg}(\{b^{\mathrm{x,y,z}} \in \hat{G}(b_k)\}) - b_k^{\mathrm{x,y,z}}| \leq 1,\\ g_{\mathrm{new}}, & \text{otherwise,} \end{cases}\\
&g_q = \{b_k \mid G(b_k) = q,\ i \in [1..S],\ k \in [1..K_i]\},
\end{aligned} \tag{14}$$

where $b^{x,y,z}$ is a 3D position of the box b and $\mathrm{avg}(\cdot)$ is the averaging function. For each group g_q, we compute the mean bounding box and the corresponding covariance matrix in the form of a 7×7 matrix for position (x, y, z), size (w, h, l) and rotation α.

Initialization of Variational TANet by pretrained model Variational TANet can be trained from scratch the same way as the original model, but instead of doing so, we can take advantage of the already trained TANet model and use it for initialization of VTANet. The base idea lies in the representation of classical networks as Bayesian Neural Networks with a Dirac delta distribution over the weights, resulting in a single data point taking up the whole distributional weight and the similarities between this distribution and a Gaussian distribution with such small variance, that all the samples from this

distribution are closer to the mean than the smallest floating-point number that can be represented in the selected computational system, resulting in computationally identical numbers. By gradually increasing the variance, we can go from the pseudo Dirac delta function to an actual Gaussian distribution, which means that by initializing the VTANet model with a pretrained TANet model and setting such almost-zero variance values, we achieve computationally identical models, and by increasing the variance values these models will deviate further apart. To train an IVTANet model, we initialize mean values with a pretrained TANet weights and variance values with small non-zero values by either using constant initialization or Xavier uniform/normal initialization [23]. The initialized model is close to the pretrained TANet, but the values of variances are still big enough to contribute to the output and be trained.

Uncertainty-Aware AB3DMOT It is shown [113, 84] that the Kalman filter benefits from providing uncertainty estimations instead of assuming a unit-Gaussian measurement uncertainty. We follow this approach but modify the predicted uncertainties from the detector by using a linear transformation:

$$\hat{\Sigma} = \alpha I + \beta \Sigma, \tag{15}$$

where $\hat{\Sigma}$ is the uncertainty that serves as an input to the 3D Kalman filter, α is a hyperparameter that controls the contribution of the default uncertainty and β is a hyperparameter that controls the contribution of the detector's uncertainty to the final value. The goal of this transformation is to utilize the predicted uncertainties, but also provide a degree of freedom for the Kalman filter. By using $(\alpha = 1, \beta = 0)$ we can achieve the regular AB3DMOT performance and by using $(\alpha = 0, \beta = 1)$ we can use the predicted uncertainties unchanged. Fig. 10 illustrates the difference between the raw predicted uncertainties and the modified uncertainties.

3.2.4 Performance evaluation

We follow the standard training procedure for TANet, VTANet and IVTANet on KITTI [21] dataset, described in TANet [54]. VTANet models are trained with different number of training samples in range $[1, \ldots, 4]$, and every trained model is evaluated in combination with AB3DMOT with different number of samples from the same range. We train VTANet and IVTANet models with both internal and external uncertainty estimation procedures and evaluate models with different values of α and β hyperparameters for AB3DMOT with limits being $\alpha \in [0, 1]$ and $\beta \in \{0.1, 1, 5, 10, 50\}$. We compare the proposed models with that of the Voxel von-Mises method [113], which is the only method, to the best of our knowledge, to incorporate uncertainty estimation into 3D object tracking. This method uses SECOND [108] for 3D object detection and SORT [5] for objects associations.

Table 6 shows the performance of different models on the KITTI tracking dataset based on Multi Object Tracking Accuracy (MOTA), F1 score, and Mostly Lost (ML) metrics. The "Tracking Parameters" column represents the hyperparameters values that are used for this model, and the type of uncertainty estimation approach is represented in the "Uncertainty Method" column. The proposed VTANet method that is trained from scratch improves all tracking metrics compared to the original TANet and the Voxel von-Mises method [113], with the external covariance method providing better results than the internal variance approach. Voxel von-Mises method provides much worse results due

Table 6: KITTI tracking results.

Model	**Uncertainty method**	**Training Samples**	**Inference Samples**	**Tracking Parameters**	**MOTA%↑**	**F1%↑**	**ML%↓**
Voxel von-Mises [113]	deterministic	-	-	SORT [5]	-	55.10	30.60
TANet [54]	-	-	-	AB3DMOT [106]	68.71	85.69	8.58
IVTANet + UA-AB3DMOT	covar	2	4	$\alpha = 0.6$, $\beta = 5$	**72.30**	**87.34**	**7.74**
IVTANet + UA-AB3DMOT	covar	2	4	$\alpha = 0$, $\beta = 1$	72.13	87.26	7.74
IVTANet + UA-AB3DMOT	covar	2	4	$\alpha = 1$, $\beta = 0$	72.05	87.22	7.74
VTANet + UA-AB3DMOT	covar	3	3	$\alpha = 0.6$, $\beta = 5$	69.63	86.46	7.95
VTANet + UA-AB3DMOT	covar	3	3	$\alpha = 0$, $\beta = 1$	69.48	86.39	8.16
VTANet + UA-AB3DMOT	covar	3	4	$\alpha = 0.6$, $\beta = 5$	69.42	86.38	9.00
VTANet + UA-AB3DMOT	covar	3	3	$\alpha = 1$, $\beta = 0$	69.15	86.31	9.00
VTANet + UA-AB3DMOT	covar	3	4	$\alpha = 1$, $\beta = 0$	68.86	85.04	9.21
VTANet + UA-AB3DMOT	covar	2	4	$\alpha = 0.6$, $\beta = 5$	68.54	86.13	8.16
VTANet + UA-AB3DMOT	var	3	4	$\alpha = 0.5$, $\beta = 5$	68.46	85.91	7.95

to the use of inferior methods at each of the steps, including the less accurate 3D detector SECOND [108] the less suited for 3D object detection tracker SORT [5] and a deterministic approach for uncertainty estimation that can be improved by considering a Bayesian alternative. VTANet improves upon TANet, but by initializing it with a pretrained TANet and training IVTANet models, we achieve a much bigger improvement in the tracking metrics. The best models of IVTANet share the same hyperparameters values as the best models of TANet, which indicates the usefulness of these values separately from the selected model.

4 Deep SLAM and 3D scene reconstruction

4.1 CoVIO: Continual Learning for Visual-Inertial Odometry

4.1.1 Introduction and objectives

Accurately gauging a robot's movement via its onboard sensors is an essential prerequisite for various downstream tasks like localization and navigation. Instruments such as inertial measurement units (IMU) or inertial navigation systems (INS) offer the direct means to measure the robot's motion by assessing acceleration and GNSS data. Visual odometry (VO), which harnesses image data from single or stereo cameras, provides an alternative approach [18, 85, 61]. Similar to other tasks in computer vision, there has been a growing interest in learning-based VO. This approach is gaining traction due to its ability to learn high-level features, which can address challenges posed by textureless regions [95, 94] and issues caused by dynamic objects [4] where traditional handcrafted methods struggle. However, a limitation of learning-based VO lies in its inability to adapt to new environments, hindering its real-world deployment potential. Thus, a recent avenue of research, known as adaptive VO [59], has emerged. For instance, it explores the use of continual learning (CL) techniques to enhance VO during inference [97].

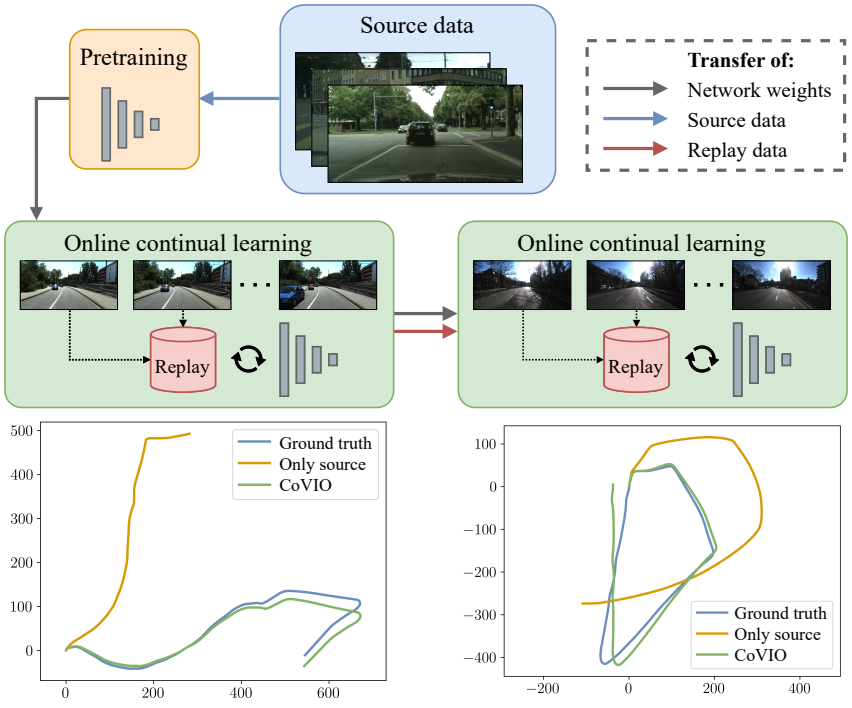


Figure 11: CoVIO for online continual learning of visual-inertial odometry. After pretraining on a source domain that is then discarded, CoVIO further updates the network weights during inference on a target domain.

Typically, learning-based VO employs monocular depth estimation as an auxiliary task [59, 49, 97]. It combines a PoseNet, which estimates camera motion between frames, with a DepthNet for single-image depth estimation [24], facilitating unsupervised joint training. This method's unsupervised nature allows for continuous training during inference. In addition to traditional domain adaptation [6], recent research on continual

SLAM [97] delves into sequential multi-domain scenarios (illustrated in Fig. 11). In this context, CL-SLAM introduces adaptive VO integrated with a graph-based SLAM backend.

Our work introduces a novel adaptive visual-inertial odometry estimation method called **CoVIO**, which directly addresses several drawbacks of **CL-SLAM**. Similar to the approach taken by Kuznietsov et al. [42], we operate in a source-free context, where experience replay excludes data from the source domain used for pretraining. Within a source-free framework, we introduce the following contributions in CoVIO. 1) We replace the dual-network structure with a single network that handles both domain adaptation and knowledge retention, leading to architectural simplification and reduced GPU memory utilization. 2) We utilize a compact replay buffer to optimize image diversity while considering storage limitations of embedded devices. 3) We propose an asynchronous version of CoVIO, which separates the core motion estimation from network updates, enabling continuous inference. 4) We extensively evaluate CoVIO on various datasets demonstrating its effectiveness in comparison to alternative visual odometry methods.

A summary of this work is provided hereafter. The corresponding paper is referenced below and can be found in Appendix 7.5:

- [98] N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada, "CoVIO: Online Continual Learning for Visual-Inertial Odometry", *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2463-2472, 2023.

4.1.2 Summary of state of the art

Continual Learning: Deep learning models are commonly trained for specific tasks but often fall short in real-world scenarios characterized by evolving data distributions and changing task objectives. Continual learning (CL) and lifelong learning [89] offer solutions to these challenges by enabling models to learn new tasks while retaining their previous knowledge. CL encompasses various strategies, including experience replay, regularization techniques, and architectural methods. Furthermore, online continual learning [58, 104, 100] updates models continuously during inference, adapting to the evolving data stream. In our work, we employ online CL, coupled with experience replay, to enhance visual-inertial odometry estimation in dynamic environments.

Adaptive Visual Odometry: Adaptive visual odometry (VO) and simultaneous localization and mapping (SLAM) have emerged as critical components to enhance the performance of robotic systems when navigating in unfamiliar environments. Commonly, learning-based VO involves monocular depth estimation as a proxy task, achieved through unsupervised training. Our approach builds upon CL-SLAM [97], a framework designed to mitigate catastrophic forgetting by utilizing a dual-network architecture. This architecture consists of an "expert" network, which facilitates online adaptation to new domains, and a "generalizer" network, which retains previously acquired knowledge through experience replay. Our work extends and enhances this approach, addressing its previous shortcomings.

4.1.3 Work description

Network Architecture and Pretraining: Our network architecture follows the standard approach for unsupervised monocular depth estimation. It consists of two distinct

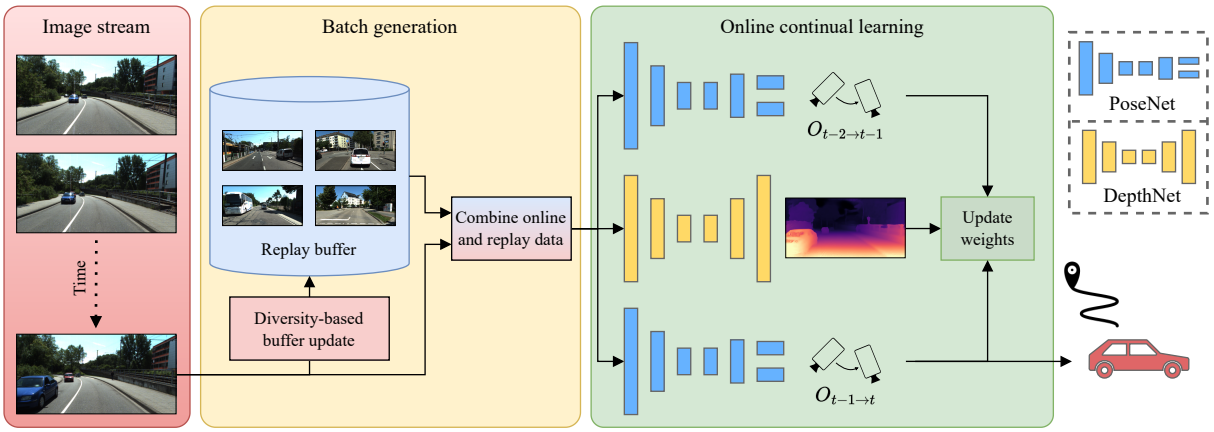


Figure 12: Our proposed CoVIO performs online continual learning on a stream of RGB images leveraging unsupervised monocular depth estimation as an auxiliary task.

networks, DepthNet and PoseNet, illustrated in Fig. 12. Similar to [97], we utilize Monodepth2 [24] to jointly predict a dense depth map $\mathbf{D}_{t-1}$ for the center image and compute the camera motion relative to both neighboring frames, denoted as $\mathbf{O}_{t-2\to t-1}$ and $\mathbf{O}_{t-1\to t}$, for an image triplet $\{\mathbf{I}_{t-2}, \mathbf{I}_{t-1}, \mathbf{I}_t\}$. In our network, we consider the latter as the visual odometry (VO) estimate. Specifically, we employ an implementation that incorporates two separate encoders based on the ResNet-18 architecture [30] for both DepthNet and PoseNet.

To initialize CoVIO, we perform unsupervised training on a source domain $\mathcal{S}$ in an offline manner. In detail, we exploit the photometric reprojection loss $\mathcal{L}_{pr}$ and the image smoothness loss $\mathcal{L}_{sm}$ to train the DepthNet and the PoseNet [24]. We additionally supervise the PoseNet with scalar velocity readings from the vehicle's IMU [27]. The applied velocity supervision term $\mathcal{L}_{vel}$ enforces metric scale-aware odometry estimates. Thus, our total loss is composed of three terms:

$$\mathcal{L} = \mathcal{L}_{pr} + \gamma\mathcal{L}_{sm} + \lambda\mathcal{L}_{vel}\,, \tag{16}$$

where γ and λ are weighting factors.

Online Continual Learning: After pretraining on a source domain $\mathcal{S}$, we use CoVIO to perform online continual learning on an unseen target domain $\mathcal{T}$. As illustrated in Fig. 12, each new RGB image triggers the following steps:

1. Create a data triplet comprising the new frame $\mathbf{I}_t$ and the two previous frames $\mathbf{I}_{t-1}$ and $\mathbf{I}_{t-2}$ along with the corresponding IMU readings.
2. Check whether this triplet should be added to the replay buffer using the proposed diversity-based update mechanism.
3. Sample from the replay buffer and combine the samples with the previously generated data triplet.
4. Estimate the depth map $\mathbf{D}_{t-1}$ and the camera motions $\mathbf{O}_{t-2\to t-1}$ and $\mathbf{O}_{t-1\to t}$.
5. Compute the loss defined in Eq. (16) and update the network weights via backpropagation.

6. Repeat steps (4) and (5) for c iterations.

7. Output $\mathbf{O}_{t-1\rightarrow t}$ as the odometry estimate.

Online Asynchronous CoVIO: We also propose an asynchronous variant of CoVIO to address true continuous inference on robotic platforms in a real-time capable setting. Since multiple update iterations c can result in a situation, in which the network update takes longer than the frame rate of the input camera stream, we also design a version that decouples the VO estimation from the CL updates. The *predictor* continuously generates VO estimates for each incoming image. The *learner* contains a copy of the network that is updated using the previously introduced online CL strategy but disregards images if the update step takes longer than the time until the next frame is available. Compared to caching frames, this strategy ensures that always the latest information is used to update the network. Then, after a given number of update cycles, the network weights are transferred from the *learner* to the *predictor*.

4.1.4 Performance evaluation

Throughout all experiments, we report the translation error t_{err} (in %) and the rotation error r_{err} (in °/m) as proposed by Geiger [22]. These metrics evaluate the error as a function of the trajectory length. To ensure a fair comparison with the base work CL-SLAM [97], we further utilize the set of network weights that is provided by the authors and was pretrained on the Cityscapes Dataset [11]. We also follow CL-SLAM and only consider new frames when the IMU measures a driven distance of at least 0.2 m. For the standard evaluation, we have used The KITTI Dataset [22]. For continual learning of new domains, we use images and ground truth poses of multiple sequences from the odometry benchmark and combine them with the respective IMU data from the raw dataset.

Table 7: Comparison of continual odometry estimation on the KITTI odometry benchmark.

Method	Seq. 04		Seq. 05		Seq. 06		Seq. 07		Seq. 10	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
ORB-SLAM [67]	0.62	0.11	2.51	0.25	7.80	0.35	1.53	0.35	2.96	0.52
Only target	10.72	1.69	34.55	11.88	15.20	5.62	12.77	6.80	55.27	9.50
DeepSLAM [48]	5.22	2.27	4.04	1.40	5.99	1.54	4.88	2.14	10.77	4.45
Only source	28.94	4.64	46.13	19.20	49.57	20.79	37.75	25.42	30.91	15.28
CL-SLAM [97]	4.37	**0.51**	4.30	1.01	2.53	0.63	**2.10**	**0.83**	11.18	1.74
CoVIO *(ours)*	**2.11**	0.53	**2.88**	**0.94**	**2.13**	**0.47**	3.19	1.26	**3.71**	**1.55**

Comparison of the translation and rotation errors of our CoVIO with baseline methods evaluated on the KITTI odometry benchmark. "Only target" and DeepSLAM are trained on sequences {00, 01, 02, 08, 09} without further adaptation. "Only source", CL-SLAM, and CoVIO are trained on Cityscapes. Both CL-SLAM and CoVIO perform online adaptation on the respective KITTI sequence.

4.2 Row Guidance

4.2.1 Introduction and method

It was investigated if it is possible to calculate the cross-track error of the robot based on the CropEye camera's physical position using either the crop's bounding box or the plant stem emergent zone (PSEZ). The cross-track error (XTE) is the difference between the calculated plan and the crop row's actual position.

Three analyses were conducted, the first one to get an idea if the method was possible, the second to investigate more deeply if the method was possible and the third to compare the performance of plant stem emergent zone to the crop's bounding box.

The robot's planned route and the actual route data for each executed plan is stored in the cloud. It is also displayed on the remote supervisor page so farmers can see if the robot's driving performance is acceptable. The CropEye camera is mounted perpendicular to the ground on the robot at -43 cm to the left of the center line of Robotti. As the mounting position doesn't change, it is possible to calculate the position of the crop row and compare it to the planned route and actual route. This difference is referred to as cross-track error (XTE) in all three analyses.

When the images are acquired by the robot, meta data is also acquired for each image and stored in the cloud. When the images are annotated, additional annotation data is added to the image, see Fig. 13.

```
## List of 7
##  $ UploadId      : int 1205
##  $ ImageId       : int 1123794
##  $ FileName      : chr "95dc87fb-7167-422a-b794-89a78edf506f.jpg"
##  $ AnnotationArea: num 0.8
##  $ AnnotationData:'data.frame':  169 obs. of  7 variables:
##   ..$ PlantId : int [1:169] -101 -101 -101 -101 -101 -101 -101 -101 -101 -100 ...
##   ..$ MaxX    : num [1:169] 1571 1188 31 428 243 ...
##   ..$ MaxY    : num [1:169] 1479 1807 1605 1607 1091 ...
##   ..$ MinX    : num [1:169] 1541 1135 1 381 233 ...
##   ..$ MinY    : num [1:169] 1451 1739 1531 1553 1078 ...
##   ..$ EPPOCode: chr [1:169] "PPPDD" "PPPDD" "PPPDD" "PPPDD" ...
##   ..$ Approved: logi [1:169] FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ UseForTraining: logi TRUE
##  $ Tags          :List of 1
##   ..$ : chr "PSEZ_Crop_image_Approved"
```

Figure 13: Example of meta data from a CropEye image after it has been imported and exported from the plant annotation program.

The images that were used to conduct the analyses were acquired from 3 different fields, with the names 4-1, 8-1 and 5-0, see Fig. 14 and Fig. 15. All fields were located in Denmark and all images were acquired during the 2022 farming season. The images were collected with the CropEye system while band spraying and interrow weeding. The crop was maize.

The method used in Analysis 1-3 was to, for each one-pixel column along the x axis, sum all pixels inside the bounding boxes. The column with the most pixels is the center of the crop row. Equations, see below, were used to calculate the XTE of the image.

$$XTE = (PlantRowPosition/I_{\mathrm{w}} - Center/X_{\mathrm{w}}) * I_{\mathrm{w}} \tag{17}$$

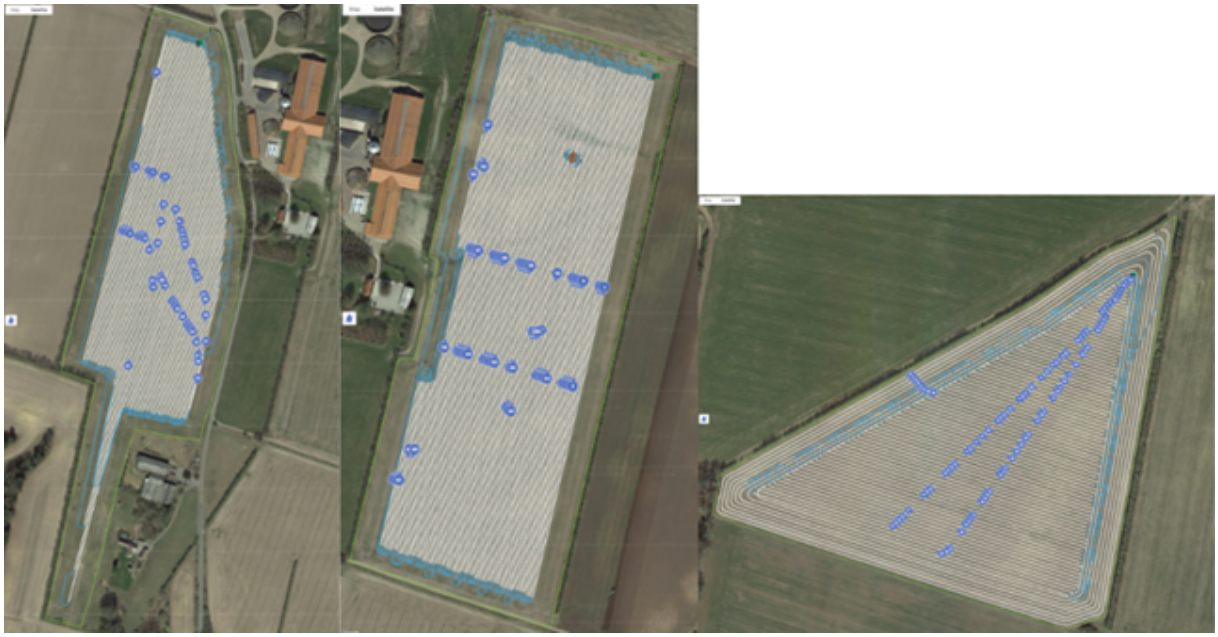


Figure 14: Left: Field 4-1. Center: field 8-1. Right: Field 5-0. White lines are the planned path for Robotti and the blue lines are turns. The blue icons are row numbers.

$$PlantRowPosition = (I_w/2 + CameraOffSet + [RowOffSet/2, -RowOffSet/2] \quad (18)$$

I_w = width of the image. In this case, I_w = 1 m based on measurements at physical robot.

X_w = number of pixels along the horizontal axis of the image. In this case, X_w = 2448 pixels.

Center = sum pixels in each 1 px wide column inside the bounding boxes. Choose the max. Center is the number of pixels along the x axis of the maximum value.

CameraOffSet = Distance from the center of the robot to the camera. In this case, *CameraOffSet* = -0.43 m.

RowOffSet = Distance between each row. In this case, *RowOffSet* = 0.75 m.

4.2.2 Analysis 1: Finding the cross-track error (XTE) of 5 images using the bounding box of the crop

An analysis was performed to find the cross-track error (XTE) of 5 images using the bounding box of the maize crop. The five images were selected from the same GPS location at different time intervals, see Fig. 16.

It can be observed in figure Fig. 16, that the position of the crop (maize, yellow bounding boxes), is stable. The robot is following the same plan, therefore, it is highly likely that the XTE of the robot can be calculated based on the position of the crop.

To find the center of the of the pixels of the crop (maize), the pixels are counted inside the maize bounding box horizontally in the vertical direction and is used to estimate a probability density function. The maximum of this function is assumed to be the center of the row in pixels, see Fig. 17.

The XTE appears to increase as the days after sowing increases and the crop is getting taller, see Tab. 8. This error is likely because the camera is not mounted center in the

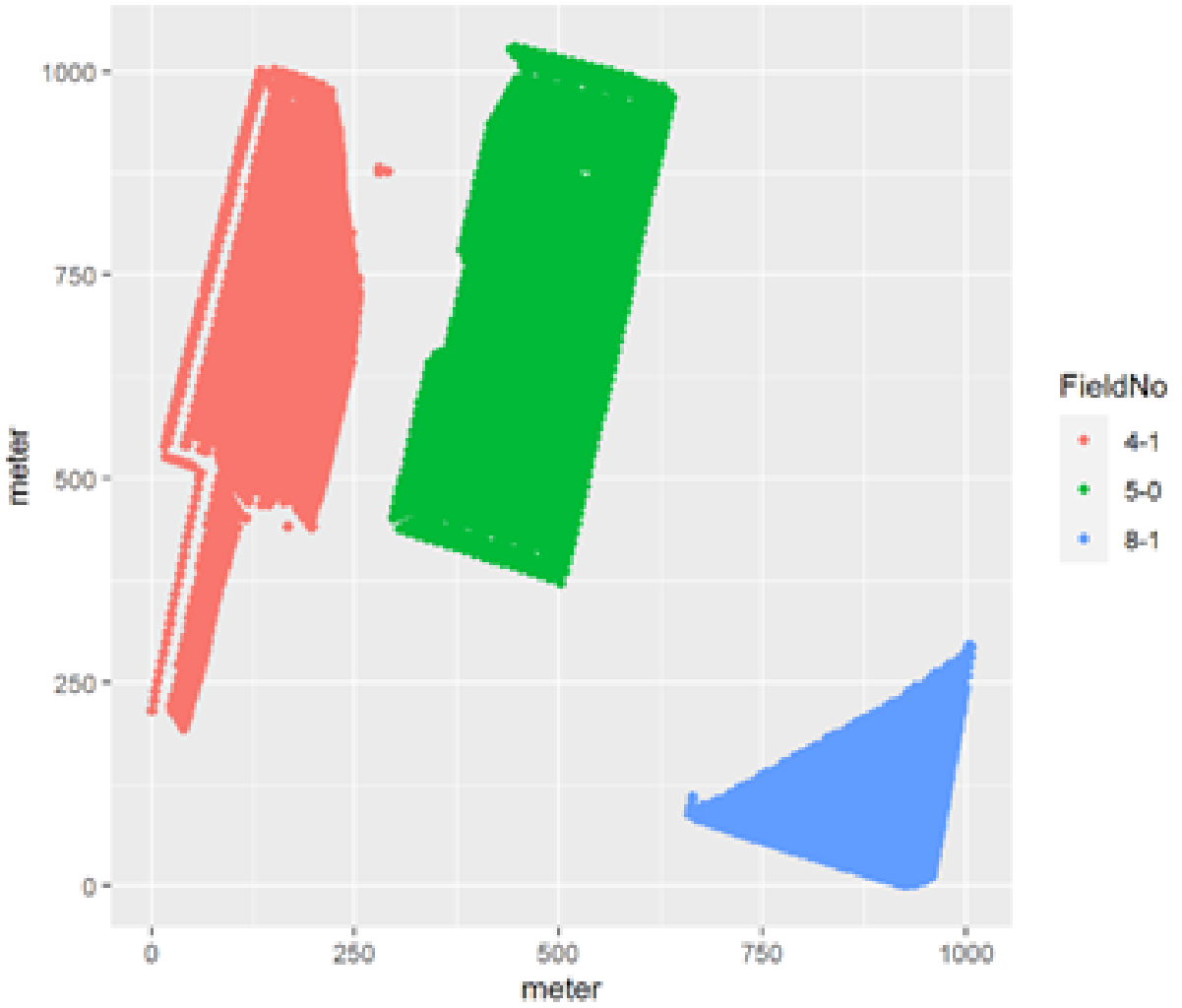


Figure 15: Sizes and shapes of the three fields, 4-1, 5-0, and 8-1 based on input from where the images were taken. Each dots represents one image.

maize row and as the crop grows, there will be an increase of the side view of the crop. Currently, the cameras are set at fixed points on the center boom of the robot and the farmers cannot adjust the camera's position. In past projects, when the farmer was required to input camera angle, height, etc for the system to work, there were often errors in the measurements. Therefore, it has been decided not to allow the farmer to change the camera's position to the center of the crop row.

As the height of the crop is unknown, it is difficult to counteract this error through calculations. However, with small crops, this is less of a problem. On the other side, it is possible to counteract the problem by training the Crop and Weed tool to find the plant stem emergence zone – the area where the crop emerges from the soil.

When navigating by RTK GNSS, the robot can achieve up to +/- 2 cm accuracy. The first analysis showed that by using the bounding boxes of the maize plants, an accuracy of +4.5 to -7.7 cm can be achieved. This is greater deviation than desired, however, by using the bounding boxes of the plant stem emergence zone, it is expected that the accuracy will increase. Therefore, it was evaluated that the method is acceptable and can be further investigated.

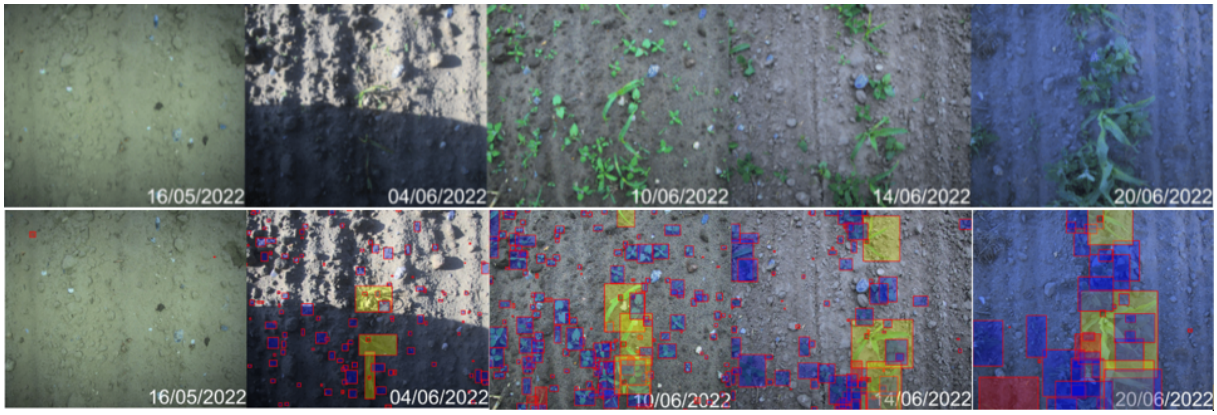


Figure 16: Top: A series of images with the same GPS location taken over time. Bottom: The same series with bounding boxes.

Table 8: Image no. and date correlates to images in Fig. 16. XTE is the difference between the green dotted line and the red dotted line, see Fig. 17.

Image no.	Days after sowing	Date	Cross-track error
1	12	16/05/2022	N/A
2	31	04/06/2022	4.512571
3	37	10/06/2022	-2.192426
4	41	14/06/2022	-7.678604
5	47	20/06/2022	-4.202712

4.2.3 Analysis 2: Analysis of XTE when evaluating 3 fields

An analysis was conducted on 3 different fields to investigate the mean cross-track error (XTE), 5th percentile (probability 0.05), and percentile 95 (probability 0.95) using the same calculation described in method section and section Analysis 1, see Tab. 9

When studying the numbers in Tab. 9, the mean cross-track error is less than 1 and in many cases, less than 0.5 cm, which is very promising. However, when studying the XTE max, the most frequent number varies 0 to -3.1 cm, indicating that the method may have flaws. Overall, it can also be seen that the q5 and q95 increases as the days after sowing increases. This error is due to the size of the crop – the larger the crop, the harder it is to use the bounding boxes to find the center of the row. This error can potentially be significantly reduced by training the model to find the crop emergence zone.

To improve the finding of the row, the Crop and Weed tool was updated with plant stem emergence zones (PSEZ) for both maize and sugar beets.

4.2.4 Analysis 3: Large quantity of images to see performance of XTE using bounding boxes vs. plant stem emergence zones

A third analysis was conducted using all images collected in the three fields 4-1, 5-1 and 8-1 using both bounding boxes and plant stem emergence zones to find the center of the crop row. A total of 29949 images were used for the maize bounding boxes (ZEAMX) and 23019 images were used for the plant stem emergence zone (PSEZ). The difference in images quantities is because for the maize bounding box, the image must include at

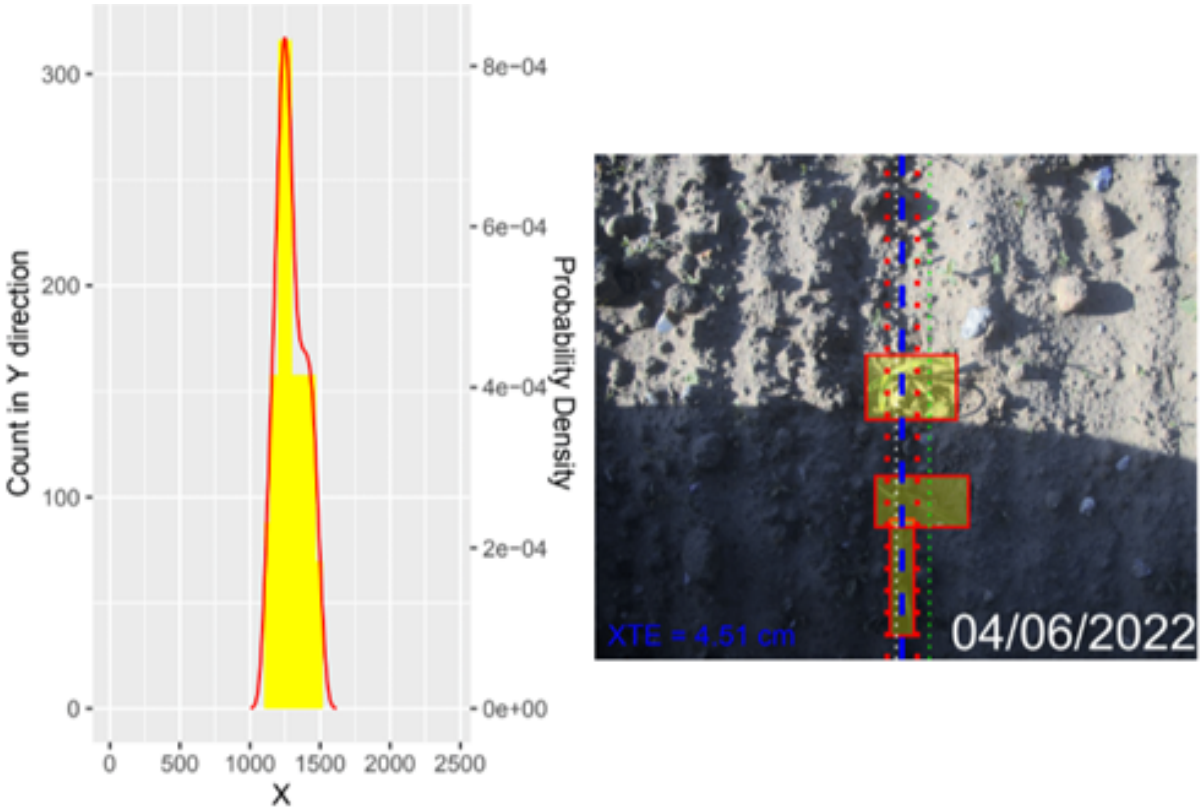


Figure 17: Left: Histogram showing bounding boxes pixel counts and derived probability density function used to estimate the crop row center. Right: The CropEye image only showing the crop bounding box'es. Calculated cross-track error (XTE) is 4,51 cm. The green dotted line is where the crop row center should have been assuming XTE=0. The dotted blue visualized line is the calculated center of the maize bounding boxes. The dotted red lines are the crop row center +/- 2.5 cm. The white dotted line indicates the center of the image.

least 1 maize bounding box and for the plant stem emergence zone, the image must also include 1 bounding box. As the robot does not center itself over the crops before it takes an image, it is possible for the camera to acquire images where it can find the maize plant but the plant stem emergence zone is not in the image.

When using maize bounding boxes to determine the placement of the robot, the cross-track error (XTE) was 0.61 cm, see Tab. 10. When using the plant stem emergence zone bounding boxes to determine the placement of the robot, the cross-track error (XTE) was 0.66 cm, see Tab. 10. However, when presenting the standard devation as a range, it is clear that the XTE for plant stem emergence zone is more accurate than the maize bounding boxes, see Fig. 18.

Table 9: Field 4-1, 5-0 and 8-1 for cross-track error (XTE) for images acquired while band spraying and interrow weeding. XTE mean is the average XTE in cm of the CropEye images where XTE could be estimated. XTE max is the number that occurred the most often. q5 is the 5th percentile (probs = 0.05) of XTE˙cm˙bbox after removing any missing values and rounded to 1 decimal place. q95 is the 95th percentile (probs = 0.95) of XTE˙cm˙bbox after removing any missing values and rounded to 1 decimal place.

Field no.	Days after sowing	No. of images	XTE mean (cm)	XTE max (cm)	XTE q5 (cm)	XTE q95 (cm)
4-1	28	1840	0.5	1.1	-8.2	8.3
4-1	37	1708	0.8	0.2	-8.9	10.1
4-1	44	1695	0.8	0	-9.2	10.7
5-0	31	2622	1	-0.5	-8.5	10.1
5-0	38	2345	0.4	0.8	-9.6	10.4
5-0	46	1625	0.7	-3.1	-11.2	12.5
5-0	47	830	0.3	-2.1	-12.1	12.4
5-0	48	167	-0.6	0.4	-10.8	9.4
8-1	31	847	0	0.6	-4.6	4.4
8-1	36	853	0.1	0.6	-5.6	5.7
8-1	40	764	0.4	-0.1	-5.4	6.9
8-1	47	852	0	1.1	-5.8	6.1

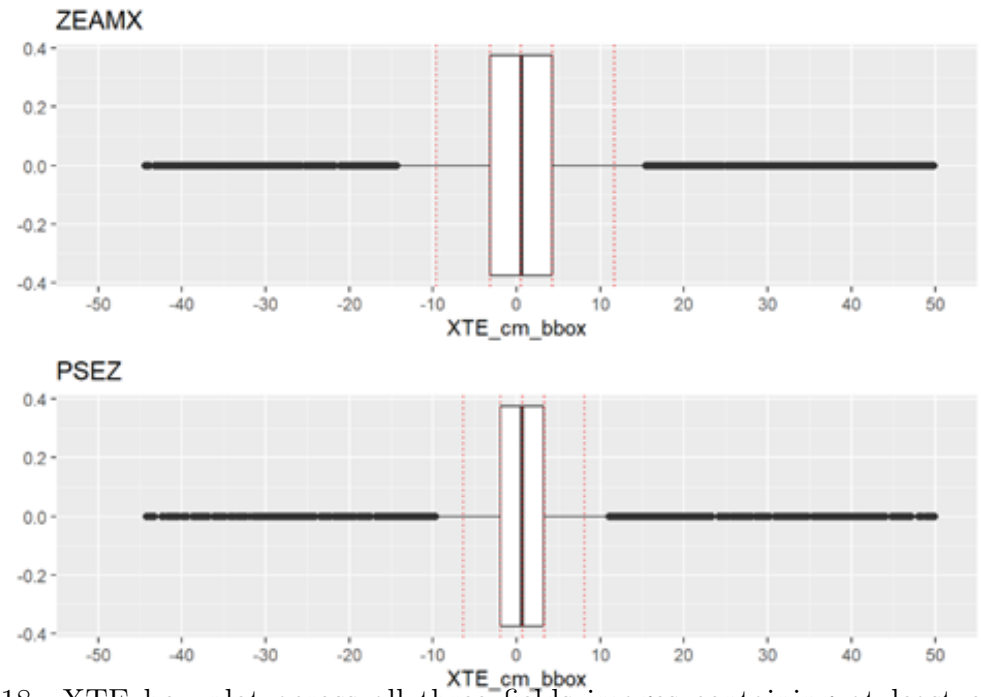


Figure 18: XTE box plot across all three fields images containing at least one maize bounding box (ZEAMX) or one plant stem emergence zone bounding box (PSEZ). The red vertical dotted lines illustrates 5, 25, 50, 75, 95 percent quantiles.

Tab. 10 shows the quantiles for the XTE for the maize bounding boxes (ZEAMX)

and for the plant stem emergence zones (PSEZ). For the XTE of maize bounding boxes (ZEAMX), the quantiles ranged from -9.6 to 11.7 cm. For the XTE of the plant stem emergence zones (PSEZ), the quantiles ranged from -6.3 to 8.1.

Table 10: Quantiles for XTE for the maize bounding boxes (ZEAMX) and for the plant stem emergence zones (PSEZ).

Label	**5% (cm)**	**25% (cm)**	**50% (cm)**	**75% (cm)**	**95% (cm)**
ZEAMX	-9.6	-3.2	0.5	4.3	11.7
PSEZ	-6.3	-1.9	0.7	3.3	8.1

Although the mean for the plant stem emergence zone is greater than the bounding boxes, the plant stem emergence zone is more accurate.

4.2.5 Robotti integration

The row guidance infrastructure on Robotti includes the GPU (Xavier), Robotti's CPU and the cloud (AWS), see Fig. 19. The method analyzed in the first section is in the process of being integrated into the GPU. The GPU will acquire the image, run the tool, output the bounding boxes, find the row, and output the XTE. The XTE is then transferred to the robot's CPU where the XTE is used to actuate the robot and the actual position of the robot, RTK GNSS, is recorded (completed). The recorded information is sent to the cloud, where the actual path is stored (completed). The actual path can then be visualized in Robotti portal and or analytics can be performed (completed).

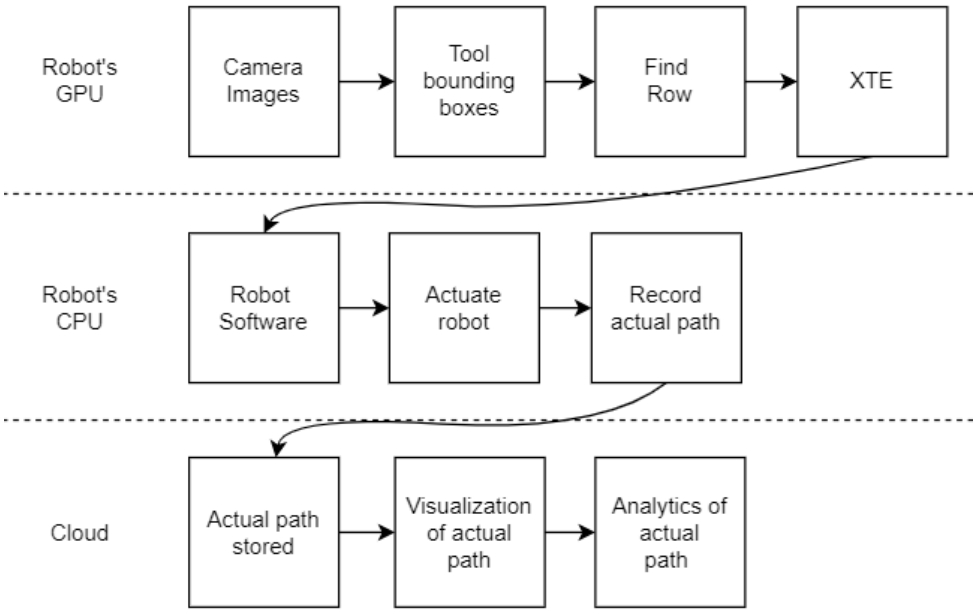


Figure 19: Final integration will be presented in D7.5. Tests and evaluation of the row guidance system will be presented in D8.4.

Final integration will be presented in D7.5. Tests and evaluation of the row guidance system will be presented in D8.4.

5 Sensor information fusion

5.1 Multimodal Feature Fusion Framework for Manipulation

A robot, while performing a specific task, often has access to multiple different data sources from its sensors. Such sensors include RGB cameras, lidars, force feedback sensors, microphones, infrared sensors and more. While taking advantage of such diverse data modalities is obviously beneficial, it is a complex task. TAU has previously proposed a general feature fusion framework suitable for these scenarios, with the primary application area being robotic arm manipulation (pick and place, insertion, etc.). The following requirements are set for the framework to adhere to:

- Support for input signals of modalities that are expected to be supported on the robotic arm in an industrial setting (RGB and depth camera feeds, force sensors, self-pose measurements).
- Inclusion of baseline feature fusion modules, supporting arbitrary combinations of modalities.
- Support for both task-specific surrogate objectives and input reconstruction objectives.
- Extensibility of the framework with respect to inputs/modalities, fusion approaches and outputs.
- Compactness of the intermediate and final representations, for faster computation.
- Support for robustness: minimizing the impact of damaged or missing data from individual modalities during inference.
- Capability to train on real and/or synthetic data.
- Integration with the simulation environment (Webots) for collecting synthetic data.

More detailed descriptions of the framework itself have been previously provided in Deliverables D4.1 and D4.2. To avoid redundancy, we do not replicate them here.

In the remaining period, we aim to focus on the evaluation of the aforementioned functionality, drawing on the assistance and expertise of other partners where appropriate. In addition to debugging and speed enhancements, we are working on extending the framework capabilities by adding new input decoders for new and existing modalities. We plan to continue experimenting with NAS for feature fusion optimization. Finally, time permitting, we plan to investigate the active perception possibilities for our multimodal setting and tasks.

6 Conclusions

In the final year of the project, the consortium carried out a series of activities following the overall objectives of the project. In the context of work package 4, we focused on objective O1 for providing a modular, open and non-proprietary toolkit for core robotic functionalities enabled by lightweight deep learning, and objective O2 for leveraging AI and cognition in robotics to go from perception to action.

ALU-FR contributed to O2a and O2c by proposing several methods for applying online continual learning to core robot perception tasks. In particular, they introduced CoDEPS (Section 2.1), the first method addressing online continual learning for joint depth estimation and panoptic segmentation. Furthermore, they presented CoVIO (Section 4.1) which is a continuation of their previous work on Continual SLAM but provides multiple improvements with respect to network design, pretraining requirements, and overall performance.

AUTH worked towards O1b by developing lightweight object detection algorithms and models for agricultural applications (Section 2.3), focusing on high resolution object detection and image understanding, while tackling the specific challenges that arise within this application field. Furthermore, AUTH continued working towards O2c on methods to tackle the shortcomings associated with the one-hot encoding by introducing a framework for learning soft label embeddings (Section 2.2).

AU worked towards O1a by introducing methods that estimate and utilize uncertainty for 3D perception tasks. Variational Voxel Pseudo Image Tracking (Section 3.1) introduces uncertainty estimation to Voxel Pseudo Image Tracking by replacing its convolutional neural networks with their Variational Neural Network (VNN) versions, and then utilizes the estimated uncertainties to improve the 3D single object tracking accuracy of the model. Furthermore, Variational TANet, in combination with Uncertainty-Aware AB3DMOT (Section 3.2), uses a VNN version of TANet to estimate the uncertainty of 3D detections and utilizes it in the Uncertainty-Aware AB3DMOT to improve the tracking process. The utilization of uncertainties leads to a direct improvement of the tracking, but can also be used as an input to decision-based methods.

TAU worked towards objective O2c by preparing benchmark evaluations for the multimodal feature fusion framework (Section 5), circumventing the recent data collection issues in both real and simulated domains, and allowing for further toolkit integration in the remaining project period.

AGI contributed to O2c by developing and analyzing a method to navigate and map the agriculture field robot based on the actual position of the plant rows. Using the output of the Crop and Weed tool, the crop rows are found along with the cross-track error of the crop rows. The agriculture field robot is then able to navigate, using the cross track error as a correction to its current position. The actual position is mapped using the robot's RTK GNSS (Section 4.2).

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019. 12

[2] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr. Fully-convolutional siamese networks for object tracking. *arXiv:1606.09549*, 2016. 21

[3] B. Bešić, N. Gosala, D. Cattaneo, and A. Valada. Unsupervised domain adaptation for LiDAR panoptic segmentation. *IEEE Robotics and Automation Letters*, 7(2):3404–3411, 2022. 8

[4] B. Bešić and A. Valada. Dynamic object removal and spatio-temporal RGB-D inpainting via geometry-aware adversarial learning. *IEEE Transactions on Intelligent Vehicles*, 7(2):170–185, 2022. 32

[5] A. Bewley, Z. Ge, L. Ott, F. T. Ramos, and B. Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468, 2016. 26, 30, 31

[6] B. Bešić, N. Gosala, D. Cattaneo, and A. Valada. Unsupervised domain adaptation for LiDAR panoptic segmentation. *IEEE Robotics and Automation Letters*, 7(2):3404–3411, 2022. 32

[7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight Uncertainty in Neural Networks. 2015. 22, 28

[8] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, 2010. 21

[9] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen. Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12472–12482, 2020. 9, 10, 11

[10] G. Cheng and J. H. Elder. VCSeg: Virtual camera adaptation for road segmentation. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1969–1978, 2022. 8

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016. 12, 35

[12] B. Daubney and X. Xie. Tracking 3d human pose with large root node uncertainty. In *CVPR*, pages 1321–1328, 2011. 21

[13] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: Keypoint triplets for object detection, 2019. 17

[14] N. El Gayar, F. Schwenker, and G. Palm. A study of the robustness of knn classifiers trained using soft labels. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pages 67–80. Springer, 2006. 13

[15] Z. Fang, S. Zhou, Y. Cui, and S. Scherer. 3d-siamrpn: An end-to-end learning method for real-time 3d single object tracking using raw point cloud. *IEEE Sensors Journal*, 21(4):4995–5011, 2021. 21, 22

[16] H. M. Fayek, M. Lech, and L. Cavedon. Modeling subjectiveness in emotion recognition with deep neural networks: Ensembles vs soft labels. In *2016 international joint conference on neural networks (IJCNN)*, pages 566–570. IEEE, 2016. 13

[17] D. Feng, L. Rosenbaum, and K. Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In *ITSC*, pages 3266–3273, 2018. 21, 22

[18] C. Fu, A. Carrio, and P. Campoy. Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. In *International Conference on Unmanned Aircraft Systems*, pages 957–962, 2015. 32

[19] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *JMLR Workshop and Conference Proceedings*, volume 48, pages 1050–1059, 2016. 21

[20] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. M. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, and X. X. Zhu. A survey of uncertainty in deep neural networks. *arxiv:2107.03342*, 2021. 22

[21] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 23, 25, 30

[22] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. 35

[23] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, volume 9, pages 249–256, 2010. 30

[24] C. Godard, O. M. Aodha, M. Firman, and G. Brostow. Digging into self-supervised monocular depth estimation. In *Int. Conf. on Computer Vision*, pages 3827–3837, 2019. 10, 32, 34

[25] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017. 9

[26] N. Gosala and A. Valada. Bird's-eye-view panoptic segmentation using monocular frontal view images. *IEEE Robotics and Automation Letters*, 7(2):1968–1975, 2022. 9

[27] V. Guizilini, R. Ambruș, S. Pillai, A. Raventos, and A. Gaidon. 3D packing for self-supervised monocular depth estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 34

[28] V. Guizilini, J. Li, R. Ambruș, and A. Gaidon. Geometric unsupervised domain adaptation for semantic segmentation. In *Int. Conf. on Computer Vision*, pages 8537–8547, 2021. 8, 9, 10

[29] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu. Ghostnet: More features from cheap operations, 2020. 17

[30] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 10, 14, 34

[31] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. *1604.01802*, 2016. 21

[32] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015. 21

[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 17

[34] J. Huang, D. Guan, A. Xiao, and S. Lu. Cross-view regularization for domain adaptive panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10133–10144, 2021. 9

[35] S. Jiayao, S. Zhou, Y. Cui, and Z. Fang. Real-time 3d single object tracking with transformer. *IEEE Transactions on Multimedia*, 2022. 22

[36] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. 22, 28

[37] I. Kandel and M. Castelli. Improving convolutional neural networks performance for image classification using test time augmentation: a case study using MURA dataset. *Health Inf. Sci. Syst.*, 9(1):33, 2021. 22, 28

[38] M. Klingner, M. Ayache, and T. Fingscheidt. Continual batchnorm adaptation (CBNA) for semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20899–20911, 2022. 10

[39] F. Kraus and K. Dietmayer. Uncertainty estimation in one-stage object detection. In *ITSC*, pages 53–60, 2019. 22

[40] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 28

[41] Y. Kuznietsov, M. Proesmans, and L. Van Gool. CoMoDA: Continuous monocular depth adaptation using past experiences. In *IEEE Winter Conference on Applications of Computer Vision*, 2021. 8

[42] Y. Kuznietsov, M. Proesmans, and L. Van Gool. Towards unsupervised online domain adaptation for semantic segmentation. In *Europ. Conf. on Computer Vision*, pages 261–271, 2022. 33

[43] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 23, 26

[44] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 14

[45] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan. Siamrpn++: Evolution of siamese visual tracking with very deep networks. *arXiv:1812.11703*, 2018. 21

[46] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8971–8980, 2018. 21

[47] F.-F. Li, M. Andreeto, M. Ranzato, and P. Perona. Caltech 101, Apr 2022. 14

[48] R. Li, S. Wang, and D. Gu. DeepSLAM: A robust monocular SLAM system with unsupervised deep learning. *IEEE Transactions on Industrial Electronics*, 68(4):3577–3587, 2021. 35

[49] S. Li, X. Wang, Y. Cao, F. Xue, Z. Yan, and H. Zha. Self-supervised deep visual odometry with online adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 32

[50] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection, 2020. 17, 18

[51] Y. Liao, J. Xie, and A. Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022. 12

[52] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018. 17

[53] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, pages 21–37, 2016. 16

[54] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai. TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. In *AAAI Conference on Artificial Intelligence*, 2020. 26, 28, 30, 31

[55] D. Lopez-Paz and M. A. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017. 9

[56] A. Lopez-Rodriguez and K. Mikolajczyk. DESC: Domain adaptation for depth estimation via semantic consistency. *Int. Journal of Computer Vision*, 131(3):752–771, Mar 2023. 8, 9

[57] A. Loquercio, M. Segu, and D. Scaramuzza. A general framework for uncertainty estimation in deep learning. *RA-L*, 5(2):3153–3160, 2020. 21

[58] M. Lunayach, J. Smith, and Z. Kira. Lifelong wandering: A realistic few-shot online continual learning setting. *arXiv preprint arXiv:2206.07932*, 2022. 33

[59] H. Luo, Y. Gao, Y. Wu, C. Liao, X. Yang, and K.-T. Cheng. Real-time dense monocular SLAM with online adapted depth prediction network. *IEEE Transactions on Multimedia*, 21(2):470–483, 2019. 32

[60] M. Magris and A. Iosifidis. Bayesian learning for neural networks: an algorithmic survey. *Artificial Intelligence Review*, 2023. 22, 28

[61] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers. *Journal on Field Robotics*, 24(3):169–186, 2007. 32

[62] G. Mattolin, L. Zanella, E. Ricci, and Y. Wang. ConfMix: Unsupervised domain adaptation for object detection via confidence-based mixing. In *IEEE Winter Conference on Applications of Computer Vision*, pages 423–433, January 2023. 9

[63] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *CVPR*, pages 12677–12686, 2019. 21, 22

[64] G. P. Meyer and N. Thakurdesai. Learning an uncertainty-aware object detector for autonomous driving. In *IROS*, pages 10521–10527, 2020. 21

[65] R. Mohan and A. Valada. Amodal panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20991–21000, 2022. 9

[66] R. Mohan and A. Valada. Perceiving the invisible: Proposal-free amodal panoptic segmentation. *IEEE Robotics and Automation Letters*, 7(4):9302–9309, 2022. 9

[67] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 35

[68] I. Oleksiienko and A. Iosifidis. Uncertainty-aware ab3dmot by variational 3d object detection. *arxiv:2302.05923*, 2023. 26

[69] I. Oleksiienko, P. Nousi, N. Passalis, A. Tefas, and A. Iosifidis. Vpit: Real-time embedded single object 3d tracking using voxel pseudo images. *arXiv:2206.02619*, 2022. 21, 22, 23

[70] I. Oleksiienko, P. Nousi, N. Passalis, A. Tefas, and A. Iosifidis. Variational voxel pseudo image tracking. *arxiv:2302.05914*, 2023. 21, 90

[71] I. Oleksiienko, D. T. Tran, and A. Iosifidis. Variational neural networks. *arxiv:2207.01524*, 2022. 23, 28

[72] I. Oleksiienko, D. T. Tran, and A. Iosifidis. Variational neural networks implementation in pytorch and jax. *Software Impacts*, 14:100431, 2022. 22, 23, 28

[73] I. Oleksiienko, D. T. Tran, and A. Iosifidis. Variational neural networks. *Procedia Computer Science*, 222C:104–113, 2023. 22

[74] I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. In *NeurIPS*, volume 31, pages 8626–8638, 2018. 22, 28

[75] I. Osband, Z. Wen, M. Asghari, M. Ibrahimi, X. Lu, and B. V. Roy. Epistemic Neural Networks. 2021. 21, 23

[76] N. Passalis and A. Tefas. Learning deep representations with probabilistic knowledge transfer, 2019. 17

[77] N. Passalis, M. Tzelepi, and A. Tefas. Heterogeneous knowledge distillation using information flow modeling, 2020. 16

[78] H. Qi, C. Feng, Z. Cao, F. Zhao, and Y. Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. *arXiv:2005.13888*, 2020. 22, 23

[79] S. Qiao, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen. ViP-DeepLab: Learning visual perception with depth-aware video panoptic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3996–4007, 2021. 9

[80] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 16

[81] S. Ren, K. He, R. Girshick, and J. Sun. In *NeurIPS*, volume 28, 2015. 28

[82] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 16

[83] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression, 2019. 17

[84] R. L. Russell and C. Reale. Multivariate uncertainty in deep learning. *TNNLS*, 33(12):7937–7943, 2022. 30

[85] T. Schöps, J. Engel, and D. Cremers. Semi-dense visual odometry for ar on a smartphone. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 145–150, 2014. 32

[86] M. Sensoy, L. Kaplan, and M. Kandemir. Evidential deep learning to quantify classification uncertainty. In *NeurIPS*, page 3183–3193, 2018. 22, 28

[87] J. Shan, S. Zhou, Z. Fang, and Y. Cui. Ptt: Point-track-transformer module for 3d single object tracking in point clouds. 2021. 22, 23

[88] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 17

[89] S. Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, volume 8, 1995. 33

[90] W. Tranheden, V. Olsson, J. Pinto, and L. Svensson. DACS: Domain adaptation via cross-domain mixed sampling. In *IEEE Winter Conference on Applications of Computer Vision*, pages 1378–1388, 2021. 9

[91] M. Tzelepi, N. Passalis, and A. Tefas. Online subclass knowledge distillation. *Expert Systems with Applications*, 181:115132, 2021. 13

[92] M. Tzelepi and A. Tefas. Efficient training of lightweight neural networks using online self-acquired knowledge distillation. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021. 13

[93] A. Valada, G. Oliveira, T. Brox, and W. Burgard. Towards robust semantic segmentation using deep fusion. In *Robotics: Science and Systems Workshop, Are the Sceptics Right*, 2016. 8

[94] A. Valada, N. Radwan, and W. Burgard. Deep auxiliary learning for visual localization and odometry. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6939–6946. IEEE, 2018. 32

[95] A. Valada, N. Radwan, and W. Burgard. Incorporating semantic and geometric priors in deep pose regression. In *Workshop on Learning and Inference in Robotics: Integrating Structure, Priors and Models at Robotics: Science and Systems (RSS)*, 2018. 32

[96] M. Valdenegro-Toro. Deep sub-ensembles for fast uncertainty estimation in image classification. *arxiv:1910.08168*, 2019. 22, 28

[97] N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada. Continual SLAM: Beyond lifelong simultaneous localization and mapping through continual learning. In A. Billard, T. Asfour, and O. Khatib, editors, *Robotics Research*, pages 19–35, Cham, 2023. Springer Nature Switzerland. 8, 10, 32, 33, 34, 35

[98] N. Vödisch, D. Cattaneo, W. Burgard, and A. Valada. Covio: Online continual learning for visual-inertial odometry. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2463–2472, June 2023. 33, 102

[99] N. Vyas, S. Saxena, and T. Voice. Learning soft labels via meta learning. *arXiv preprint arXiv:2009.09496*, 2020. 13

[100] N. Vödisch, K. Petek, W. Burgard, and A. Valada. Codeps: Online continual learning for depth estimation and panoptic segmentation. *Robotics: Science and Systems (RSS)*, July 2023. 8, 33, 53

[101] G. Wang, W. Li, M. Aertsen, J. Deprest, S. Ourselin, and T. Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019. 22, 28

[102] G. Wang, W. Li, S. Ourselin, and T. Vercauteren. Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation. In *BrainLes*, volume 11384, pages 61–72. Springer, 2018. 22, 28

[103] J. Wang, S. Ancha, Y.-T. Chen, and D. Held. Uncertainty-aware self-supervised 3d data association. In *RSJ*, pages 8125–8132, 2020. 21

[104] J. Wang, X. Wang, Y. Shang-Guan, and A. Gupta. Wanderlust: Online continual object detection in the real world. In *Int. Conf. on Computer Vision*, pages 10809–10818, 2021. 33

[105] Q. Wang, O. Fink, L. Van Gool, and D. Dai. Continual test-time domain adaptation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7201–7211, 2022. 10

[106] X. Weng, J. Wang, D. Held, and K. Kitani. AB3DMOT: A baseline for 3d multi-object tracking and new evaluation metrics. *CoRR*, abs/2008.08063, 2020. 26, 31

[107] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015. 25

[108] Y. Yan, Y. Mao, and B. Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 28, 30, 31

[109] J. Zarzar, S. Giancola, and B. Ghanem. Efficient bird eye view proposals for 3d siamese tracking. *arXiv:1903.10168*, 2020. 22

[110] D. Zhang, Y. Fu, and Z. Zheng. UAST: Uncertainty-aware siamese tracking. In *ICML*, volume 162, pages 26161–26175, 2022. 22

[111] Z. Zhang, S. Lathuilière, E. Ricci, N. Sebe, Y. Yan, and J. Yang. Online depth learning against forgetting in monocular videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4493–4502, 2020. 10

[112] C. Zheng, X. Yan, J. Gao, W. Zhao, W. Zhang, Z. Li, and S. Cui. Box-aware feature enhancement for single object tracking on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13199–13208, 2021. 22

[113] Y. Zhong, M. Zhu, and H. Peng. Uncertainty-aware voxel based 3d object detection and tracking with von-mises loss. *arXiv:2011.02553*, 2020. 21, 22, 26, 28, 30, 31

[114] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1851–1858, 2017. 9

7 Appendix

7.1 CoDEPS: Online Continual Learning for Depth Estimation and Panoptic Segmentation

The appended paper [100] follows.

CoDEPS: Online Continual Learning for Depth Estimation and Panoptic Segmentation

Niclas Vödisch[1*], Kürsat Petek[1*], Wolfram Burgard[2], and Abhinav Valada[1]
[1]University of Freiburg [2]University of Technology Nuremberg

***Abstract*—Operating a robot in the open world requires a high level of robustness with respect to previously unseen environments. Optimally, the robot is able to adapt by itself to new conditions without human supervision, e.g., automatically adjusting its perception system to changing lighting conditions. In this work, we address the task of continual learning for deep learning-based monocular depth estimation and panoptic segmentation in new environments in an online manner. We introduce CoDEPS to perform continual learning involving multiple real-world domains while mitigating catastrophic forgetting by leveraging experience replay. In particular, we propose a novel domain-mixing strategy to generate pseudo-labels to adapt panoptic segmentation. Furthermore, we explicitly address the limited storage capacity of robotic systems by leveraging sampling strategies for constructing a fixed-size replay buffer based on rare semantic class sampling and image diversity. We perform extensive evaluations of CoDEPS on various real-world datasets demonstrating that it successfully adapts to unseen environments without sacrificing performance on previous domains while achieving state-of-the-art results. The code of our work is publicly available at http://codeps.cs.uni-freiburg.de.**

I. Introduction

Deploying robots such as autonomous cars in urban scenarios requires a holistic understanding of the environment with a unified perception of semantics, instances, and depth. The joint solution of these tasks enables vision-based methods to generate a 3D semantic reconstruction of the scene, which can be leveraged for downstream applications such as localization or planning. While deep learning-based state-of-the-art approaches perform well when inference is done under similar conditions as used for training, their performance can drastically decrease when the new target domain differs from the source domain, e.g., due to environmental conditions [32], different sensor parameters [4, 7]. This domain gap poses a great challenge for robotic platforms that are deployed in the open world without prior knowledge about the target domain. Additionally, unlike the source domain where ground truth annotations are generally assumed to be known and can be used for the initial training, such supervision is not applicable to the target domain due to the absence of labels, rendering classical domain adaptation methods unsuitable. Unsupervised domain adaptation attempts to overcome these limitations. However, the vast majority of proposed approaches focuses on sim-to-real domain adaptation mostly in an offline manner [13, 24], i.e., a directed knowledge transfer without the need to avoid catastrophic forgetting and with access to abundant target annotations. Additionally, such works might not consider limitations on a robotic platform, e.g., available compute hardware and limited storage capacity [18, 33].

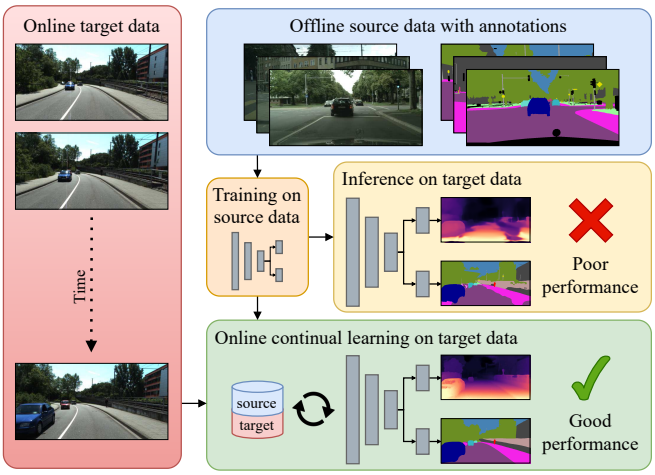


Fig. 1. Neural networks often perform poorly when deployed on a target domain that differs from the source domain used for training. To close this domain gap, we propose to continuously adapt the network by exploiting online target images. To mitigate catastrophic forgetting and enhance generalizability, we leverage a fixed-size replay buffer allowing the method to revisit data from both the source and target domains.

In this work, we use online continual learning to address these challenges for depth estimation and panoptic segmentation in a multi-task setup. As shown in Fig. 1, we leverage images from an onboard camera to perform online continual learning enhancing performance during inference time. While a naive approach would result in overfitting to the current scene, our method CoDEPS mitigates forgetting by leveraging experience replay of both source data and previously seen target images. We combine a classical replay buffer with generative replay in the form of a novel cross-domain mixing strategy allowing us to exploit supervised ideas also for unlabeled target data. We explicitly address the aforementioned hardware limitations by using only a single GPU and restricting the replay buffer to a fixed size. We demonstrate that CoDEPS successfully improves on new target domains without sacrificing performance on previous domains.

The main contributions of this work are as follows:

1) We introduce CoDEPS, the first online continual learning approach for joint monocular depth estimation and panoptic segmentation.
2) We propose a novel cross-domain mixing strategy to adapt panoptic segmentation to unlabeled target data.

*Equal contribution.

3) To address the storage restrictions of robotic platforms, we leverage a fixed-size replay buffer based on rare class sampling and image diversity.
4) We extensively evaluate CoDEPS and compare it to other methods in challenging real-to-real settings.
5) We release our code and the trained models at http://codeps.cs.uni-freiburg.de.

II. Related Work

In this section, we provide an overview of monocular depth estimation, panoptic segmentation, and unsupervised domain adaptation including continual learning.

Monocular Depth Estimation: Monocular depth estimation is the task of predicting a dense depth map from a single RGB image. While supervised approaches exploit measurements from range sensors to supervise the network predictions [30], unsupervised methods leverage geometric cues from temporal context [10, 38]. Most of the research on unsupervised learning tackles the limitations of the so-called photometric loss function that is usually employed for unsupervised depth learning, e.g., dynamic object handling [3, 5, 20], occlusion [11], and abrupt illumination changes [36]. In this work, we leverage Monodepth2 [11] for unsupervised depth learning and employ it similarly to Guizilini et al. [13] for the purpose of domain adaptation.

Panoptic Segmentation: Panoptic segmentation unifies the two tasks of semantic and instance segmentation by fusing the respective targets into a joint output. Furthermore, semantic classes are grouped into "stuff" classes, e.g., *road* or *building*, and "thing" classes, e.g., *car* or *pedestrian*. In particular, the goal of vision-based panoptic segmentation is to assign a semantic class to every pixel of an image and an additional instance label to each object belonging to the "thing" classes. Panoptic segmentation networks usually comprise a joint encoder and separate decoders for each subtask, whose outputs are subsequently merged by a panoptic fusion module. Existing works can be categorized into bottom-up [6, 28] and top-down [12, 27] approaches. Whereas bottom-up methods detect instances in a proposal-free manner from the semantic prediction, top-down methods include an additional proposal generation step. Contradictions to the semantic predictions are then resolved during post-processing. In this work, we build upon the bottom-up Panoptic-Deeplab [6] with changes to the semantic head according to Guizilini et al. [13].

Unsupervised Domain Adaptation: Domain adaptation aims to bridge the domain gap between a source domain $\mathcal{S}$ used for training and a target domain $\mathcal{T}$ used for inference to mitigate a loss in performance. An important aspect is whether the performance on the source domain must be maintained, linking domain adaptation to continual learning (CL) [23], where the objective of a task or the task itself can change over time. A CL system has to adapt to the new target objective while retaining the knowledge to solve the previous task(s), i.e., avoiding catastrophic forgetting. Ideally, the CL system can further achieve positive forward transfer, i.e., improve on future yet untrained tasks given the current task. In many real-world scenarios ground truth annotations for the target domain are not available, thus requiring unsupervised domain adaptation (UDA) methodology. Offline UDA assumes that abundant target data is accessible. However, in order to guarantee the continuous operation of a robot in new domains, UDA approaches have to work online without previous target data collection.

Offline UDA can leverage both annotated source data and abundant unlabeled target data, enabling learning a given task from $\mathcal{S}$ while simultaneously adapting the network to $\mathcal{T}$. For depth estimation, DESC [24] adapts from a synthetic source domain containing RGB images and ground truth depth to a real-world target domain by performing source-to-target style transfer and using a consistency loss between depth predictions from RGB and semantic maps. GUDA [13] tackles UDA for semantic segmentation using depth estimation as a proxy task. A shared encoder with task-specific heads for depth estimation and semantic segmentation is trained via source supervision. Simultaneously, data from $\mathcal{T}$ is used to update the encoder and depth head in an unsupervised manner. Due to the refined weights of the encoder, the semantic predictions on $\mathcal{T}$ improve as well. Another common approach for adapting semantic segmentation is cross-domain sampling enabling partial supervision on $\mathcal{T}$. DACS [31] mixes images from $\mathcal{S}$ and $\mathcal{T}$ by copying the pixels of a source image to a target image based on the semantic labels [29]. The semantic prediction of the target image is updated with ground truth source labels for the same set of pixels. The network is then jointly trained on annotated source data and the pseudo-labeled mixing data. Recently, ConfMix [25] proposed a simple yet effective mixing strategy for object detection, where a target image is divided into rectangular image regions. The region with the most confident predictions is then copied onto a source image and the respective ground truth annotations. Finally, Huang et al. [16] propose a UDA method for panoptic segmentation by regularizing complementary features from semantic and instance segmentation. In this work, we extend the aforementioned mixing strategies to instance-based sampling and explicitly address differing camera parameters.

During online UDA, samples from $\mathcal{T}$ can only be accessed in a consecutive manner resembling the image stream of a camera. Typically, a network is trained offline via supervision on $\mathcal{S}$ and then adapted to $\mathcal{T}$ during inference time. Such a setup rises two main challenges: first, incoming target samples originate from highly similar scenes and thus drastically reduce the diversity; second, this similarity of consecutive samples leads to a strong overfitting of the model to the scene [37]. Initial works for online UDA focused on depth learning [18, 37] and visual odometry [21, 33], for which unsupervised training schemes are already well established. Whereas Zhang et al. [37] propose novel network modules that are adapted via a meta-learning paradigm to mitigate forgetting, CoMoDA [18] employs a common CL strategy, i.e., experience replay to combine the online target sample with previously seen samples. Continual SLAM [33] also uses un-

supervised depth estimation as a proxy task to enhance visual odometry during inference time. Additionally, it demonstrates that incorporating samples from $\mathcal{S}$ and previous target domains $\mathcal{T}_i$ prevents catastrophic forgetting when revisiting domains. Similar settings involving multiple target domains, which are hence closely related to classical CL, are also addressed for semantic segmentation. CBNA [17] mixes statistics from $\mathcal{S}$ and $\mathcal{T}$ to update the batch normalization layers and showcases the efficacy of the approach on continually visited target domains. CoTTA [35] adapts the entire network without using source data but self-supervision. To tackle error accumulation, it uses an exponential moving average filter and student-teacher consistency when updating the network weights. Using depth estimation as a proxy task, Kuznietsov et al. [19] extend GUDA [13] to online UDA with experience replay and confidence regularization on the semantic predictions. To the best of our knowledge, we propose the first approach for online continual UDA for joint depth estimation and panoptic segmentation.

III. Technical Approach

The setting investigated in this work consists of two steps. First, we train a neural network on the source domain $\mathcal{S}$ partly using ground truth supervision. Second, to close the gap between domains, we continuously adapt the network during inference time on the target domain $\mathcal{T}$ using a replay buffer and unsupervised training strategies.

A. Network Architecture and Source Domain Pretraining

In this section, we detail the network architecture and loss functions that we employ during the pretraining stage on the source domain.

Architecture: We build our network following a common multi-task design scheme, i.e., using a single backbone followed by task-specific heads. A high-level overview of the network architecture is shown in Fig. 2. In detail, we use a ResNet-101 [14] as the shared encoder for all three tasks including depth prediction, semantic segmentation, and instance segmentation. The depth head follows the design of Monodepth2 [11] comprising five consecutive convolutional layers with skip connections to the encoder. Additionally, we include a separate PoseNet consisting of a ResNet-18 encoder and a four-layer CNN to estimate the camera motion between two image frames. For panoptic segmentation, we follow the bottom-up method Panoptic-Deeplab [6], leveraging separate heads for semantic segmentation and instance segmentation, and slightly modify the semantic head [13]. Specifically, the instance head consists of two sub-heads to predict the center of an object and to associate each pixel of an image to the corresponding object or the background. Finally, a panoptic fusion module [6] assigns a semantic label to the class-agnostic instance predictions using majority voting over the semantic predictions of all pixels within an instance.

Source Domain Pretraining: During the initial training phase on the source domain, we assume to have access to image sequences as well as ground truth panoptic segmentation annotations. In the following, we briefly describe the respective loss functions that we employ for training the three task-specific heads.

We train the depth estimation head using the common methodology of unsupervised training based on the photometric error [11]. In particular, we leverage an image triplet $\{\mathbf{I_{t_0}}, \mathbf{I_{t_1}}, \mathbf{I_{t_2}}\}$ to predict depth $\mathbf{D_{t_1}}$ and camera motion $\mathbf{M_{t_0 \to t_1}}$ and $\mathbf{M_{t_1 \to t_2}}$. Afterwards, we compute the photometric error loss $\mathcal{L}^d_{pe}$ as a weighted sum of the reprojection loss $\mathcal{L}^d_{pr}$ and the image smoothness loss $\mathcal{L}^d_{sm}$:

$$\mathcal{L}^d_{pe} = \lambda_{pr}\mathcal{L}^d_{pr} + \lambda_{sm}\mathcal{L}^d_{sm}. \tag{1}$$

We train the semantic segmentation head in a supervised manner using the bootstrapped cross-entropy loss with hard pixel mining $\mathcal{L}^{sem}_{bce}$ following Panoptic-Deeplab [6].

For training the instance segmentation head, we adopt the MSE loss $\mathcal{L}^{ins}_{center}$ for the center head and the L1 loss $\mathcal{L}^{ins}_{offset}$ for the offset head. The total loss to supervise instance segmentation is then computed as a weighted sum:

$$\mathcal{L}^{ins}_{co} = \lambda_{center}\mathcal{L}^{ins}_{center} + \lambda_{offset}\mathcal{L}^{ins}_{offset}. \tag{2}$$

B. Online Adaptation

After the described network has been trained on the source domain $\mathcal{S}$ using the aforementioned losses, we aim to adapt it to the target domain $\mathcal{T}$ in a continuous manner. That is, unlike other works, data from the target domain is revealed frame by frame resembling the online stream of an onboard camera. As depicted in Fig. 2, every adaptation iteration consists of the following steps:

1) Construct an update batch by combining online and replay data.
2) Generate pseudo-labels using the proposed cross-domain mixing strategy.
3) Perform backpropagation to update the network weights.
4) Update the replay buffer.

In this section, we first detail the structure of the utilized replay buffer and then propose adaptation schemes for both depth estimation and panoptic segmentation.

Replay Buffer and Batch Generation: Upon receiving a new image taken by the robot's onboard camera, we construct a batch that is used to perform backpropagation on the network weights. In detail, a batch $\mathbf{b_t}$ consists of the current online image $\mathbf{I_t} \in \mathcal{T}$, previously received target images $\mathbf{I}_{\mathcal{T}_\mathrm{i}} \in \mathbf{B}_\mathcal{T}$, and fully annotated source samples $\mathbf{I}_{\mathcal{S}_\mathrm{i}} \in \mathbf{B}_\mathcal{S}$. Here, $\mathbf{B}_\mathcal{T} \subseteq \mathcal{T}$ and $\mathbf{B}_\mathcal{S} \subseteq \mathcal{S}$ denote the respective replay buffers. Formally*,

$$\mathbf{b_t} = \{\mathbf{I_t}, \mathbf{I}_{\mathcal{T}_0}, \mathbf{I}_{\mathcal{T}_1}, \dots, \mathbf{I}_{\mathcal{S}_0}, \mathbf{I}_{\mathcal{S}_1}, \dots\}. \tag{3}$$

By revisiting target images from the past, we increase the diversity in the loss signal on the target domain and hence mitigate overfitting to the current scene. This further accounts for situations in which the current online image suffers from visual

*To improve readability, we omit in the notation that each image sample includes its two previous frames enabling unsupervised depth estimation.

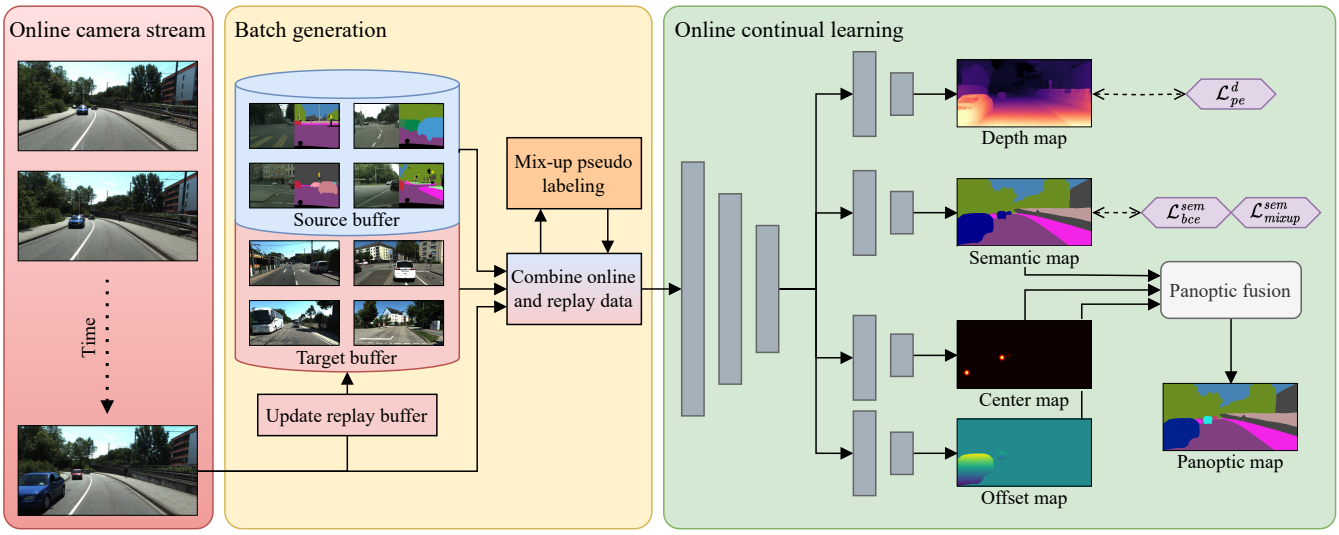


Fig. 2. Overview of our proposed CoDEPS. Unlabeled RGB images from an online camera stream are combined with samples from a replay buffer comprising both annotated source samples and previously seen target images. Cross-domain mixing enables pseudo-supervision on the target domain. The network weights are then updated via backpropagation using the constructed data batch. The additional PoseNet required for unsupervised monocular depth estimation is omitted in this visualization.

artifacts, e.g., overexposure. Similarly, revisiting samples from the source domain addresses the problem of catastrophic forgetting by ensuring that previously acquired knowledge can be preserved. Additionally, the annotations of the source samples enable pseudo-supervision on the target domain by exploiting cross-domain mixing strategies. For both the target and the source replay, we randomly draw multiple samples from the respective replay buffer and apply augmentation to stabilize the loss. In particular, we perform RGB histogram matching of the source images to the online target image, and all available source samples have to be selected once before repetition is allowed to ensure diverse source supervision.

Similar to previous works [1, 34], we explicitly consider limitations on the size of the replay buffer to closely resemble the deployment on a robotic platform, where disk storage is an important factor. This poses two questions: First, how to sample from $\mathcal{S}$ to construct the fixed source buffer $\mathbf{B}_\mathcal{S}$ that is prebuilt offline and, second, how to update the dynamic target buffer $\mathbf{B}_\mathcal{T}$ during deployment? To construct $\mathbf{B}_\mathcal{S}$, we propose a refined version of rare class sampling (RCS) [15]. The frequency f_c of each class $c \in \mathcal{C}$ is calculated based on the number of pixels with class c:

$$f_c = \frac{\sum_{\mathbf{I} \in \mathcal{S}} \sum_{p}^{H \times W} \mathbb{1}_c(p_{c'})}{|\mathcal{S}| \cdot H \cdot W}, \tag{4}$$

where H and W denote the height and width of the images in $\mathcal{S}$ and $p_{c'} \in \mathbf{I}$ refers to a pixel with class c'. The indicator function is 1 if c' equals c and 0 otherwise. The probability of sampling a class is then given by

$$P(c) = \frac{e^{(1-f_c)/T}}{\sum_{c' \in \mathcal{C}} e^{(1-f_{c'})/T}}, \tag{5}$$

with temperature T controlling the smoothness of the distribution, i.e., a smaller T assigns a higher probability to rare classes. In detail, we first sample a class $c \sim P$ and then retrieve all candidate images containing pixels with class c. Instead of taking a random image from these candidates, we sample according to the number of pixels with class c. We repeat both steps $|\mathbf{B}_\mathcal{S}|$ times without selecting the same image more than once. Using RCS ensures that $\mathbf{B}_\mathcal{S}$ contains sufficiently many images with rare classes such that the performance on these classes will further improve during adaptation.

Since $\mathcal{T}$ does not contain annotations and, particularly in the beginning, predictions are not reliable, we cannot use RCS for updating $\mathbf{B}_\mathcal{T}$. Instead, we invert the common methodology of loop closure detection for visual SLAM [33], i.e., the image $\mathbf{I_t}$ is only added to $\mathbf{B}_\mathcal{T}$ if its cosine similarity with respect to all samples within the buffer is below a threshold.

$$\text{sim}_{\cos}(\mathbf{I_t}) = \max_{\mathbf{I}_{\mathcal{T}_i} \in \mathbf{B}_\mathcal{T}} \cos\left(\text{feat}(\mathbf{I_t}), \text{feat}(\mathbf{I}_{\mathcal{T}_i})\right), \tag{6}$$

where $\text{feat}(\cdot)$ refers to the image features extracted from the final layer of the shared encoder, which is not adapted. If $\mathbf{B}_\mathcal{T}$ is completely filled, we remove the following image to maximize image diversity:

$$\arg\max_{\mathbf{I}_{\mathcal{T}_i} \in \mathbf{B}_\mathcal{T}} \sum_{\mathbf{I}_{\mathcal{T}_j} \in \mathbf{B}_\mathcal{T}} \cos\left(\text{feat}(\mathbf{I}_{\mathcal{T}_i}), \text{feat}(\mathbf{I}_{\mathcal{T}_j})\right) \tag{7}$$

Depth Adaptation: To adapt the monocular depth estimation head along with the PoseNet, we exploit the fact that the photometric error loss (Eq. 1) does not require ground truth annotations. Hence, we can directly transfer it to the implemented continual adaptation. In particular, we compute $\mathcal{L}^d_{pe}$ for the constructed batch $\mathbf{b_t}$ and average the loss such that each sample contributes by the same amount:

$$\mathcal{L}^d_{pe}(\mathbf{b_t}) = \frac{\mathcal{L}^d_{pe}(\mathbf{I_t}) + \sum_i \mathcal{L}^d_{pe}(\mathbf{I}_{\mathcal{T}_i}) + \sum_j \mathcal{L}^d_{pe}(\mathbf{I}_{\mathcal{S}_j})}{|\mathbf{b_t}|}. \tag{8}$$

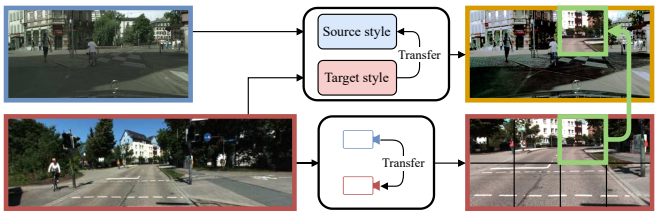


Fig. 3. Our proposed cross-domain mixing strategy first transfers the image style from the target to the source sample. Then it augments the target image to match the appearance of the source camera. Finally, a random image patch is copied from the target to the source image. The source annotations are retained and completed by the network's estimate on the copied image patch. The result serves as pseudo-label, combining self-iterative learning with ground truth supervision.

Furthermore, if the predicted camera motion is below a threshold, i.e., the robot is presumably not moving, we do not compute the $\mathcal{L}_{pe}^{d}(\mathbf{I_t})$ and subtract 1 from the denominator to avoid adding a bias to the current scene.

Panoptic Adaptation: As described in Sec. III-A, panoptic segmentation is the fused output of a semantic head and an instance head. We observe that the decrease in performance on samples from unseen domains can mostly be attributed to the semantic head, while instance predictions remain stable. Cross-domain mixing strategies allow leveraging ideas from supervised training to an unsupervised setting, where ground truth annotations are unknown. In CoDEPS, we bootstrap annotated source samples and high-confident target predictions to artificially generate pseudo-labels for the target samples in an online fashion to supervise the semantic head. Similar to depth adaptation, we continue to compute $\mathcal{L}_{bce}^{sem}$ on $\{\mathbf{I}_{\mathcal{S}_0}, \mathbf{I}_{\mathcal{S}_1}, \dots\}$ to mitigate forgetting, and freeze the instance head.

We further design a mixing strategy combining pixels of images from both $\mathcal{S}$ and $\mathcal{T}$, that considers multiple factors, which are unique to the online continual learning scenario: (1) the robust pretraining on a dedicated source dataset, which may result in significant performance degradation on the target dataset if the pre-trained weights are strongly adapted; (2) the existence of different cameras leading to significant changes in the field-of-view, geometric appearance of objects, resolution, and aspect ratio of the images; and (3) the continuously evolving visual appearance of street scenes during adaptation. To address these challenges, our cross-domain mixing approach employs a three-step method to generate the adaptation signal. First, we perform style transfer from the target image $\mathbf{I}_{\mathcal{T}_\mathrm{i}}$ to the source image $\mathbf{I}_{\mathcal{S}_\mathrm{j}}$ by aligning their pixel value histograms, as depicted in Fig. 3. This allows supervision with ground truth labels on images that are of similar visual appearance as the target image. Second, we apply a geometric transformation on $\mathbf{I}_{\mathcal{T}_\mathrm{i}}$ based on the camera intrinsics of the source and target domains denoted by $\mathbf{K}_\mathcal{S}$ and $\mathbf{K}_\mathcal{T}$, respectively. To this end, we assume a constant depth distribution over $\mathbf{I}_{\mathcal{S}_\mathrm{j}}$, lift the pixel values into Euclidean space via inverse camera projection, and project the lifted points back into the camera view of $\mathbf{I}_{\mathcal{T}_\mathrm{i}}$ as follows:

$$\mathbf{I}'_\mathcal{T}(\mathbf{p_s}) = \mathbf{I}_\mathcal{T}\langle \mathbf{K}_\mathcal{T}\mathbf{K}_\mathcal{S}^{-1}\mathbf{p_s}\rangle, \tag{9}$$

where $\langle\cdot\rangle$ denotes the bilinear sampling operator and $\mathbf{p_s}$ is a pixel coordinate in the source image. Equation 9 results in an adapted target image $\mathbf{I}'_\mathcal{T}$ with an adjusted field of view, resolution, and a geometric appearance of the scene similar to that of $\mathbf{I}_\mathcal{S}$. The final step in the process involves separating $\mathbf{I}'_\mathcal{T}$ into multiple segments and randomly selecting one of them to be inserted into the style-transferred source image, see Fig. 3. To avoid providing a flawed supervision signal caused by geometrically unrealistic images, we only insert a single patch. Similarly, the ground truth labels of the pixels from $\mathbf{I}_\mathcal{S}$ are retained, and the semantic labels estimated by the network are used to label the inserted patch after intrinsics transformation. The generated image is then fed into the network and training is performed using the cross-entropy loss and the generated pseudo-labels of the mixed image. To mitigate the decline in performance commonly associated with self-iterative training on predicted pseudo-labels, often resulting in class collapse, we utilize an exponentially moving average (EMA) filter for updating the network weights. In detail, we create a duplicate with network weights w_EMA of the initial model with weights w and use this so-called EMA model to generate the semantic predictions. During continual learning, the weights w are updated via backpropagation on $\mathbf{b_t}$. Then, the EMA model is updated as follows:

$$w_\mathrm{EMA} \leftarrow \alpha \cdot w_\mathrm{EMA} + (1-\alpha)\cdot w, \tag{10}$$

where α denotes the contribution of the EMA model.

IV. Experimental Evaluation

In the following sections, we provide further details on the pretraining step and the datasets that we evaluate on. We present extensive experimental results on the efficiency and efficacy of our proposed approach and include ablation studies on important design choices. Finally, we expand the experimental setup to multi-domain adaptation closely resembling classical continual learning settings.

We follow the evaluation protocol of Zhang et al. [37]. In detail, we compute the evaluation metrics on the frame of the current timestamp before using the same frame to perform backpropagation and update the model weights. Once $70\,\%$ of a sequence is processed, we calculate the average of the accumulated metrics. Additionally, we report the scores on the remaining $30\,\%$ of the same sequence without further weight updates to analyze the performance of the adapted model. In the tables, we refer to these types of evaluation by protocol 1 and protocol 2, respectively. We further denote the respective parts of a sequence by *adapt* and *eval*. Unlike Zhang et al. [37], we define our task in the context of continual learning. To measure knowledge retention and hence mitigate catastrophic forgetting, we introduce protocol 3 as evaluating the adapted model on the *val* split of the source dataset.

A. Datasets

To simulate data from a variety of domains, we employ our method on three datasets, namely Cityscapes [8], KITTI-360 [22], and SemKITTI-DVPS [2]. In particular, we utilize Cityscapes for pre-training and sequences of both KITTI-360

TABLE I
EFFICACY OF THE NETWORK

Method	Dataset	mIoU ↑	RMSE ↓	Abs Rel ↓	δ_1 ↑
GUDA	KITTI	—	4.42	0.11	0.88
CoDEPS		62.8	3.52	0.09	0.90
GUDA	Cityscapes	72.9	—	—	—
CoDEPS		72.9	10.16	0.19	0.78

Our utilized network is able to reproduce the performance of the baseline method GUDA [13] for both semantic segmentation (mIoU) and depth estimation (RMSE, Abs Rel, δ_1). The performance of GUDA is reported by the authors. To evaluate CoDEPS on KITTI, we use sequence 08 *eval* of SemKITTI-DVPS.

and SemKITTI-DVPS for adaptation. In the supplementary video we further provide qualitative results on in-house data recorded with our robotic platform.

Cityscapes: The Cityscapes Dataset [8] is a large-scale autonomous driving dataset that was recorded in 50 cities in Germany and bordering regions. It includes RGB images, panoptic annotations, and vehicle metadata. In this work, we utilize the fine panoptic labels to train the semantic and instance heads in a supervised manner. Additionally, we leverage the sequence image data of the left camera to train the depth prediction in an unsupervised fashion. Finally, we compute the depth error metrics using the provided disparity maps.

KITTI-360: The KITTI-360 Dataset [22] is a relatively recently released public dataset for the domain of autonomous driving, which was recorded in the city of Karlsruhe, Germany. It includes both 2D and 3D panoptic annotations for RGB images and LiDAR data. In this work, we predominantly utilize the RGB images to simulate an online image stream of an onboard camera. In particular, we use these images to adapt our network in a self-supervised manner. To compute evaluation metrics, we compare our predictions with the ground truth measurements and annotations of the dataset.

SemKITTI-DVPS: The SemKITTI-DVPS [30] is based on the odometry benchmark of the KITTI Dataset [9], which was recorded in Karlsruhe, Germany. We utilize the RGB images to simulate an onboard camera and to adapt our network to the new domain. Furthermore, we compute depth metrics based on the provided projected LiDAR points and the semantic/panoptic metrics using the extension SemanticKITTI [2].

Semantic Labels: As the aforementioned datasets use different labeling policies for the semantic annotations, we use the 19 classes of Cityscapes as the reference definition and remap classes of the other datasets. However, certain classes do not exist in the adaptation datasets (*wall*, *traffic light*, *bus*, *train*). For consistency across the datasets, we merge *wall* with *building* and remove the other three classes. Additionally, we merge *motorcycle* and *bicycle* into *two-wheeler* to increase the number of annotated pixels. Consequently, we consider nine "stuff" classes and five "thing" classes, listed in Table V. Note that *sky* is not included in SemKITTI-DVPS due to using LiDAR annotations and hence excluded in the evaluation on this dataset.

B. Pretraining Protocol

The initial state of the network weights before adaptation is obtained by initializing the encoders using pretrained weights from the ImageNet dataset, followed by training the entire model on the Cityscapes dataset. In detail, we use the Adam optimizer with a constant learning rate $lr = 0.0001$ and train the entire network for 250 epochs. In our experiments, we compare the performance of our approach to directly training on the target dataset, which can be considered as a theoretical upper limit having full target knowledge. Due to the unbalanced class distribution of KITTI-360, we train a copy of the network in two steps, using the Adam optimizer with $lr = 0.0001$ on sequences 00-07. We train for 45 epochs while ignoring the most common classes *road*, *sidewalk*, *building*, and *vegetation*, followed by 55 epochs including all classes. Similarly, for SemKITTI-DVPS, we train another copy of the network on sequences 00-06, 09, and 10 for 30 epochs without the aforementioned classes plus *terrain* and *sky*, which is not included in the dataset, followed by 30 epochs including all classes. In Table I, we demonstrate that our implemented network is able to reproduce the performance of the baseline method GUDA [13].

C. Online Adaptation

In this section, we extensively evaluate our proposed CoDEPS with respect to both adapting to a new domain and retaining knowledge to mitigate forgetting. In detail, for all presented experiments, we freeze the shared encoder following the study by McCraith et al. [26]. Based on the ablation study in Sec. IV-D, we use a buffer size of 300. For RCS, we follow Hoyer et al. [15] and set $T = 0.01$. Updating the EMA model is done with $\alpha = 0.99$.

In Table II, we assess the performance of CoDEPS on all sequences of the KITTI-360 dataset and compare it with the baseline method "only source", which is also pretrained on Cityscapes but does not perform further adaptation to the target domain $\mathcal{T}$. This approach should be interpreted as a lower performance bound that must be improved. We demonstrate the key performance metrics of both protocols 1 and 2. As shown in Table II, CoDEPS achieves a performance boost across the board, as measured by the mIoU metric and all depth metrics of protocol 1. We attribute this improvement to the additional supervision signals incorporated into the segmentation head through our mixing strategy and the self-supervised reconstruction loss for depth adaptation. The improvement in semantic segmentation further enhances the panoptic segmentation metrics. With respect to protocol 2, CoDEPS reduces the depth errors on all sequences and improves the performance of semantic and panoptic segmentation on the vast majority of sequences. Note that on sequence 03 the panoptic metrics increase significantly despite the consistent mIoU, which we attribute to the more refined segmentation of objects due to our proposed cross-domain mixing strategy.

For the following experiments, we consider the case of using Cityscapes as the source domain and sequence 10 of KITTI-360 as the target domain. In Fig. 4, we illustrate

TABLE II
ADAPTATION PERFORMANCE

Method	Sequence	Protocol 1						Protocol 2					
		mIoU ↑	PQ ↑	SQ ↑	RQ ↑	RMSE ↓	Abs Rel ↓	mIoU ↑	PQ ↑	SQ ↑	RQ ↑	RMSE ↓	Abs Rel ↓
Only source	00	51.61	39.10	72.72	50.48	6.54	0.36	49.94	35.29	72.14	45.50	6.08	0.34
CoDEPS		**53.76**	**40.72**	**72.90**	**52.51**	**5.09**	**0.19**	**52.08**	**36.08**	**72.58**	**46.08**	**4.34**	**0.15**
Only source	02	45.97	31.83	67.62	41.08	6.26	0.35	46.55	30.13	65.03	39.30	6.06	0.36
CoDEPS		**46.62**	**32.11**	**67.74**	**41.62**	**4.31**	**0.16**	**47.48**	**30.33**	**65.35**	**39.46**	**3.76**	**0.13**
Only source	03	46.63	28.15	57.41	35.23	**8.20**	0.34	**52.10**	28.20	56.67	35.77	7.34	0.29
CoDEPS		**47.94**	**29.05**	**58.07**	**36.10**	8.26	**0.33**	52.00	**31.13**	**61.51**	**39.65**	**6.98**	**0.18**
Only source	04	45.02	29.34	65.48	38.15	6.70	0.37	45.53	30.13	**70.85**	38.89	6.61	0.38
CoDEPS		**45.40**	**29.78**	**65.89**	**38.84**	**5.00**	**0.19**	**45.68**	**30.63**	66.18	**39.89**	**4.33**	**0.17**
Only source	05	48.94	32.19	66.80	41.37	6.76	0.37	**44.52**	**27.34**	**60.72**	**35.58**	5.93	0.43
CoDEPS		**49.26**	**32.96**	**66.98**	**42.40**	**5.25**	**0.21**	43.79	26.48	60.33	34.88	**4.68**	**0.25**
Only source	06	46.03	29.88	66.58	38.42	6.09	0.39	46.28	31.79	70.47	41.40	6.12	0.37
CoDEPS		**46.53**	**30.45**	**66.66**	**39.20**	**4.97**	**0.22**	**47.27**	**31.99**	**70.74**	**41.71**	**4.23**	**0.18**
Only source	07	40.54	28.48	66.52	34.42	7.83	0.34	59.07	27.62	45.88	35.41	9.64	0.38
CoDEPS		**41.46**	**29.30**	**67.64**	**35.58**	**6.50**	**0.22**	**60.57**	**30.91**	**50.25**	**39.79**	**6.48**	**0.20**
Only source	09	50.59	37.26	74.06	47.38	6.03	0.36	50.78	36.57	72.22	46.75	5.60	0.35
CoDEPS		**52.29**	**38.02**	**74.88**	**48.21**	**4.74**	**0.19**	**51.53**	**37.56**	**72.87**	**47.99**	**4.56**	**0.16**
Only source	10	51.94	32.60	71.27	32.60	8.06	0.35	45.74	30.62	69.56	39.49	7.90	0.33
CoDEPS		**53.02**	**33.50**	**71.53**	**33.50**	**7.19**	**0.22**	**49.91**	**31.91**	**70.68**	**40.95**	**5.57**	**0.15**

Comparison between our CoDEPS and the performance of the same architecture without performing online continual learning on the respective sequence of the KITTI-360 dataset. Thus, "only source" refers to the model weights after pretraining on Cityscapes. The listed metrics are mean intersection over union (mIoU) for semantic segmentation; panoptic quality (PQ), segmentation quality (SQ), and recognition quality (RQ) for panoptic segmentation; root mean squared error (RMSE) and absolute relative error (Abs Rel) for monocular depth estimation. Bold values denote the best result on each sequence.

TABLE III
CONTINUAL LEARNING FOR MONOCULAR DEPTH ESTIMATION

Method	Batch	Protocol 1					Protocol 2					Protocol 3				
	current/target/source	RMSE	Abs Rel	δ_1	δ_2	δ_3	RMSE	Abs Rel	δ_1	δ_2	δ_3	RMSE	Abs Rel	δ_1	δ_2	δ_3
Only target	0 / 0 / 0	6.13	0.15	0.84	0.93	0.96	4.78	0.12	0.88	0.95	0.97	12.22	0.26	0.51	0.82	0.94
Only source	0 / 0 / 0	8.06	0.35	0.43	0.77	0.91	7.90	0.33	0.44	0.77	0.93	**10.16**	**0.19**	**0.78**	**0.93**	**0.97**
Online image	1 / 0 / 0	8.33	0.27	0.64	0.84	0.93	6.06	0.33	0.46	0.73	0.90	13.72	0.57	0.30	0.50	0.68
Target replay	1 / 2 / 0	**6.35**	**0.19**	**0.77**	**0.91**	**0.96**	**5.34**	**0.15**	**0.81**	**0.93**	**0.97**	12.48	0.44	0.34	0.68	0.88
CoDEPS	1 / 2 / 2	7.19	0.22	0.73	0.89	0.94	5.57	**0.15**	**0.81**	**0.93**	**0.97**	11.38	0.21	0.75	0.91	0.96

The root mean squared error (RMSE), absolute relative error (Abs Rel) as well as accuracies $\delta_1 = \delta < 1.25$, $\delta_2 = \delta < 1.25^2$, and $\delta_3 = \delta < 1.25^3$, obtained by adapting Cityscapes to sequence 10 of the KITTI-360 dataset. Best results without access to ground truth target data ("only target") in each category are in **bold**; second best are underlined.

the adaptation progress using unseen validation samples and compare the results to the ground truth. For depth, we visualize predictions generated by the network if it was only trained on $\mathcal{S}$ and $\mathcal{T}$, respectively. For panoptic segmentation, the progressive adaptation on the target domain is particularly visible on the *sidewalk* and *terrain* image regions, which CoDEPS learns to differentiate from the similarly looking classes *road* and *vegetation*. Furthermore, instances become more pronounced, e.g., the cyclist in the right sample. Despite the enhancements on the target domain, CoDEPS successfully maintains its performance on the source domain with only minimum decreases in depth estimation.

Depth Adaptation: We present the results for monocular depth estimation in Table III. The first row "only target" shows the theoretical performance on $\mathcal{T}$ (protocols 1 and 2) if the network would have been trained directly on this domain. Note that such a setup is infeasible in the real world when continuous operation must be guaranteed. The second row "only source" denotes the performance after pretraining on $\mathcal{S}$ without performing online continual learning. Comparing the absolute relative error as well as the accuracies δ_1, δ_2, and δ_3 between these rows reveals the domain gap. Note that the opposite gap can be observed when evaluating on $\mathcal{S}$ (protocol 3). While continual learning using the current online sample increases the accuracy of protocol 1, it also overfits to the current scene. That is, generalizability to the entire target domain is not achieved as shown by protocol 2. Introducing replay samples from the target buffer overcomes this issue and accounts for online samples of poor quality, improving protocols 1 and 2. However, both of the above result in catastrophic forgetting with respect to $\mathcal{S}$ (protocol 3). The final CoDEPS adds additional source replay yielding low errors and high accuracy by compromising on both $\mathcal{S}$ and $\mathcal{T}$.

Panoptic Adaptation: In Table IV, we also demonstrate the domain gap between $\mathcal{S}$ and $\mathcal{T}$ for semantic and panoptic segmentation. Similar to depth estimation, both "only target" and "only source" only perform well on their respective training domain without being able to generalize to the other. We further evaluate CoDEPS by comparing it with two competitive baselines that perform domain adaptation on segmentation

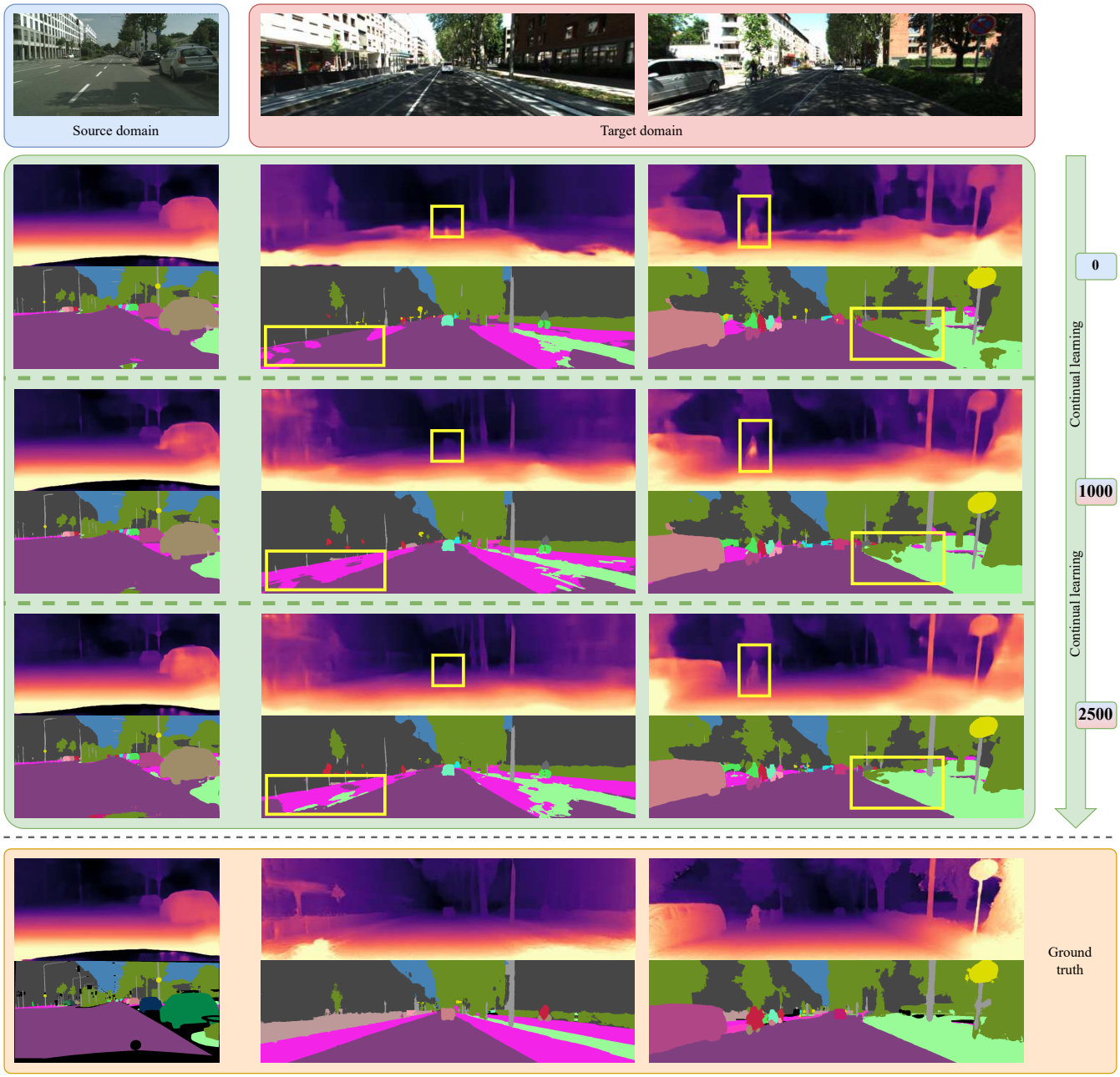


Fig. 4. Qualitative results for Cityscapes to KITTI-360 adaptation after pretraining on the source, i.e., 0 steps, and after having seen 1,000 and 2,500 frames. As shown in the left column, CoDEPS is able to avoid catastrophic forgetting on the source domain. The progressive adaptation on the target domain is particularly visible in the image areas highlighted by yellow boxes. "Stuff" classes of similar appearance like *sidewalk* vs. *road* (left image) and *terrain* vs. *vegetation* (right image) can be better distinguished by CoDEPS. Furthermore, instances become more pronounced as can be observed for the highlighted car (left image) and the cyclist (right image).

TABLE IV
CONTINUAL LEARNING FOR PANOPTIC SEGMENTATION

Method	Protocol 1				Protocol 2				Protocol 3			
	mIoU	PQ	SQ	RQ	mIoU	PQ	SQ	RQ	mIoU	PQ	SQ	RQ
Only target	64.65	41.91	76.68	51.48	55.12	36.58	67.41	46.15	46.33	28.24	70.03	36.92
Only source	51.94	32.60	71.27	42.44	45.74	30.62	69.56	39.49	72.87	49.19	77.45	60.40
GUDA [13]	45.56	29.70	70.67	39.05	47.62	31.03	64.00	40.49	66.57	44.39	75.95	55.32
DACS [31]	51.14	32.09	71.12	42.23	45.24	29.05	69.47	38.11	72.66	49.27	77.33	60.60
CoDEPS (online image)	**53.22**	33.46	**71.63**	43.46	49.51	31.49	64.17	40.71	72.81	**49.83**	77.25	**61.49**
CoDEPS (random sampling)	52.36	33.24	71.60	43.25	48.78	31.50	68.83	40.56	72.05	49.11	77.18	60.52
CoDEPS	53.02	**33.50**	71.53	**43.62**	**49.91**	**31.91**	**70.68**	**40.95**	**72.90**	49.76	**77.49**	61.22

The mean intersection over union (mIoU), panoptic quality (PQ), semantic quality (SQ), and recognition quality (RQ) are obtained by adapting Cityscapes to sequence 10 of the KITTI-360 dataset. Best results without access to ground truth target data ("only target") in each category are in **bold**; second best are underlined.

tasks: GUDA [13], which combines semantic segmentation and depth estimation, rendering their task comparable to ours, and DACS [31], which employs a class-mix strategy for offline domain adaptation of semantic segmentation. To ensure a fair comparison, both baselines are evaluated using the same settings as CoDEPS, including diversity sampling-based experience replay. The results in Table IV indicate that both approaches lead to a significant performance decrease across all three protocols. GUDA's reliance on self-supervised feature alignment using depth training is not effective in the continual learning setting, as shown in the results. DACS also suffers from a decline in performance, likely due to the strong intervention of its mixing strategy into the pretrained network, which can already produce reasonable predictions on the target domain without adaptation.

These results imply that traditional approaches from offline sim-to-real adaptation may not perform well in the online continual learning scenario. To further assess the impact of target replay and our diversity-based buffer sampling, we selectively deactivate both components. Applying the proposed cross-domain mixing strategy results in an improvement in protocol 1. However, similar to depth adaptation, the results are not fully generalizable to the entire target domain, e.g., SQ of protocol 2. Instead of diversity-based sampling, we use random sampling when both creating the source buffer and when updating the target buffer. Compared to CoDEPS, the performance heavily degrades demonstrating the efficacy of the sampling method. Finally, we present the classwise evaluation of the segmentation performance in Table V, which demonstrates improvements of CoDEPS in the IoU metrics for most classes. In particular, we observe significant enhancements of the *two-wheeler* and *terrain* classes. The latter can also be observed in Fig. 4. In fact, CoDEPS outperforms even the model trained directly on the target domain using ground truth supervision for the latter class.

TABLE V
CLASSWISE EVALUATION

	Class	Only target	Only source	CoDEPS
Stuff	Road	93	89	91
	Sidewalk	40	32	37
	Building	88	85	85
	Fence	43	14	22
	Pole	35	29	32
	Traffic sign	40	35	38
	Vegetation	78	73	75
	Terrain	54	21	39
	Sky	82	79	81
Thing	Person	47	38	38
	Rider	47	29	36
	Car	91	83	84
	Truck	1	4	2
	Two-wheeler	33	27	38
	Mean	55.1	45.7	49.9

The classwise mIoU is based on protocol 2 in Table IV. We compare CoDEPS against two baselines that were trained using source ("only source") or target data ("only target"), respectively. CoDEPS provides a significant performance boost of 4.2% in terms of the mIoU metric.

D. Ablation Study of the Replay Buffer

We extensively study different sizes of the replay buffer and the effect of diversity sampling as explained in Sec. III-B. We list our results in Table VI. Note that an infinite replay buffer contains 2,975 source and a maximum of 2,683 target samples in the employed setting, i.e., adapting from Cityscapes *train* to KITTI-360 using sequence 10 *adapt*. Generally, a larger replay buffer yields higher performance with respect to both adaptation capability and avoiding catastrophic forgetting. Additionally, the proposed diversity sampling using semantic classes for the source and image features for the target samples increases the performance throughout the experiments. However, a greater buffer size increases the required storage posing a challenge for real-world deployment. Based on the presented results, we select a buffer size of 300 with active diversity sampling as for smaller sizes the performance of semantic segmentation on the target domain degrades.

E. Continual Adaptation

Finally, we evaluate the performance of CoDEPS in the context of multi-domain adaptation, i.e., $\mathcal{S} \rightarrow \mathcal{T}_1 \rightarrow \mathcal{T}_2$. In particular, we first adapt to sequence 10 of KITTI-360 followed by sequence 08 of SemKITTI-DVPS, then we invert the adaptation order. To analyze forward and backward transfer as defined for continual learning [23], we compute the metrics on the *val* split of the source and the *adapt* parts of the respective target domains. We report the results in Table VII. Note that we use $\alpha_{\mathcal{S}\rightarrow\mathcal{T}_1} = 0.9$ and $\alpha_{\mathcal{T}_1\rightarrow\mathcal{T}_2} = 0.7$ for updating the EMA model according to Eq. 10 since the network should adapt more strongly when deployed to $\mathcal{T}_2$ due

TABLE VI
ABLATION STUDY ON THE REPLAY BUFFER

Size	Div.	Protocol 2						Protocol 3					
		mIoU ↑	PQ ↑	SQ ↑	RQ ↑	RMSE ↓	Abs Rel ↓	mIoU ↑	PQ ↑	SQ ↑	RQ ↑	RMSE ↓	Abs Rel ↓
∞		49.15	31.95	69.08	40.96	4.94	0.15	73.25	50.37	77.77	61.87	10.76	0.21
1000		49.11±0.69	31.85±0.25	66.82±3.06	40.93±0.06	**5.04**±0.01	**0.14**±0.00	72.84±0.33	49.93±0.20	77.51±0.05	61.39±0.28	11.35±0.39	0.22±0.01
1000	✓	49.36	31.83	68.89	**41.01**	5.30	0.15	**73.50**	**50.05**	**77.67**	**61.48**	12.06	0.23
500		48.77±0.39	31.54±0.39	67.39±2.16	40.66±0.54	5.20±0.20	0.15±0.00	72.38±0.26	49.48±0.14	77.45±0.16	60.90±0.23	**11.14**±0.54	0.22±0.01
500	✓	49.56	31.83	70.11	40.96	5.55	0.16	72.78	49.68	77.39	61.10	11.30	0.22
300		48.78±0.05	31.50±0.19	68.83±2.31	40.56±0.15	5.27±0.16	0.15±0.00	72.05±0.25	49.11±0.30	77.18±0.07	60.52±0.35	**11.14**±0.22	0.22±0.01
300	✓	**49.91**	**31.91**	**70.68**	40.95	5.57	0.15	72.90	49.76	77.49	61.22	11.38	**0.21**
100		48.27±0.84	30.71±0.41	63.95±0.41	39.79±0.38	5.83±0.15	0.16±0.00	69.75±1.77	47.94±0.95	76.66±0.25	59.39±1.16	10.86±0.56	0.22±0.02
100	✓	48.40	30.85	64.07	39.95	5.31	0.16	72.35	48.81	77.16	60.25	11.71	0.22
25		46.03±1.03	29.62±0.37	66.10±2.26	38.48±0.45	5.25±0.26	**0.14**±0.01	67.23±0.85	45.90±0.66	75.69±0.38	57.21±0.76	11.81±0.22	0.22±0.01
25	✓	46.35	29.73	63.35	38.58	5.62	0.17	68.84	46.34	76.06	57.78	12.51	0.24

The numbers above are obtained by adapting Cityscapes to sequence 10 of the KITTI-360 dataset. Here, an infinite buffer size equals 2,975 source samples and a maximum of 2,683 target samples. Note that the effective size is two times the shown value as it refers to both source and target replay. The term "Div." refers to diversity sampling. Where diversity sampling is not used, the same experiment is repeated three times with different random seeds to ensure a statistically reliable measure of performance. The results of these experiments are presented as the mean and standard deviation. Best results in each category are in **bold**; second best are underlined.

TABLE VII
CONTINUAL LEARNING ON MULTIPLE DOMAINS

Domain	mIoU	PQ	SQ	RQ	RMSE	Abs Rel	mIoU	PQ	SQ	RQ	RMSE	Abs Rel	mIoU	PQ	SQ	RQ	RMSE	Abs Rel
	→ Pretraining on Cityscapes						→ Adaptation on KITTI-360						→ Adaptation on SemKITTI-DVPS					
Cityscapes	72.87	49.19	77.45	60.40	10.16	0.19	72.90	49.76	77.49	61.22	11.38	0.21	72.42	48.74	77.08	60.20	10.65	0.21
KITTI-360 seq. 10	45.74	30.62	69.56	39.49	7.90	0.33	49.91	31.91	70.68	40.95	5.57	0.15	49.26	32.32	64.08	40.95	5.23	0.15
SemKITTI-DVPS seq. 08	51.95	45.24	76.07	57.20	6.17	0.34	49.48	43.26	74.24	57.26	5.60	0.21	53.70	46.50	76.53	59.43	4.32	0.16
	→ Pretraining on Cityscapes						→ Adaptation on SemKITTI-DVPS						→ Adaptation on KITTI-360					
Cityscapes	72.87	49.19	77.45	60.40	10.16	0.19	72.75	49.01	77.36	60.35	10.82	0.22	72.51	48.87	76.98	60.28	11.41	0.21
KITTI-360 seq. 10	45.74	30.62	69.56	39.49	7.90	0.33	49.26	31.66	70.26	41.40	6.30	0.17	50.05	31.92	70.50	41.48	5.47	0.16
SemKITTI-DVPS seq. 08	51.95	45.24	76.07	57.20	6.17	0.34	52.31	44.29	75.58	56.87	4.56	0.16	53.83	47.29	76.55	60.01	4.25	0.16

CoDEPS is continually applied to three domains using Cityscapes as the initial source domain and then adapting to KITTI-360 and SemKITTI-DVPS. The listed numbers on the target domains are based on protocol 2.

to the larger amount of previously seen data. As shown in the first row of both adaptation orders, CoDEPS is able to mitigate catastrophic forgetting with respect to $\mathcal{S}$ maintaining its performance. We make a similar observation when re-evaluating $\mathcal{T}_1$ after the second adaptation step to $\mathcal{T}_2$. In particular, CoDEPS achieves positive backward transfer on SemKITTI-DVPS when adapting to KITTI-360. On the same adaptation order, we observe positive forward transfer for KITTI-360, i.e., the performance increases although CoDEPS was only adapted to SemKITTI-DVPS.

In Fig. 5, we illustrate the evolution of the performance metrics on SemKITTI-DVPS sequence 08 during adaptation (protocol 1). We compare the error without adaptation to directly adapting to SemKITTI-DVPS versus first adapting to KITTI-360. For both semantic segmentation and depth estimation, it can be clearly observed that the performance improves if more images have been seen. Additionally, adapting first to KITTI-360 results in a large performance increase for both semantic and panoptic segmentation. We account this to the fact that KITTI-360 sequence 10 leads to strongly improved performance, shown in Table VII, that can be transferred to the SemKITTI-DVPS domain.

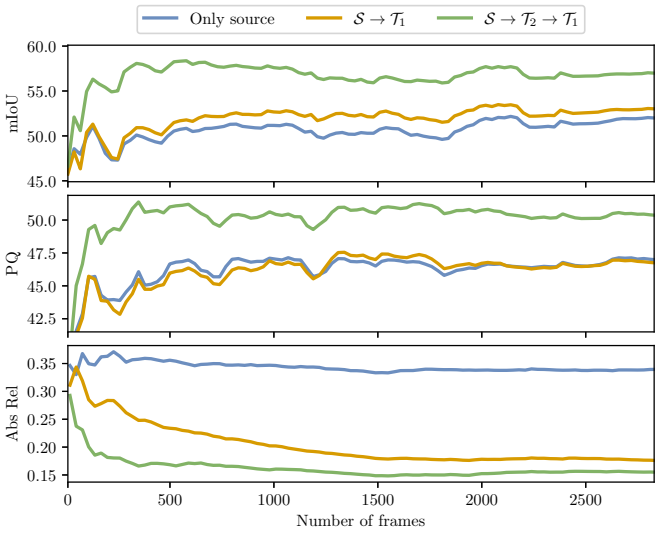


Fig. 5. Evolution of performance metrics on SemKITTI-DVPS sequence 08 during adaptation (protocol 1). The metrics are averaged until the given frame number. The target domains $\mathcal{T}_1$ and $\mathcal{T}_2$ refer to SemKITTI-DVPS and KITTI-360, respectively. It can be seen that there is positive forward transfer when first adapting on $\mathcal{T}_2$.

V. CONCLUSION

In this paper, we present CoDEPS as the first approach for online continual learning for joint monocular depth estimation and panoptic segmentation. CoDEPS enables the vision system of a robotic platform to continually enhance its performance in an online fashion. In particular, we propose a new cross-domain mixing strategy to adapt panoptic segmentation combining annotated source data with unlabeled images from a target domain. To mitigate catastrophic forgetting, CoDEPS leverages experience replay using a buffer composed of source and target samples. We explicitly address the limited storage

capacity of robotic platforms by setting a fixed size for the replay buffer. To ensure distinct replay samples, we use rare class sampling on the source set and employ image-based diversity sampling when updating the target buffer. Using extensive evaluations, we demonstrate that CoDEPS outperforms competitive baselines while avoiding catastrophic forgetting in the online continual learning setting. Future work will explore cross-task synergies and the use of pretext tasks for domain adaptation.

Acknowledgment

This work was partly funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR and the Bundesministerium für Bildung und Forschung (BMBF) under grant agreement No FKZ 16ME0027.

References

[1] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Conference on Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *International Conference on Computer Vision*, 2019.

[3] Borna Bešić and Abhinav Valada. Dynamic object removal and spatio-temporal RGB-D inpainting via geometry-aware adversarial learning. *IEEE Transactions on Intelligent Vehicles*, 7(2):170–185, 2022.

[4] Borna Bešić, Nikhil Gosala, Daniele Cattaneo, and Abhinav Valada. Unsupervised domain adaptation for LiDAR panoptic segmentation. *IEEE Robotics and Automation Letters*, 7(2):3404–3411, 2022.

[5] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Unsupervised monocular depth and ego-motion learning with structure and semantics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[6] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 12472–12482, 2020.

[7] Gong Cheng and James H. Elder. VCSeg: Virtual camera adaptation for road segmentation. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1969–1978, 2022.

[8] 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 *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[10] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[11] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. In *International Conference on Computer Vision*, pages 3827–3837, 2019.

[12] Nikhil Gosala and Abhinav Valada. Bird's-eye-view panoptic segmentation using monocular frontal view images. *IEEE Robotics and Automation Letters*, 7(2):1968–1975, 2022.

[13] Vitor Guizilini, Jie Li, Rareș Ambruș, and Adrien Gaidon. Geometric unsupervised domain adaptation for semantic segmentation. In *International Conference on Computer Vision*, pages 8537–8547, 2021.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[15] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. DAFormer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 9924–9935, 2022.

[16] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. Cross-view regularization for domain adaptive panoptic segmentation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 10133–10144, 2021.

[17] Marvin Klingner, Mouadh Ayache, and Tim Fingscheidt. Continual batchnorm adaptation (CBNA) for semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20899–20911, 2022.

[18] Yevhen Kuznietsov, Marc Proesmans, and Luc Van Gool. CoMoDA: Continuous monocular depth adaptation using past experiences. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2907–2917, 2021.

[19] Yevhen Kuznietsov, Marc Proesmans, and Luc Van Gool. Towards unsupervised online domain adaptation for semantic segmentation. In *European Conference on Computer Vision*, pages 261–271, 2022.

[20] Hanhan Li, Ariel Gordon, Hang Zhao, Vincent Casser, and Anelia Angelova. Unsupervised monocular depth learning in dynamic scenes. In *Conference on Robot Learning*, pages 1908–1917. PMLR, 2021.

[21] Shunkai Li, Xin Wang, Yingdian Cao, Fei Xue, Zike Yan, and Hongbin Zha. Self-supervised deep visual odometry with online adaptation. In *IEEE/CVF Conference Com-*

puter Vision and Pattern Recognition, pages 6339–6348, 2020.

[22] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022.

[23] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Conference on Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[24] Adrian Lopez-Rodriguez and Krystian Mikolajczyk. DESC: Domain adaptation for depth estimation via semantic consistency. *International Journal of Computer Vision*, 131(3):752–771, Mar 2023.

[25] Giulio Mattolin, Luca Zanella, Elisa Ricci, and Yiming Wang. ConfMix: Unsupervised domain adaptation for object detection via confidence-based mixing. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 423–433, January 2023.

[26] Robert McCraith, Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Monocular depth estimation with self-supervised instance adaptation. *arXiv preprint arXiv:2004.05821*, 2020.

[27] Rohit Mohan and Abhinav Valada. Amodal panoptic segmentation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 20991–21000, 2022.

[28] Rohit Mohan and Abhinav Valada. Perceiving the invisible: Proposal-free amodal panoptic segmentation. *IEEE Robotics and Automation Letters*, 7(4):9302–9309, 2022.

[29] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. ClassMix: Segmentation-based data augmentation for semi-supervised learning. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1368–1377, 2021.

[30] Siyuan Qiao, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. ViP-DeepLab: Learning visual perception with depth-aware video panoptic segmentation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 3996–4007, 2021.

[31] Wilhelm Tranheden, Viktor Olsson, Juliano Pinto, and Lennart Svensson. DACS: Domain adaptation via cross-domain mixed sampling. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1378–1388, 2021.

[32] Abhinav Valada, Gabriel Oliveira, Thomas Brox, and Wolfram Burgard. Towards robust semantic segmentation using deep fusion. In *Robotics: Science and Systems Workshop, Are the Sceptics Right*, 2016.

[33] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Continual SLAM: Beyond lifelong simultaneous localization and mapping through continual learning. In Aude Billard, Tamim Asfour, and Oussama Khatib, editors, *Robotics Research*, pages 19–35, Cham, 2023. Springer Nature Switzerland.

[34] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. CoVIO: Online continual learning for visual-inertial odometry. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2023.

[35] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

[36] Nan Yang, Lukas von Stumberg, Rui Wang, and Daniel Cremers. D3VO: Deep depth, deep pose and deep uncertainty for monocular visual odometry. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 1281–1292, 2020.

[37] Zhenyu Zhang, Stéphane Lathuilière, Elisa Ricci, Nicu Sebe, Yan Yan, and Jian Yang. Online depth learning against forgetting in monocular videos. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 4493–4502, 2020.

[38] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

7.2 Deep Label Embedding Learning for Classification

The appended paper (under review) follows.

Deep Label Embedding Learning for Classification

Paraskevi Nousi[a], Anastasios Tefas[a]

[a]*Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece*

Abstract

The one-hot 0/1 encoding method is the most popularized encoding method of class labels for classification tasks. Despite its simplicity and popularity, it comes with limitations and weaknesses, like failing to capture the inherent uncertainty in data labels, and making classifiers more prone to overfitting. In this paper, we tackle these shortcomings with a framework for learning soft label embeddings. Two variants are proposed: first, a learnable general-class embedding which aims to capture information regarding inter-class similarities, and second, a neural architecture which can be added to any neural classifier and aims to learn inter-instance similarities. The inherent uncertainty in data labels is thus somewhat alleviated, allowing the network to focus on incorrectly classified samples, instead of difficult but correctly classified ones. Our experimental study on multiple classification benchmarks of increasing difficulty, using neural networks of varying depth and width, show that the proposed method leads to better classification accuracy, highlighting its ability to generalize to unseen samples.

Keywords: label embedding, soft labels, class similarities, instance similarities

1. Introduction

The one-hot 0/1 encoding method is the most popularized encoding method of class labels for classification tasks. Most existing neural network based classification methods use this encoding along with some variant of the cross entropy loss function [1]. Despite its simplicity and popularity, it comes with limitations and weaknesses which we investigate in this work.

Email addresses: paranous@csd.auth.gr (Paraskevi Nousi), tefas@csd.auth.gr (Anastasios Tefas)

One-hot encoding fails to capture the *inherent uncertainty* in data labeling processes. There are multiple aspects to this issue. First, a dataset typically consists of data samples belonging to classes of the same category, e.g., MNIST depicts handwritten digits. In general, a class of digits may resemble another class more or less than the remaining classes. For example, *fours* tend to resemble - and be confused as - *nines* more often than any other digit. Class similarities like these are present in most classification datasets, even more generic ones. Second, some instances of a class may heavily resemble instances of other classes, much like a small dog may resemble a cat or a fox, or a flying bird may resemble a plane. These similarities are specific to each instance of the dataset and may be considered as outliers. Finally, there is always the potential of human error, either related to the aforementioned uncertainties or simply by mistake.

Figure 1 illustrates this point using the CIFAR10 dataset [2]. Two dimensional representations are obtained using t-SNE [3] and a DenseNet [4] classifier, using the representations of the penultimate layer. The general class similarities, like those between the *dog* and *cat* classes, or the *automobile* and *truck* classes, reflect the actual similarity between these real categories. Instance specific similarities exist as well, as shown in the marked misclassified samples: (a) a dog misclassified as a cat, (b) a truck misclassified as a car, (c) a deer misclassified as a horse, (d) a bird misclassified as a ship, (e) a bird misclassified as a ship, (f) a deer misclassified as a horse, and (g) a dog misclassified as a bird.

Furthermore, one-hot encoding can lead to overfitting more easily, as neural networks have the capacity to bend their representation space in ways such that they perfectly capture the training samples, despite their outlying status. As an example, consider a binary classification example, where a difficult sample may be correctly classified with a predicted probability of 0.8. Despite the prediction being correct, most recent approaches to this problem would further modify the network weights such that the predicted probability lies closer to 1. In order to do so, the separating hyperplane may bend in a way that excludes unseen samples even if they lie close to the groups of training samples.

In this paper, we tackle these shortcomings with a framework for learning soft label embeddings. Two variants are proposed: first, a learnable general-class embedding which aims to capture information regarding inter-class similarities, and second, a neural architecture which can be added to any neural classifier and aims to learn inter-instance similarities. The inherent

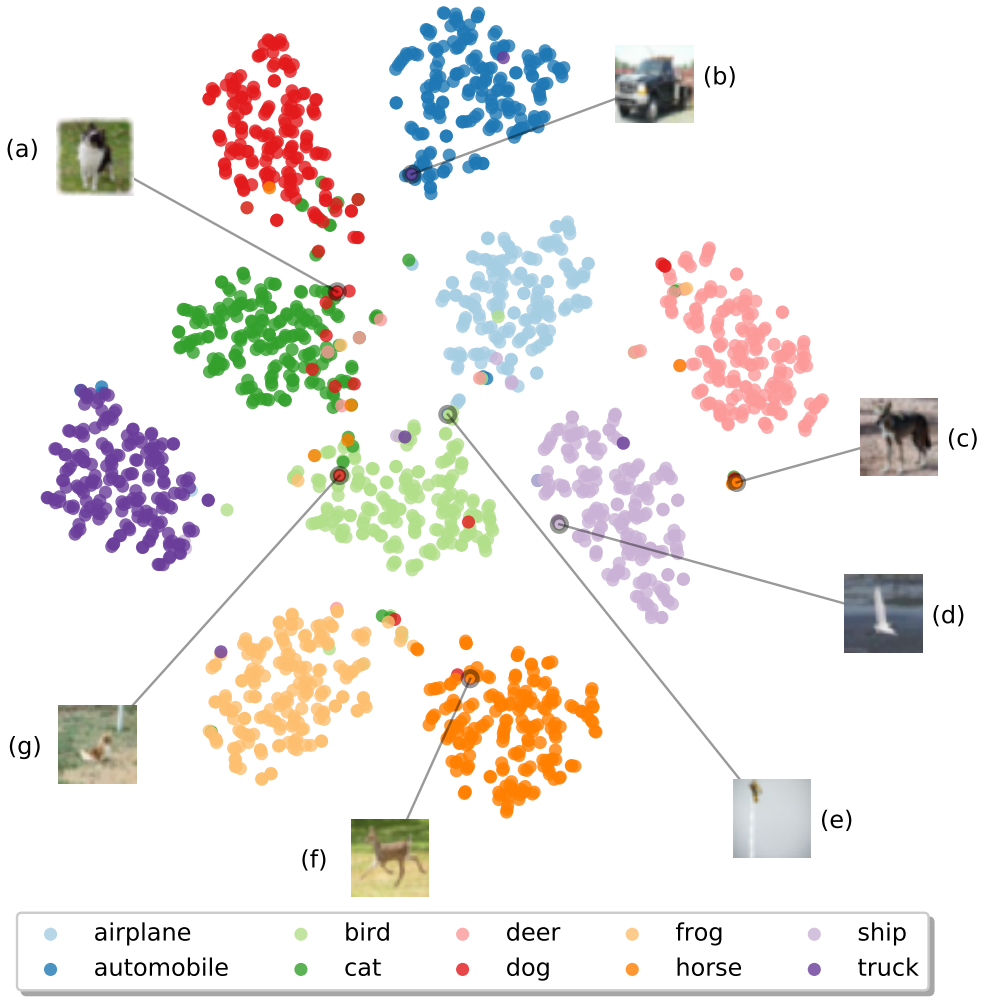


Figure 1: Visualization of feature vectors from the CIFAR10 dataset using t-SNE.

uncertainty in data labels is thus somewhat alleviated, allowing the network to focus on incorrectly classified samples, instead of difficult but correctly classified ones. The concept of this framework is similar to real-life learning procedures, where uncertainty rules over many subjects. Students are encouraged to eventually become teachers themselves, in the broader sense of these words, and to achieve that they must be allowed some liberties when learning a new subject which presents uncertainties. During the learning process, students may draw from their own experiences and go on to even enhance the subject with their understanding of it. To the best of our knowledge, our work is the first label embedding framework to consider similarities the class level as well as at the instance level as well as a combination of the two.

The rest of this paper is structured as follows. Section 3 introduces the notation used in the proposed method as well as the general concept. Sections 3.1 and 3.2 describe the two variants of the proposed method, aiming to capture general-class and instance-specific similarities in the soft embedding. The method for combining of the two embedding variants is presented in Section 3.3. Finally, Section 4 summarizes our experimental results and our conclusions are drawn in Section 5.

2. Related Work

Soft labels, label embedding and label smoothing are all problems similar to the one tackled in this work, and they have been studied to some extent in recent literature.

Image annotation is a tedious and uninspiring task, prone to errors, in both objective and subjective criteria. Subjective errors in particular are almost impossible to overcome, forcing neural networks to either learn these mistakes or to find ways to overcome these *noisy* labels. In fact, labelling errors are evident in the large number of works revolving around learning from noisy labels. In [5], two types of label noise are considered: label flips, where images have been assigned a wrong class, and outliers, where the image does not depict any of the classes in the dataset but has been erroneously assigned to one. To mitigate these errors, a linear layer was added at the final softmax layer, to learn the noise distribution without supervision, i.e., prior modeling of this distribution. More recently, in [6], a single layer network was used to learn to generate soft labels from noisy labels.

Soft labels have been studied in the past, in the context of simpler classifiers like the k-Nearest Neighbor one [7], and recently scientific research in this task has resurfaced. In [8], a method for modeling subjectivity in emotion recognition was proposed. An ensemble method, where different networks are trained with different annotators, was compared against a single network trained on soft labels, generated by averaging the labels from different annotators. In [9], a soft label method was introduced for ordinal regression, i.e., classification problems were classes are not independent, but follow some sort of order. Maintaining inter-class relationships is crucial in such tasks, and we further argue that a natural order between classes can be found in most classification datasets, especially as the number of classes increases. In [10], soft labels were learned in a meta learning fashion, by treating them as learnable parameters, modeling both class-level and instance-level similarities.

Soft labels have also recently been linked to knowledge distillation methods [11, 12]. In [13], soft labels were explored in the context of relation extraction with neural networks. A teacher network was used to learn well-informed soft labels and its knowledge was distilled and transferred to a student network. In [11], posterior class probabilities were estimated using the data samples, which were used as soft labels, encoding information about the similarities between the data samples. Further, in [12], a distillation method aiming to reveal subclass similarities was introduced. In [14], a teacher-free knowledge distillation method was proposed, using label smoothing regularization. It was also proven that knowledge distillation is a type of learned label smoothing.

The significance of label smoothing has been gaining interest in recent years. In [15], it was shown that label smoothing does not hurt the generalization of a model's predictions, but using a simple label smoothing method on a network makes it a less effective teacher in a knowledge-distillation setting. In [16], it was shown that label smoothing can indeed be useful to teacher networks in the presence of noisy labels. In [17], an investigation was conducted into the effect of label smoothing regularization on the convergence of stochastic gradient descent methods. It was shown that label smoothing can help speed up the convergence. In [18], a theoretical framework was introduced, to explain how label smoothing controls the generalization loss. In [19], an online label smoothing strategy was proposed, to generate soft labels based on the statistics of the model's predictions.

Label embedding is closely related to label smoothing. In [20], a label embedding method was proposed for text classification, in a multi-task learning

context. Also in text classification, in [21], a model-based label embedding method was coupled with a self-interaction attention mechanism. In [22], a label embedding network was introduced for soft training of deep neural networks. The embedding network learns from the predictions of the classifier and the two networks are trained in unison. Label embedding is also useful in multi-label classification tasks, as recently showcased in [23].

We are interested in modelling relationships both at a class-level and an instance-level into the proposed label embedding methods. Inter-class similarities were also studied in [24], using a label smoothing technique to instill general class similarity information into the learning task. We achieve the same goal using a simple linear network, which learns to map the one-hot encodings into soft versions of themselves, using the network's predictions. Instance-specific labels were recently studied in [25], in a self-distillation mechanism. Instead of self-learning the instance-specific labels, we use an external observer, in the form of an Autoencoder (AE) [26], to uncover the instance level similarities in its learned latent space.

3. Proposed Methodology

Let $\mathbf{x}_i \in \mathbb{R}^D$ denote an input sample and $\mathbf{y}_i \in \{0,1\}^K$ such that $\sum_{k=1}^{K} y_{i,k} = 1$ be its one-hot encoding, in a set of $i = 1, \ldots, N$ training samples spanning over K distinct classes. A standard classifier $f(\mathbf{x})$ is trained to map the input samples $\mathbf{x}_i$ to their corresponding one-hot labels and to make predictions on unseen samples. Typically, the Cross Entropy loss function is used for this purpose:

$$CE(\mathbf{y}, \mathbf{p}) = -\sum_{k=1}^{K} y_k \cdot \log(p_k) \tag{1}$$

where $\mathbf{p}$ are the probabilities predicted by $f(\mathbf{x})$, obtained as the output of a softmax function on the output, or logits, of the classifier. We propose the use of soft labels $\hat{\mathbf{y}}_i \in \mathbb{R}^K$ such that $\sum_{k=1}^{K} \hat{y}_{i,k} = 1$ for all samples in the dataset, such that they capture not only the groundtruth label of each sample but also inter-class similarities as well as similarities between each sample and other samples present in the dataset. Our hypothesis is that by using soft labels, a neural classifier can learn to generalize better, as the sum of errors from correctly-classified but difficult samples will decrease. In terms

of the representation learned, less effort will be consumed towards separating such samples from their similar neighbors.

The new learning task is formulated as:

$$CE(\hat{\mathbf{y}}, \mathbf{p}) = -\sum_{k=1}^{K} \hat{y}_k \cdot \log(p_k) \tag{2}$$

where $\hat{\mathbf{y}}$, i.e., the soft labels, are given by a differentiable function $g(\cdot)$, the parameters of which can be learned in conjunction with the parameters of the classifier during training.

In the following Sections we define two methods to generate such soft labels. In both cases, the label embedding learned stems from the representations learned by the classifier itself, hence the title of self-learned embedding. The first method aims to capture resemblances between the classes present in the dataset and is described in Section 3.1. The second method aims to capture resemblances between each sample and any and all instances in the dataset, regardless of their corresponding classes. Section 3.2 describes this method in detail, while Section 3.3 presents our proposed method of combining the aforementioned methods in a single architecture.

3.1. General Class Soft-Label Embedding

We formulate the function $g(\cdot)$ for the case of general class similarities as a simple embedding of the form $\hat{\mathbf{y}} = g(\mathbf{y}) = \mathbf{W} \cdot \mathbf{y}$ where $\mathbf{W} \in \mathbb{R}^{K \times K}$ is a learnable weights matrix. This can be viewed as a neural network with a single linear hidden layer. Despite its simplicity, this embedding can model extreme cases where:

$$\mathbf{W} = \mathbf{I}_K \tag{3}$$

corresponding to the 0/1 one-hot encoding, or:

$$\mathbf{W} = \frac{1}{K}\mathbf{J}_K \tag{4}$$

where the notation $\mathbf{J}_K$ denotes an all-ones matrix of size $K \times K$, corresponding to the case where all class probabilities are equal.

We are, however, interested in the more generic case where each class label is given as a linear combination of all classes, including itself which should intuitively hold a larger weight:

$$\hat{\mathbf{Y}} = g(\mathbf{Y}) = \mathbf{W} \cdot \mathbf{Y} = \mathbf{W}. \tag{5}$$

where $\hat{\mathbf{Y}} \in \mathbb{R}^{K \times K}$ is the soft label matrix, i.e., each row $\hat{\mathbf{y}}_k$ corresponds to the soft labels of the k-th class, and $\mathbf{Y} = \mathbf{I}_K$ holds the one-hot encodings of all classes. That is, the new label vector for the k-th class is given by:

$$\hat{\mathbf{y}}_k = g(\mathbf{y}_k) = \mathbf{W} \cdot \mathbf{y}_k = [W_{k1}, W_{k2}, \ldots, W_{kk}, \ldots, W_{kK}] \tag{6}$$

i.e., practically the k-th row of the weight matrix $\mathbf{W}$, as $\mathbf{y}_k$ is the one-hot encoding of the k-th class. As mentioned, the weights matrix should optimally conform to:

$$W_{kk} \geq \sum_{l \neq k}^{K} W_{kl}, \tag{7}$$

so as to retain the groundtruth information. A softmax function is applied on $\hat{\mathbf{y}}_k$ to ensure that $\sum_{l=1}^{K} \hat{y}_{kl} = 1$.

Figure 2 illustrates the use of the proposed general-class network alongside a generic neural classifier. The two networks are trained in parallel using a single learning objective, optimizing the cross entropy given by Eq. (2) for all input samples. If not for the constraint imposed by Eq. (7), it is evident that this architecture is in direct danger of collapsing to the case presented in Eq. (4), where all class labels are equal and both networks learn random weights. There are multiple straightforward ways to avoid this scenario. One is to add a regularization term to the loss to directly enforce Eq. (7). However, this entails the threat of collapsing to the case Eq. (3), i.e., the labels remain binary. Instead, we enforce this constraint by first initializing the weights matrix using an identity matrix. Some noise is added to these weights to avoid harsh 0/1 numbers. Furthermore, a different learning rate is used for the label embedding network which forces the network weights to update slowly in comparison to the weights of the classification network, for which a larger learning rate is used. Another way to circumvent this issue is to set the new classification targets to be a weighted combination of the one-hot encoding and their linear combinations given by $g(\mathbf{y})$, i.e.:

$$\hat{\mathbf{y}}_k = \alpha \cdot g(\mathbf{y}_k) + (1 - \alpha) \cdot \mathbf{y}_k \tag{8}$$

where $\alpha \in (0, 1)$ controls the softness of the labels. This approach is used in this work, for its simplicity.

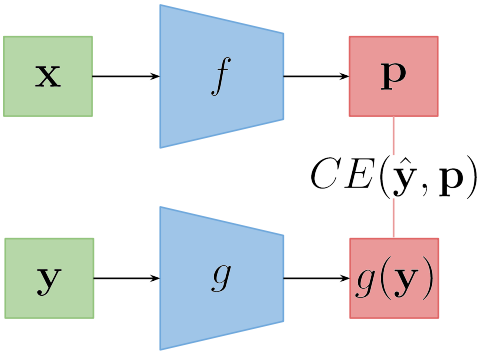


Figure 2: Schematic representation of the learning procedure of a classification network attached with the proposed general-class label embedding network.

3.2. Instance Specific Soft-Label Embedding

In the instance-specific case, the soft labels for each instance $\mathbf{x}_i$ are a function of the instance itself, that is:

$$\hat{\mathbf{y}}_i = h(\mathbf{x}_i). \tag{9}$$

The criterion from Eq. (2) still applies in this case and the function $h(\cdot)$ can take the form of a network which takes $\mathbf{x}_i$ as input and outputs $\hat{\mathbf{y}}_i$.

An *external* and *objective* representation of $\mathbf{x}_i$ can be extracted using an Autencoder, where the intermediate representation $\mathbf{z}_i$ is set to be K-dimensional. An AE can be formally defined by its two parts, the encoder and decoder networks, as a composite function:

$$\hat{\mathbf{x}}_i = h_{dec}(h(\mathbf{x_i})), \tag{10}$$

where $h(\cdot)$, $h_{dec}(\cdot)$ are the encoding and decoding functions respectively, and $\hat{\mathbf{x}}_i \in \mathbb{R}^D$ is the network's output, which is trained to approximate the input. The intermediate representation $\mathbf{z}_i = h(\mathbf{x}_i)$ is given by the encoder. To ensure a proper mapping between the AEs latent dimensions and the one-hot encoding, a cross entropy criterion is added to the objective. In this setting, in an extreme case of overfitting, the intermediate representation can take the form of a one-hot encoding at the cost of large reconstruction errors. Furthermore, it is also possible that the AE objective will collapse to mapping every sample to the same representation, which is another tedious solution of the problem. Both of these scenarios are mitigated by the addition

of the reconstruction loss of the AE, in terms of MSE between its input and predicted output $\tilde{\mathbf{x}}$:

$$MSE(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2. \tag{11}$$

The final learning objective is a weighted combination of the reconstruction, intermediate representation to labels association and final classification loss functions:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} CE(\mathbf{y}_i, \sigma(\mathbf{z}_i)) + CE(\hat{\mathbf{y}}_i, \mathbf{p}_i) + \lambda \cdot MSE(\mathbf{x}_i, \tilde{\mathbf{x}}_i) \tag{12}$$

where λ is a constant introduced to weigh the classification and reconstruction losses. Figure 3 summarizes the proposed architecture for this case. The input is fed into both the classifier and an AE. The intermediate representation of the AE is trained to match the groundtruth labels using a softmax function $\sigma(\cdot)$, and the cross entropy criterion. The AE is also trained using the MSE between the input and its prediction. Finally, the classifier's predictions are matched to the soft targets, which are given by the intermediate representation. An equation similar to Eq. 8 is used to generate the soft targets, so as to maintain the groundtruth information:

$$\hat{\mathbf{y}}_k = \beta \cdot \sigma(\mathbf{z}_k/T) + (1 - \beta) \cdot \mathbf{y}_k \tag{13}$$

where T denotes the softmax temperature.

3.3. Combined General-Class and Instance-Specific Label Embedding

The two methods described in the previous Sections can easily be combined into a single architecture, at the cost of increased training time. The combination is straightforward and a graphical description is given by Figure 4. The main difference to the instance-specific case is the addition of the general-class label embedding network $g(\mathbf{y})$, and the learning objective is modified accordingly:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} CE(\mathbf{y}_i, \sigma(\mathbf{z}_i)) + \beta \cdot CE(\hat{\mathbf{y}}_i, \mathbf{p}_i) + \gamma \cdot MSE(\mathbf{x}_i, \tilde{\mathbf{x}}_i) \tag{14}$$

where β and γ weigh the classification and reconstruction losses. In this case, it is crucial to properly initialize the general-class embedding network so as

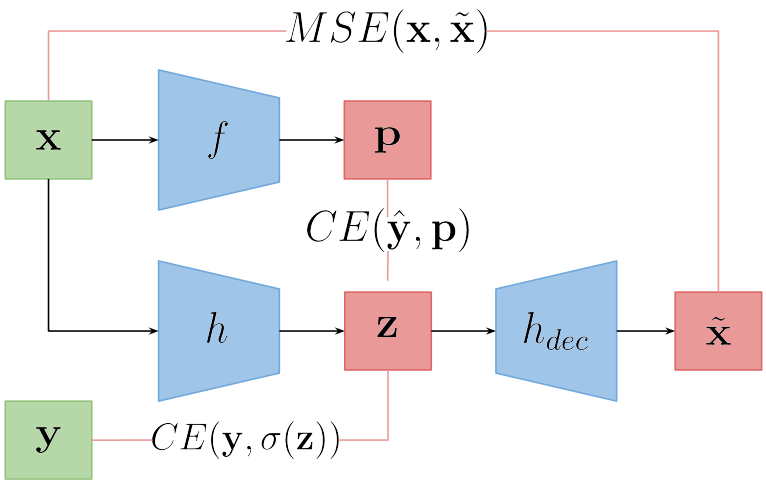


Figure 3: Schematic representation of the learning procedure of a classification network attached with the proposed instance-specific label embedding network.

to output labels as close as possible to the groundtruth one-hot encoding. In practice, the soft labels $\hat{\mathbf{y}}$ which are used to train the classifier, are a linear combination of the one-hot groundtruth labels, the general-class embeddings and the instance-specific embeddings, weighted by two hyperparameters to control the effect of each embedding on the final targets, i.e.:

$$\hat{\mathbf{y}}_k = \alpha \cdot g(\mathbf{y}_k) + \beta \cdot \sigma(\mathbf{z}_k/T) + (1 - \alpha - \beta) \cdot \mathbf{y}_k \tag{15}$$

4. Experimental Study

4.1. Experiments Setup

Datasets. We conducted experiments on CIFAR10, CIFAR100 [2], Fashion MNIST [27], STL10 [28] and SVHN [29] datasets. The CIFAR-10 dataset consists of 60000 RGB images of size 32×32, spanning over 10 classes, i.e., with 6000 images per class. The training set contains 50000 and the remaining 10000 images constitue the test set. The CIFAR100 dataset is similar, except it contains 100 classes with 600 images each, 500 of which are used for training and the remaining 100 are used for testing. The STL10 dataset is similar to the CIFAR10 dataset, but each class has fewer labeled training examples. There are 10 classes in this dataset, the images are of size

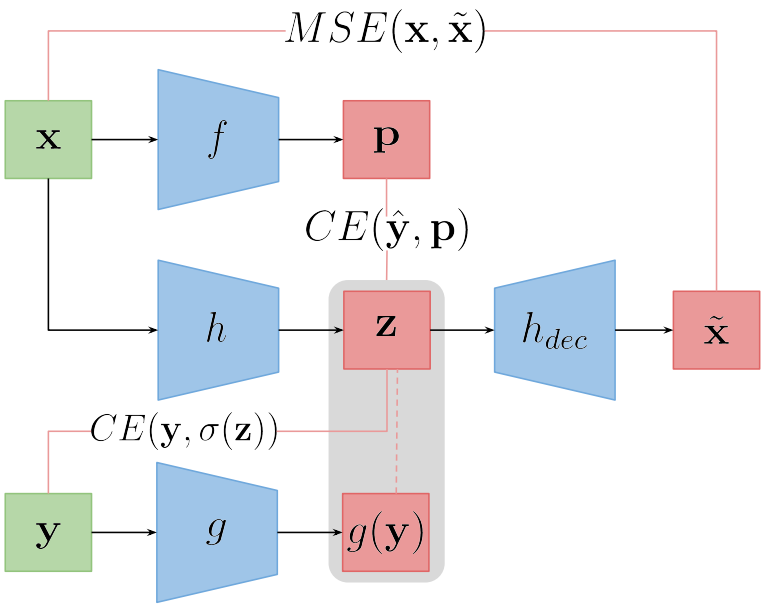


Figure 4: Schematic representation of the learning procedure of a classification network attached with the combination of the proposed general-class and instance-specific label embedding networks.

96×96 and colour, and there are 500 training images and 800 test images per class. The SVHN dataset is a real-world image dataset for digit recognition. There are 10 classes in the dataset, the digits are cropped and resized to 32×32, and in total there are 73257 digits for training, and 26032 digits for testing. Finally, the FashionMNIST dataset consists of a training set of 60000 samples and a test set of 10000 samples. Each sample is a 28×28 grayscale image and there are 10 classes in this dataset, corresponding to clothing categories.

Networks. Four types of networks are used based on the ResNet [30] architecture: a ResNet-8, a ResNet-18, a ResNet-6 model, and a version of the ResNet-6 net with fewer channels, specifically using only a quarter of the learnable filters (ResNet-6l0.25). The models are chosen for their parameter efficiency, and the two most lightweight ones were chosen so that they can run at real-time on embedded devices for HD inputs.

Hyperparameters. We train all networks for 100 epochs for all datasets, starting with a learning rate of 0.1 and dividing it by 0.1 every 25 epochs. For α, β we use 0.1 for the class level and 0.2 instance level similarities, while setting the temperature T to 10. The AE architecture used is based on a ResNet-18 encoder and similar decoder, and pretrained on the dataset for 25 epochs.

4.2. Experimental Results

The results of our experiments are summarized in Tables 1,2,3 and 4 for the ResNet6-l0.25, ResNet-6, ResNet-8 and ResNet-18 networks and all datasets. The general-class method is denoted as GC, the instance-specific case as IS and a combination of the two is also investigated and denoted as GC+IS. The proposed variants lead to increases in accuracy in all cases over the baseline models. Note that, despite the increased training cost, the cost during deployments remains the same as that of the baseline models. All experiments are run 5 times and performance is noted in terms of mean accuracy and the corresponding standard deviation.

Table 1: ResNet-6l0.25 results on all datasets in terms of classification accuracy.

Dataset	Baseline	GC	IS	GC+IS
SVHN	94.18 ± 0.11	$\mathbf{94.51} \pm 0.17$	$\mathbf{94.58} \pm 0.05$	$\underline{\mathbf{94.66}} \pm 0.18$
FashionMNIST	92.62 ± 0.14	$\mathbf{92.81} \pm 0.12$	$\underline{\mathbf{93.19}} \pm 0.06$	$\mathbf{93.08} \pm 0.09$
CIFAR10	87.28 ± 0.11	$\mathbf{87.66} \pm 0.21$	$\underline{\mathbf{88.35}} \pm 0.31$	$\mathbf{88.10} \pm 0.29$
STL10	67.76 ± 0.77	$\mathbf{68.28} \pm 1.19$	$\mathbf{70.39} \pm 0.55$	$\underline{\mathbf{70.40}} \pm 0.63$
CIFAR100	61.55 ± 0.39	$\mathbf{61.82} \pm 0.19$	$\mathbf{62.44} \pm 0.29$	$\underline{\mathbf{62.48}} \pm 0.15$

For the lightweight ResNet-6l0.25 model, all of the proposed variants improve the performance in terms of accuracy over the baseline training, i.e., using standard cross entropy. Furthermore, the instance-specific methods outperform the general-class ones. Note that, GC+IS, despite offering slightly improved mean accuracy, overall leads to less stable performance compared to IS, as indicated by the larger standard deviation in all datasets except for CIFAR100.

The results are similar for the ResNet-6 model. The proposed GC, IS and GC+IS outperform the baseline training for all datasets. The combined GC+IS variant exceeds the performance of each of the GC, IS methods only

on the CIFAR100 dataset in this case. Note that this classifier contains four times as many learnable parameters as the ResNet-6l0.25 model, which is visible in the overall better results as well as larger improvements of the proposed variants over the baseline training, i.e., +3.41% for STL10, compared to +2.64% for the same dataset using ResNet-6l0.25, and +1.49 for CIFAR100 compared to +0.93 for ResNet-6l0.25.

Table 2: ResNet-6 results on all datasets in terms of classification accuracy.

Dataset	Baseline	GC	IS	GC+IS
SVHN	95.05 ± 0.37	$\underline{\mathbf{95.67}} \pm 0.14$	$\mathbf{95.40} \pm 0.09$	$\mathbf{95.44} \pm 0.09$
FashionMNIST	93.09 ± 0.14	$\mathbf{93.40} \pm 0.18$	$\underline{\mathbf{93.81}} \pm 0.15$	$\mathbf{93.78} \pm 0.14$
CIFAR10	90.78 ± 0.09	$\mathbf{91.27} \pm 0.30$	$\underline{\mathbf{92.55}} \pm 0.25$	$\mathbf{92.51} \pm 0.09$
STL10	72.21 ± 0.94	$\mathbf{72.61} \pm 1.26$	$\underline{\mathbf{75.62}} \pm 0.19$	$\mathbf{75.45} \pm 0.31$
CIFAR100	69.76 ± 0.32	$\mathbf{70.34} \pm 0.37$	$\mathbf{70.97} \pm 0.26$	$\underline{\mathbf{71.25}} \pm 0.10$

Moving on to the ResNet-8 model, the performance of the baseline model is significantly better than the previous, more lightweight models. Furthermore, all proposed methods improve upon that baseline. The GC variant slightly outperforms IS and GC+IS, although performance is generally more stable when using the IS method, as indicated by the lower standard deviations with FashionMNIST being the only exception.

Table 3: ResNet-8 results on all datasets in terms of classification accuracy.

Dataset	Baseline	GC	IS	GC+IS
SVHN	95.59 ± 0.06	$\mathbf{95.76} \pm 0.08$	$\underline{\mathbf{95.90}} \pm 0.06$	$\mathbf{95.86} \pm 0.04$
FashionMNIST	93.80 ± 0.12	$\underline{\mathbf{94.12}} \pm 0.05$	$\mathbf{93.98} \pm 0.14$	$\mathbf{94.03} \pm 0.07$
CIFAR10	93.26 ± 0.15	$\underline{\mathbf{93.94}} \pm 0.13$	$\mathbf{93.93} \pm 0.13$	$\mathbf{93.80} \pm 0.16$
STL10	73.94 ± 0.60	$\mathbf{74.94} \pm 0.42$	$\underline{\mathbf{78.03}} \pm 0.19$	$\mathbf{77.81} \pm 0.26$
CIFAR100	73.81 ± 0.29	$\underline{\mathbf{74.69}} \pm 0.19$	$\mathbf{74.37} \pm 0.10$	$\mathbf{74.24} \pm 0.26$

Finally, the deepest network, namely ResNet-18, achieves overall better baseline results than the previous models. For this model, we also evaluate the proposed methods on the larger and more challenging Caltech-101 [31] and Tiny-Imagenet datasets [32]. Once again, all of the proposed variants improve the performance of the model on all datasets. Notice that the

combined GC+IS method performs better than the GC and IS methods for three out of six datasets for this model. The performance gain is greater in the CIFAR10, CIFAR100, STL10, Caltech-101 and Tiny-ImageNet datasets, and smaller for SVHN and FashionMNIST, although it is consistent over the baseline training in all datasets and networks.

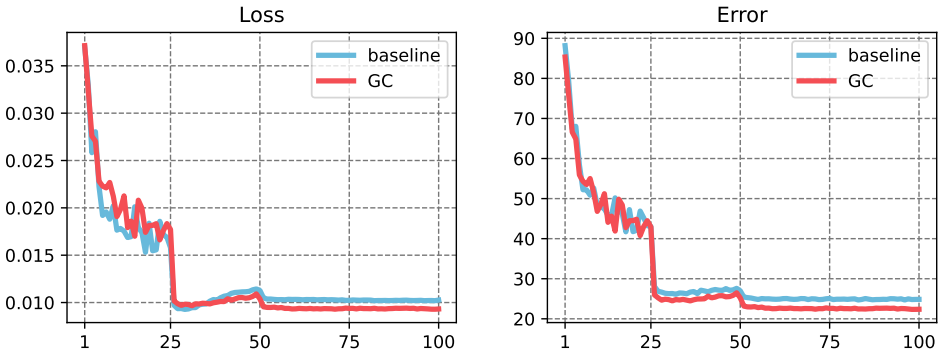
Table 4: ResNet-18 results on all datasets in terms of classification accuracy.

Dataset	Baseline	GC	IS	GC+IS
SVHN	95.88 ± 0.08	$\mathbf{\underline{96.30}} \pm 0.13$	$\mathbf{96.24} \pm 0.07$	$\mathbf{96.21} \pm 0.06$
FashionMNIST	94.03 ± 0.18	$\mathbf{94.07} \pm 0.15$	$\mathbf{\underline{94.42}} \pm 0.10$	$\mathbf{94.31} \pm 0.12$
CIFAR10	94.23 ± 0.10	$\mathbf{94.38} \pm 0.07$	$\mathbf{94.30} \pm 0.07$	$\mathbf{\underline{94.40}} \pm 0.03$
STL10	80.65 ± 0.27	$\mathbf{\underline{82.01}} \pm 0.21$	$\mathbf{81.88} \pm 0.33$	$\mathbf{81.97} \pm 0.48$
CIFAR100	75.74 ± 0.30	$\mathbf{77.15} \pm 0.18$	$\mathbf{77.27} \pm 0.32$	$\mathbf{\underline{77.40}} \pm 0.20$
Caltech-101	60.29 ± 0.55	$\mathbf{61.53} \pm 1.32$	$\mathbf{64.19} \pm 1.12$	$\mathbf{\underline{64.55}} \pm 1.35$
Tiny-ImageNet	58.45 ± 0.18	$\mathbf{60.58} \pm 0.28$	$\mathbf{62.26} \pm 0.26$	$\mathbf{\underline{62.44}} \pm 0.10$

As mentioned, the combination of the two methods is riskier than each of the methods used separately, and it seems to work better when the classifier used has a higher learning capacity. This can be attributed to two factors. First, to the better performance of the deeper baseline networks, corresponding to a deeper knowledge of the datasets. Second, the networks with more parameters have greater capacity for learning complex relationships.

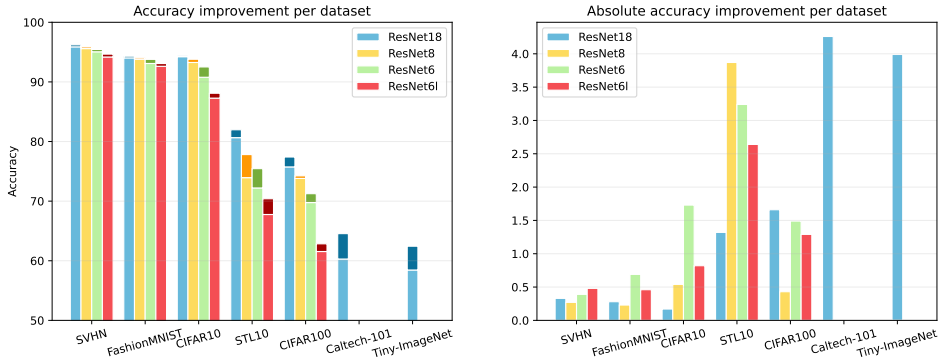
Figures 5a and 5b show the loss and accuracy error progression during training of the ResNet-18 model on the CIFAR100 dataset, for the baseline and GC methods. The proposed method improves accuracy in the long run, even if the classification loss (but not the corresponding accuracy error) is lower at some point during training for the baseline method. Overall, the results indicate that the proposed method works well on complex datasets with multiple classes, and especially when the classes are semantically related. Figures 6a and 6b show the accuracy improvement for each network, and the datasets have been arranged in increasing difficulty. The improvement that the proposed method offers is larger for the more complex datasets, and is larger in general for the smaller networks than it is for the heaviest ResNet-18.

We finally compare the performance of the proposed method with various state-of-the-art soft label methods, in the CIFAR100 dataset. The results are



(a) (b)

Figure 5: (a) Loss curves and (b) accuracy error progression, for baseline and GC methods, using ResNet-18 on CIFAR100.



(a) (b)

Figure 6: (a) Baseline and GC+IS accuracy, and (b) absolute accuracy improvement, for baseline and GC+IS methods.

presented in Table 5, in terms of baseline accuracy, accuracy using each compared method, and relative improvement in accuracy, to account for different training hyperparameters for each method. The proposed method provides a significant improvement, and the best result on this dataset, compared to similar methods.

4.3. Label Visualization

Figures 7 and 8 are visualizations, via t-SNE, of the soft labels acquired by the proposed IS method for CIFAR10 and Tiny-ImageNet respectively. In

Table 5: Comparative improvement in accuracy for the CIFAR100 dataset.

Network	Method	Baseline Acc	Acc w/ Method
ResNet-18	Ours, GC		77.15
	Ours, IS	75.74	77.27
	Ours, GC+IS		77.40
	Tf-KD [14]	75.87	77.10
	LabelEmb [22]	72.65	76.03
ResNet-56	OLS [19]	74.73	76.09
	LS [15]	72.1	72.7

Figure 7, only the soft embeddings are shown, as opposed to the final targets which are a combination of these with the groundtruth one-hot encodings of the classes. From uncorrelated, binary representations, the labels now have distributions which are in agreement with the feature visualizations shown in Figure 1. In other words, the labels capture the information we aimed to embed in them, encoding instance-specific similarities as well as general class similarities. Furthermore, clusters appear within each class distribution, which is more intuitive than a harsh binary representation.

Figure 8 is a visualization of the targets set for the IS scenario on the Tiny-ImageNet dataset. Only the first twenty classes are shown for better visualization. Notice that class-level similarities are also captured, i.e., the "tarantula" and "black widow" classes are close together, as are the "brain coral" and "jellyfish" ones. Some instances of these classes also form their own clusters within the confines of their class.

Finally, Figure 9 is a visualization of the GC targets pre-softmax for 30 classes of the Caltech-101 dataset. The classes shown are sorted by their contribution to the "*Face*" class, resulting in this specific pattern. Note that the method is able to capture the similarity between intuitively similar classes, with the most notable example being the classes "*Faces*" and "*Faces_easy*". Intuitively, a classifier confusing these two classes for each other should be punished less than it would for confusing them with other completely unrelated classes, an intuition that is captured by the proposed framework.

Figure 7: t-SNE visualization of soft labels generated by the proposed instance-specific method for the CIFAR10 dataset. Note that the final targets are a combination of these labels and the groundtruth one-hot encodings.

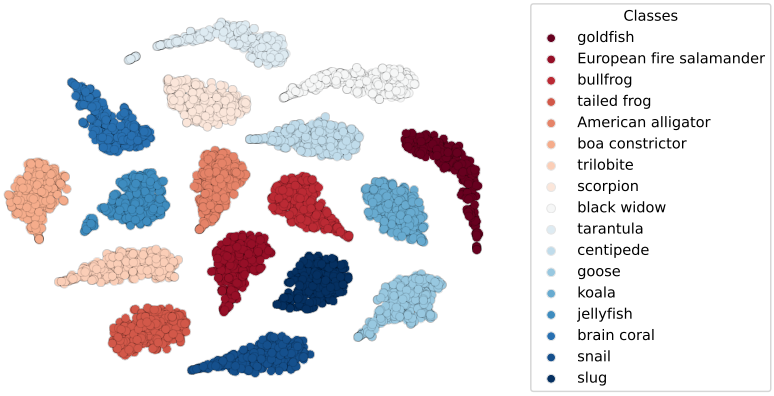


Figure 8: t-SNE visualization of targets generated by the proposed instance-specific method for the first 20 classes of the Tiny ImageNet dataset.

5. Conclusions

In this paper, soft label embedding methods were studied and two variants were proposed, with an aim to capture both general-class resemblances

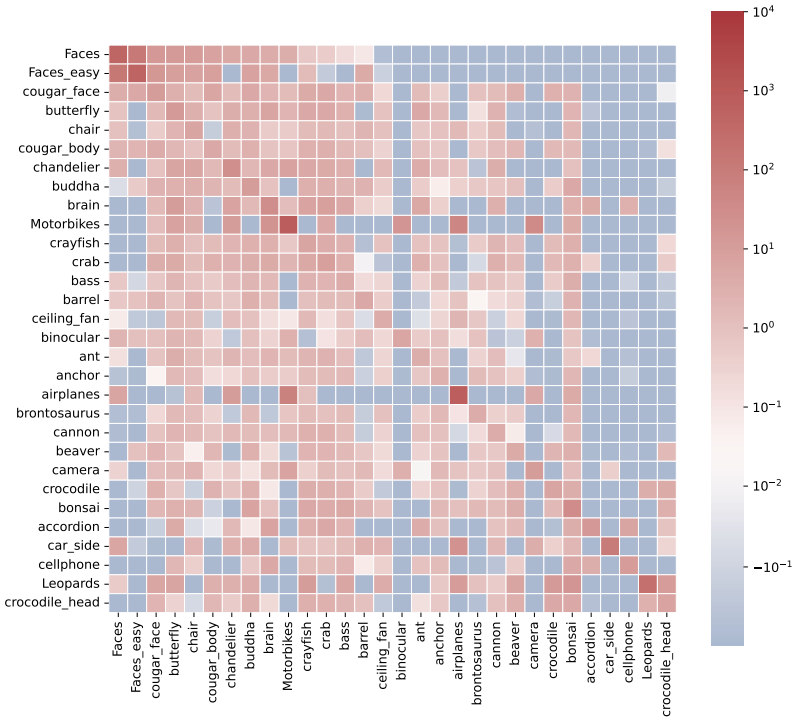


Figure 9: Caltech101 generic class similarities as captured by the proposed GC method.

as well as instance-specific similarities, and to incorporate these into the training process of neural classifiers in order to ease the training process. For the general-class a simple linear network was used to generate labels which are a linear combination of the groundtruth one-hot encodings. As the training process progresses, the network learns to capture general correlations between the classes and uses these relationships to soften the groundtruth labels essentially blurring the classification boundaries between them. In the instance-specific case, the addition of a decoder network and corresponding reconstruction error leads the logits learned by the classifier to retain information necessary for the reconstruction, which acts as a regularizer upon the class labels. The proposed methods were shown experimentally to lead to increased performance in various networks and datasets, both in the general-class and instance-specific cases, as well as in a combination of the two.

Acknowledgements

This work was supported by the European Union's Horizon2020 Research and Innovation Program (OpenDR) under Grant 871449. This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[2] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images (2009).

[3] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (11) (2008).

[4] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[5] S. Sukhbaatar, J. Bruna, M. Paluri, L. Bourdev, R. Fergus, Training convolutional networks with noisy labels, arXiv preprint arXiv:1406.2080 (2014).

[6] G. Algan, I. Ulusoy, Metalabelnet: Learning to generate soft-labels from noisy-labels, arXiv preprint arXiv:2103.10869 (2021).

[7] N. El Gayar, F. Schwenker, G. Palm, A study of the robustness of knn classifiers trained using soft labels, in: IAPR Workshop on Artificial Neural Networks in Pattern Recognition, Springer, 2006, pp. 67–80.

[8] H. M. Fayek, M. Lech, L. Cavedon, Modeling subjectiveness in emotion recognition with deep neural networks: Ensembles vs soft labels, in: 2016 international joint conference on neural networks (IJCNN), IEEE, 2016, pp. 566–570.

[9] R. Diaz, A. Marathe, Soft labels for ordinal regression, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4738–4747.

[10] N. Vyas, S. Saxena, T. Voice, Learning soft labels via meta learning, arXiv preprint arXiv:2009.09496 (2020).

[11] M. Tzelepi, A. Tefas, Efficient training of lightweight neural networks using online self-acquired knowledge distillation, in: 2021 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2021, pp. 1–6.

[12] M. Tzelepi, N. Passalis, A. Tefas, Online subclass knowledge distillation, Expert Systems with Applications 181 (2021) 115132.

[13] Z. Zhang, X. Shu, B. Yu, T. Liu, J. Zhao, Q. Li, L. Guo, Distilling knowledge from well-informed soft labels for neural relation extraction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 9620–9627.

[14] L. Yuan, F. E. Tay, G. Li, T. Wang, J. Feng, Revisiting knowledge distillation via label smoothing regularization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3903–3911.

[15] R. Müller, S. Kornblith, G. Hinton, When does label smoothing help?, arXiv preprint arXiv:1906.02629 (2019).

[16] M. Lukasik, S. Bhojanapalli, A. Menon, S. Kumar, Does label smoothing mitigate label noise?, in: International Conference on Machine Learning, PMLR, 2020, pp. 6448–6458.

[17] Y. Xu, Y. Xu, Q. Qian, H. Li, R. Jin, Towards understanding label smoothing, arXiv preprint arXiv:2006.11653 (2020).

[18] B. Chen, L. Ziyin, Z. Wang, P. P. Liang, An investigation of how label smoothing affects generalization, arXiv preprint arXiv:2010.12648 (2020).

[19] C.-B. Zhang, P.-T. Jiang, Q. Hou, Y. Wei, Q. Han, Z. Li, M.-M. Cheng, Delving deep into label smoothing, IEEE Transactions on Image Processing 30 (2021) 5984–5996.

[20] H. Zhang, L. Xiao, W. Chen, Y. Wang, Y. Jin, Multi-task label embedding for text classification, arXiv preprint arXiv:1710.07210 (2017).

[21] Y. Dong, P. Liu, Z. Zhu, Q. Wang, Q. Zhang, A fusion model-based label embedding and self-interaction attention for text classification, IEEE Access 8 (2019) 30548–30559.

[22] X. Sun, B. Wei, X. Ren, S. Ma, Label embedding network: Learning label representation for soft training of deep networks, arXiv preprint arXiv:1710.10393 (2017).

[23] C. Chen, H. Wang, W. Liu, X. Zhao, T. Hu, G. Chen, Two-stage label embedding via neural factorization machine for multi-label classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 3304–3311.

[24] C. Liu, J. JaJa, Class-similarity based label smoothing for generalized confidence calibration, arXiv preprint arXiv:2006.14028 (2020).

[25] Z. Zhang, M. R. Sabuncu, Self-distillation as instance-specific label smoothing, arXiv preprint arXiv:2006.05065 (2020).

[26] P. Nousi, A. Tefas, Deep learning algorithms for discriminant autoencoding, Neurocomputing 266 (2017) 325–335.

[27] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017). arXiv:cs.LG/1708.07747.

[28] A. Coates, A. Ng, H. Lee, An analysis of single-layer networks in unsupervised feature learning, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning (2011).

[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[31] F.-F. Li, M. Andreeto, M. Ranzato, P. Perona, Caltech 101 (Apr 2022). doi:10.22002/D1.20086.

[32] Y. Le, X. Yang, Tiny imagenet visual recognition challenge, CS 231N 7 (7) (2015) 3.

7.3 Variational Voxel Pseudo Image Tracking

The appended paper [70] follows.

Variational Voxel Pseudo Image Tracking

Illia Oleksiienko*, Paraskevi Nousi†, Nikolaos Passalis†, Anastasios Tefas† and Alexandros Iosifidis*

*DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark
†Department of Informatics, Aristotle University of Thessaloniki, Greece

{io, ai}@ece.au.dk {paranous, passalis, tefas}@csd.auth.gr

***Abstract*—Uncertainty estimation is an important task for critical problems, such as robotics and autonomous driving, because it allows creating statistically better perception models and signaling the model's certainty in its predictions to the decision method or a human supervisor. In this paper, we propose a Variational Neural Network-based version of a Voxel Pseudo Image Tracking (VPIT) method for 3D Single Object Tracking. The Variational Feature Generation Network of the proposed Variational VPIT computes features for target and search regions and the corresponding uncertainties, which are later combined using an uncertainty-aware cross-correlation module in one of two ways: by computing similarity between the corresponding uncertainties and adding it to the regular cross-correlation values, or by penalizing the uncertain feature channels to increase influence of the certain features. In experiments, we show that both methods improve tracking performance, while penalization of uncertain features provides the best uncertainty quality.**

***Index Terms*—3D Single Object Tracking, Point Cloud, Uncertainty Estimation, Bayesian Neural Networks, Variational Neural Networks**

I. Introduction

3D Singe Object Tracking (3D SOT) is the task of tracking an object in a 3D scene based on the given initial object position. This task combines challenges from both 3D Object Detection, as objects have to be accurately located in space, and 3D Multiple Object Tracking, as the object of interest has to be distinguished from similar objects. There is a variety of sensors that can be used for 3D SOT, including single or double camera setups, Lidar and Radar. While the camera setups are the cheapest option, they capture images which lack valuable for 3D SOT depth information, which can be provided by Lidar sensors. Lidars generate point clouds, which are sets of 3D points detected as the positions in the 3D scene of light beam reflections. The explicit depth information makes Lidar the most common choice for many 3D perception methods, including 3D SOT. The SOT is performed by predicting the offset of the object's position with respect to its previous known position. This has been approached by using correlation filters [1], [2], deep learning methods to directly predict the object's offset [3], or by using Siamese methods which search for the position with the highest similarity score [4]–[8]. Since 3D perception methods are often used in critical fields, such as robotics or autonomous driving, it is important to provide accurate predictions and confidence estimations to avoid costly damages.

Uncertainty estimation in neural networks allows for using the network's outputs to better indicate the confidence in its predictions and to improve their statistical qualities, leading to better performance. The practical applications of uncertainty estimation are studied for several perception tasks, including 3D Object Detection [9]–[11], 3D Object Tracking [12], [13], 3D Human Pose Tracking [14], and Steering Angle Prediction [15]. These methods provide an improvement in perception and control by using an uncertainty estimation process. However, most of these methods adopt single deterministic approaches to estimate different types of uncertainty, or use Monte Carlo Dropout (MCD) [16] as an approach to estimate epistemic uncertainty. According to experiments in [17] on the uncertainty quality of different types of Bayesian Neural Networks (BNNs), MCD achieves the worst uncertainty quality.

In this paper, we introduce a Variational Neural Network (VNN) [18] based version of the fastest 3D SOT method called Voxel Pseudo Image Tracking (VPIT) [8] and propose two ways, i.e., the uncertainty similarity approach and the penalization approach, to utilize the estimated uncertainty and improve the tracking performance of the model. The similarity-based approach computes a similarity between the estimated uncertainties to serve as an additional similarity score, while the penalization approach focuses on certain features by penalizing the feature values corresponding to high uncertainties. We train a VNN version of PointPillars for 3D Object Detection to serve as backbone for the proposed Variational VPIT (VVPIT) method. We, then, train the whole network following the VPIT's training procedure, but use the uncertainty-aware cross-correlation function and multiple samples of the Variational Feature Generation Network to compute uncertainty in the produced features. In experiments, we show that the use of uncertainty leads to an improvement in the model's tracking performance, and the choice of the penalty-based uncertainty utilization strategy leads to the highest improvement in Success and Precision metrics.

The remainder of the paper is structured as follows. Section II describes related and prior work. In Section III we describe the proposed approach, including the Variational TANet model and its training and the proposed uncertainty-aware AB3DMOt. Section IV outlines the experimental protocol and provides experimental results. Section V concludes this paper[1].

[1]Our code is available at gitlab.au.dk/maleci/opendr/vnn_vpit_opendr

II. Related Work

Gawlikowski et al. [19] define four main categories of uncertainty estimation methods, based on the strategies they use to estimate the uncertainty of the model. Deterministic Methods [12], [20] use a single deterministic network and either predict its uncertainty by using an additional regression branch, or estimate it by analyzing the output of the model. Bayesian Neural Networks (BNNs) [21], [22] consider a distribution over weights of the network and compute the outputs of multiple model samples for the same input. The variance in the network's outputs expresses the estimated uncertainty, while the mean of outputs is used as the prediction value. Ensemble Methods [23], [24] consider a categorical distribution over the weights of the network and train multiple models at once. Test-Time Data Augmentation methods [25]–[27] apply data augmentations commonly used in the training phase during the inference to pass distorted inputs to a single deterministic network and compute the variance in the model's outputs.

Variational Neural Networks [18], [28] are similar to BNNs, but instead of considering a distribution over weights, they place a Gaussian distribution over the outputs of each layer and estimate its mean and variance values by the corresponding sub-layers. All types of uncertainty estimation methods, except those in the Deterministic Methods category, use multiple model passes to compute the variance in the network's outputs. This means the Deterministic Methods generally have the lowest computational impact on the model, but they usually perform worse than other methods. The single deterministic network approach can be improved by considering the Bayesian alternative, as it can be seen as a case of BNNs with the simple Dirac delta distribution over weights, which places the whole distributional mass on a single weight point.

The 3D SOT task is usually approached by using point-based Siamese networks, which consider a pair of target and search regions, predict a position of the target region inside the search region and compute the object offset relative to the previous object position. P2B [29], BAT [30], Point-Track-Transformer (PTT) [31], [32] and 3D-SiamRPN [4] use point-wise Siamese networks and predict object positions based on the comparison of target and search point clouds. 3D Siam-2D [33] uses one Siamese network in a 2D Birds-Eye-View (BEV) space to create fast object proposals and another Siamese network in 3D space to select the true object proposal and regress the bounding box. Voxel Pseudo Image Tracking (VPIT) [8] uses voxel pseudo images in BEV space and deploys a SiamFC-like module [5] to extract and compare features from target and search regions. Instead of using different scales, VPIT uses a multi-rotation search to find the correct vertical rotation angle.

Bayesian YOLO [34] is a 2D object detection method that estimates uncertainty by combining Monte Carlo Dropout (MCD) [16] with a deterministic approach and predicts aleatoric uncertainty with a special regression branch, while computing the epistemic uncertainty from the variance in MCD model predictions. Feng et al. [9] use a Lidar-based 3D object detection method and estimate the uncertainty in the predictions of the model in a similar way to Bayesian YOLO, by using a partially MCD model for the epistemic uncertainty estimation and using a separate regression branch for the aleatoric uncertainty estimation. LazerNet [10] predicts the uncertainty of a 3D bounding box using a single deterministic network and utilizes the predicted uncertainty during the non-maximum suppression process. This approach is further improved by estimating the ground truth labels' uncertainty based on the IoU between the 3D bounding box and the convex hull of the enclosed point cloud, and using the provided uncertainties during the training process [11].

Zhong et al. [12] perform 3D Multiple Object Tracking (MOT) by using a single deterministic network for 3D Object Detection to predict the uncertainty in outputs and providing the estimated uncertainties to the tracker by replacing the unit-Gaussian measurement noise in Kalman filter [35] with the predicted uncertainties. Uncertainty-Aware Siamese Tracking (UAST) [36] performs 2D single object tracking by using a single deterministic network and computing the distribution over the outputs by quantizing over the specific range of values and predicting the softmax score for each quantized value. The final regression value is computed as an expectation of the corresponding quantized distribution, and the distributions are used to estimate better confidence scores and select the best box predictions.

To the best of our knowledge, there are no methods that utilize uncertainty for 3D Single Object Tracking. Moreover, the estimation of uncertainty for related tasks, such as 2D Single Object Tracking, 3D Multiple Object Tracking or 3D Object Detection, is based on single deterministic networks or MCD, despite the fact that the statistical quality of single deterministic networks can be improved by using a Bayesian alternative, and that MCD tends to produce the worst quality of uncertainty between BNNs [17].

III. Methodology

Voxel Pseudo Image Tracking (VPIT) uses PointPillars [37] as a backbone to create voxel pseudo images and to process them with a Feature Generation Network (FGN), which consists of the convolutional part of the PointPillars' Region Proposal Network. The search and target features are compared with a convolutional cross-correlation function that calculates a pixel-wise similarity map. The highest value in this similarity map is used to determine the object position offset between frames. The structure of VPIT is present on Fig. 1.

We train a Variational VPIT (VVPIT) by replacing the FGN subnetwork with a Variational Neural Network (VNN) [18], [28] based version of it, i.e., we create a Variational FGN (VFGN). We use multiple samples of the network for each input to compute mean and variance for the output features. The number of samples can be dynamic and is not required to be the same during training and inference. For each of target and search regions, VFGN produces a set of outputs in the form $Y = \{y_i, i \in [1, \ldots, P]\}$ which

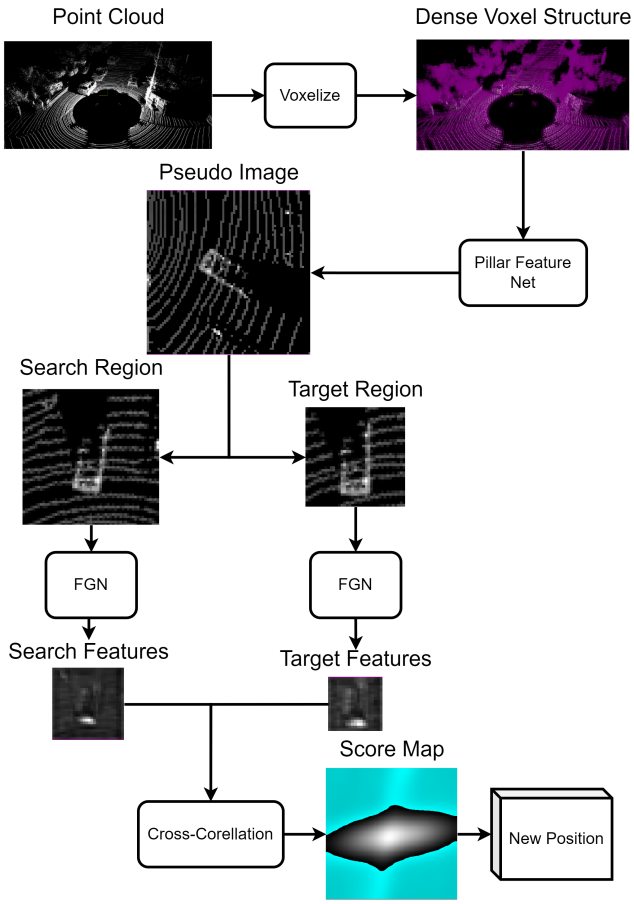


Fig. 1: Voxel Pseudo Image Tracking structure.

correspond to the outputs of P sampled VFGN models, with $Y^s = \{y_i^s, i \in [1, \dots, P]\}$ corresponding to the search region output set and $Y^t = \{y_i^t, i \in [1, \dots, P]\}$ to the target region output set. The number of samples P can be different for each set, but for simplicity, we use the same number of samples for both target and search regions. The mean and variance of the outputs are computed as follows:

$$\begin{aligned} y_m^s &= \frac{1}{P}\sum_i^P y_i^s, \\ y_m^t &= \frac{1}{P}\sum_i^P y_i^t, \\ y_v^s &= \mathrm{diag}\left(\frac{1}{P}\sum_i^P (y_i^s - y_m^s)(y_i^s - y_m^s)^T\right), \\ y_v^t &= \mathrm{diag}\left(\frac{1}{P}\sum_i^P (y_i^t - y_m^t)(y_i^t - y_m^t)^T\right), \end{aligned} \tag{1}$$

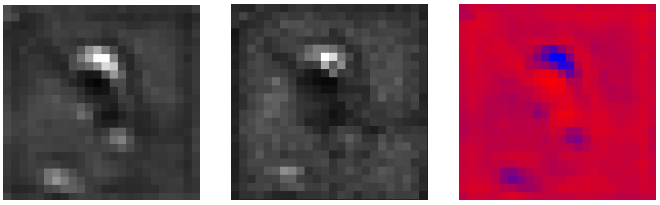
where y_m^s, y_v^s and y_m^t, y_v^t are the mean and variance values of search and target output sets, respectively, and $\mathrm{diag}(\cdot)$ is a function that returns the main diagonal of a matrix. Fig. 2 shows an example of the mean and variance values of features generated by the VFGN for a search region with a car in the center. The background pixels have mostly high certainty, as all sampled models agree on them being irrelevant. The high magnitude features at the top part of the car have the highest uncertainty, as different model samples can disagree on the details in the appearance of the object.

The proposed VVPIT method can utilize the predicted uncertainties in different ways. The simplest way is to entirely ignore the uncertainty values and process the mean outputs only with the regular cross-correlation function $g(a, b)$, defined as a 2D convolution $\mathrm{conv2D}_{\omega=b}(a)$ with ω being the kernel weights. This still leads to a statistically better model which can provide better predictions, but it can be further improved by utilizing the predicted uncertainties in the cross-correlation module. Since most 3D SOT methods compare region features in a similarity manner, we focus on similarity-based approaches to use the uncertainty values, instead of applying distance-based approaches. We propose a double similarity-based process to utilize uncertainty, which treats mean and variance values as separate feature sets and uses the convolutional similarity function $g(a, b)$ on both of them independently. The final similarity value $\hat{g}_{\mathrm{double}}$ is obtained by linearly the similarities of the mean and variance of the outputs as follows:

$$\hat{g}_{\mathrm{double}}(y_m^s, y_m^t, y_v^s, y_v^t) = g(y_m^s, y_m^t) + \lambda g(y_v^s, y_v^t), \tag{2}$$

where λ is a variance weight hyperparameter. This approach is based on the idea that positions with similar uncertainties should be prioritized, as there is a high chance of them representing the same object. Humans can also treat uncertainties as separate features. Let us consider a task of classifying triangle and circle images, where some objects are rounded triangles. Based on the deformation degree, people will have different values of aleatoric uncertainty in their predictions, as they will have harder time classifying rounded triangles as only one of the two classes. If a person is asked to track these objects, the aleatoric uncertainty in predictions may be the only feature needed to distinguish between objects, given that size, thickness and other features are identical. This is achieved by describing the tracked objects as "definitely a circle", "triangle with some curves", "in between the circle and the triangle", which leads to low chances of mixing up these objects during tracking. The same principle can be applied for Lidar-based 3D SOT task. However, there are many different sources of uncertainty, considering the varying point cloud density, possible occlusions and object rotation. Some parts of the object of interest may have uncertain features, and this uncertainty is likely to be preserved during the tracking process.

In addition to the above approach, we also define an uncertainty penalization process which places focus on features with higher certainty and penalizes the uncertain feature values. This is achieved by dividing each mean feature value during the convolutional process by the corresponding normalized



(a) Mean (b) Variance (c) Mean & certainty

Fig. 2: An example of (a) mean, (b) variance and (c) mean and certainty features of a search region with a car in the center. Lighter color in the mean and variance images corresponds to higher values. Red color channel represents certainty in the corresponding pixel values, and blue color channel represents the mean feature values. The purple color indicates that the feature values and the certainty in those values is equally high, while the blue pixels signal features with high values and low certainty.

variance score, as follows:

$$\begin{aligned} &\forall c,\ \ v_n^c(v) = (\rho - 1)\frac{v^c - \min(v^c)}{\max(v^c) - \min(v^c)} + 1, \\ &\forall p_x, \forall p_y,\ \ \hat{g}_{\text{pen}}(y_m^s, y_m^t, y_v^s, y_v^t)^{p_x,p_y} = \\ &\qquad = \frac{2\tilde{y_m^s}^{\,p_x,p_y}\tilde{y_m^t}^{\,p_x,p_y}}{v_n(\tilde{y_v^s})^{p_x,p_y} + v_n(\tilde{y_v^t})^{p_x,p_y}}, \end{aligned} \tag{3}$$

where the $v_n(v)$ function is used to normalize the variance predictions by the channel-wise minimum and maximum values to be in $[1, \rho]$ range, with a hyperparameter ρ that defines how much the uncertain predictions are penalized, $v_n^c(v)$ implements the normalization procedure for a single channel c. For an input j, $\tilde{j}$ represents the tensor with convolutional patches of j, and j^{p_x,p_y} corresponds to the values of j at position (p_x, p_y).

We follow the VPIT's training protocol and initialize a VVPIT model based on the VNN version of PointPillars for 3D Object Detection. After the initialization, the model is trained with the Binary Cross-Entropy (BCE) loss between the ground truth and the predicted score maps. Multiple VFGN samples are used during both training and inference to compute the mean and the variance in the target and search region features, which are later combined by using an uncertainty-aware cross-correlation module using one of the processes described above.

IV. Experiments

We use the KITTI [38] tracking dataset to train and test models. Following the standard protocol, we use KITTI tracking training subset for both training and testing, as the test subset does not provide the initial ground truth positions. The tracks $[0, \ldots, 18]$ are used for training and validation, and tracks 19 and 20 are used to test the trained models. Model performance is computed using the Precision and Success [39] metrics, which are based on the predicted and ground truth objects' center difference and 3D Intersection Over Union, respectively. VPIT uses a pre-trained PointPillars network to initialize its pseudo image generation and FGN modules. To follow the same procedure, we train a VNNs version of PointPillars on the KITTI [38] detection dataset, use it to initialize the VPIT model and train the corresponding model for $64,000$ steps with different number of training VFGN samples per step in $[1, \ldots, 20]$ range.

TABLE I: Precision and Success values on the KITTI single object tracking experiments for VPIT and Variational VPIT (VVPIT) models.

Method	Uncertainty	Success	Precision
VPIT	-	50.49	64.53
VVPIT	averaging	51.97	66.69
VVPIT	double similarity	52.62	66.56
VVPIT	uncertainty penalization	**53.30**	**67.79**

Table I contains the evaluation results of regular VPIT and the Variational VPIT (VVPIT) models with different ways to utilize the predicted uncertainty. We report the best-performing models for each uncertainty utilization process, which are obtained by using 20 samples of the VFGN module. By computing the average of predictions and discarding the variances, VVPIT achieves higher tracking performance compared to the VPIT model. By utilizing uncertainties, the Success and Precision values are further improved. Both double similarity and uncertainty penalization processes lead to better models, but the penalization process leads to a better tracking performance.

V. Conclusions

In this paper, we proposed a method to utilize uncertainty in 3D Single Object Tracking which uses a Variational Neural Network (VNN) based version of the VPIT 3D Single Object Tracking method to estimate uncertainty in target and search features and combines these features with an uncertainty-aware cross-correlation module. We proposed two ways to utilize uncertainty in cross-correlation, i.e., by double similarity which adds a similarity in uncertainties to the regular cross-correlation, and by uncertainty penalization which penalizes uncertain features to shift focus to the more reliable feature channels. Additionally, we tested the model's performance without exploiting the estimated uncertainties, as it still leads to a statistically better model compared to regular VPIT. The use of VNNs improves the tracking performance of VPIT in all cases, with the uncertainty penalization leading to the best Success and Precision values.

Acknowledgement

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui, "Visual object tracking using adaptive correlation filters," in *CVPR*, 2010, pp. 2544–2550.

[2] Joao F. Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista, "High-speed tracking with kernelized correlation filters," *TPAMI*, vol. 37, no. 3, pp. 583–596, 2015.

[3] David Held, Sebastian Thrun, and Silvio Savarese, "Learning to track at 100 fps with deep regression networks," *1604.01802*, 2016.

[4] Zheng Fang, Sifan Zhou, Yubo Cui, and Sebastian Scherer, "3d-siamrpn: An end-to-end learning method for real-time 3d single object tracking using raw point cloud," *Sensors*, vol. 21, no. 4, pp. 4995–5011, 2021.

[5] Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip HS Torr, "Fully-convolutional siamese networks for object tracking," *arXiv:1606.09549*, 2016.

[6] Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu, "High performance visual tracking with siamese region proposal network," in *CVPR*, 2018, pp. 8971–8980.

[7] Bo Li, Wei Wu, Qiang Wang, Fangyi Zhang, Junliang Xing, and Junjie Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," *arXiv:1812.11703*, 2018.

[8] Illia Oleksiienko, Paraskevi Nousi, Nikolaos Passalis, Anastasios Tefas, and Alexandros Iosifidis, "Vpit: Real-time embedded single object 3d tracking using voxel pseudo images," *arXiv:2206.02619*, 2022.

[9] Di Feng, Lars Rosenbaum, and Klaus Dietmayer, "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection," in *ITSC*, 2018, pp. 3266–3273.

[10] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," in *CVPR*, 2019, pp. 12677–12686.

[11] Gregory P. Meyer and Niranjan Thakurdesai, "Learning an uncertainty-aware object detector for autonomous driving," in *IROS*, 2020, pp. 10521–10527.

[12] Yuanxin Zhong, Minghan Zhu, and Huei Peng, "Uncertainty-aware voxel based 3d object detection and tracking with von-mises loss," *arXiv:2011.02553*, 2020.

[13] Jianren Wang, Siddharth Ancha, Yi-Ting Chen, and David Held, "Uncertainty-aware self-supervised 3d data association," in *RSJ*, 2020, pp. 8125–8132.

[14] Ben Daubney and Xianghua Xie, "Tracking 3d human pose with large root node uncertainty," in *CVPR*, 2011, pp. 1321–1328.

[15] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza, "A general framework for uncertainty estimation in deep learning," *RA-L*, vol. 5, no. 2, pp. 3153–3160, 2020.

[16] Yarin Gal and Zoubin Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *JMLR*, 2016, vol. 48, pp. 1050–1059.

[17] Ian Osband, Zheng Wen, Mohammad Asghari, Morteza Ibrahimi, Xiyuan Lu, and Benjamin Van Roy, "Epistemic Neural Networks," *arXiv:2107.08924*, 2021.

[18] Illia Oleksiienko, Dat Thanh Tran, and Alexandros Iosifidis, "Variational neural networks," *arxiv:2207.01524*, 2022.

[19] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu, "A survey of uncertainty in deep neural networks," *arxiv:2107.03342*, 2021.

[20] Murat Sensoy, Lance Kaplan, and Melih Kandemir, "Evidential deep learning to quantify classification uncertainty," in *NeurIPS*, 2018, p. 3183–3193.

[21] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight Uncertainty in Neural Networks," in *JMLR*, 2015, vol. 37, pp. 1613–1622.

[22] Martin Magris and Alexandros Iosifidis, "Bayesian learning for neural networks: an algorithmic survey," *Artificial Intelligence Review*, 2023.

[23] Ian Osband, John Aslanides, and Albin Cassirer, "Randomized prior functions for deep reinforcement learning," in *NeurIPS*, 2018, vol. 31, pp. 8626–8638.

[24] Matias Valdenegro-Toro, "Deep sub-ensembles for fast uncertainty estimation in image classification," *arxiv:1910.08168*, 2019.

[25] Guotai Wang, Wenqi Li, Sébastien Ourselin, and Tom Vercauteren, "Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation," in *BrainLes*. 2018, vol. 11384, pp. 61–72, Springer.

[26] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren, "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks," *Neurocomputing*, vol. 338, pp. 34–45, 2019.

[27] Ibrahem Kandel and Mauro Castelli, "Improving convolutional neural networks performance for image classification using test time augmentation: a case study using MURA dataset," *Health Inf. Sci. Syst.*, vol. 9, no. 1, pp. 33, 2021.

[28] Illia Oleksiienko, Dat Thanh Tran, and Alexandros Iosifidis, "Variational neural networks implementation in pytorch and jax," *Software Impacts*, vol. 14, pp. 100431, 2022.

[29] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao, "P2b: Point-to-box network for 3d object tracking in point clouds," *arXiv:2005.13888*, 2020.

[30] Chaoda Zheng, Xu Yan, Jiantao Gao, Weibing Zhao, Wei Zhang, Zhen Li, and Shuguang Cui, "Box-aware feature enhancement for single object tracking on point clouds," in *ICCV*, 2021, pp. 13199–13208.

[31] Jiayao Shan, Sifan Zhou, Zheng Fang, and Yubo Cui, "Ptt: Point-track-transformer module for 3d single object tracking in point clouds," *arXiv:2108.06455*, 2021.

[32] Shan Jiayao, Sifan Zhou, Yubo Cui, and Zheng Fang, "Real-time 3d single object tracking with transformer," *IEEE Trans Multimedia*, 2022.

[33] Jesus Zarzar, Silvio Giancola, and Bernard Ghanem, "Efficient bird eye view proposals for 3d siamese tracking," *arXiv:1903.10168*, 2020.

[34] Florian Kraus and Klaus Dietmayer, "Uncertainty estimation in one-stage object detection," in *ITSC*, 2019, pp. 53–60.

[35] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[36] Dawei Zhang, Yanwei Fu, and Zhonglong Zheng, "UAST: Uncertainty-aware siamese tracking," in *ICML*, 2022, vol. 162 of *PMLR*, pp. 26161–26175.

[37] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," in *CVPR*, 2019.

[38] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*, 2012, pp. 3354–3361.

[39] Matej Kristan, Jiri Matas, Ales Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Cehovin, "A novel performance evaluation methodology for single-target trackers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2137–2155, 2016.

7.4 Uncertainty-Aware AB3DMOT by Variational 3D Object Detection

The appended paper [**?**] follows.

Uncertainty-Aware AB3DMOT by Variational 3D Object Detection

Illia Oleksiienko and Alexandros Iosifidis
DIGIT, Department of Electrical and Computer Engineering, Aarhus University, Denmark
{io,ai}@ece.au.dk

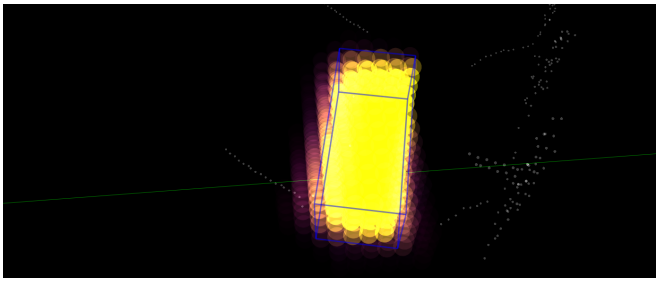
***Abstract*—Autonomous driving needs to rely on high-quality 3D object detection to ensure safe navigation in the world. Uncertainty estimation is an effective tool to provide statistically accurate predictions, while the associated detection uncertainty can be used to implement a more safe navigation protocol or include the user in the loop. In this paper, we propose a Variational Neural Network-based TANet 3D object detector to generate 3D object detections with uncertainty and introduce these detections to an uncertainty-aware AB3DMOT tracker. This is done by applying a linear transformation to the estimated uncertainty matrix, which is subsequently used as a measurement noise for the adopted Kalman filter. We implement two ways to estimate output uncertainty, i.e., internally, by computing the variance of the CNNs outputs and then propagating the uncertainty through the post-processing, and externally, by associating the final predictions of different samples and computing the covariance of each predicted box. In experiments, we show that the external uncertainty estimation leads to better results, outperforming both internal uncertainty estimation and classical tracking approaches. Furthermore, we propose a method to initialize the Variational 3D object detector with a pretrained TANet model, which leads to the best performing models.**

***Index Terms*—3D Object Detection, 3D Object Tracking, Point Cloud, Uncertainty Estimation, Bayesian Neural Networks, Variational Neural Networks**

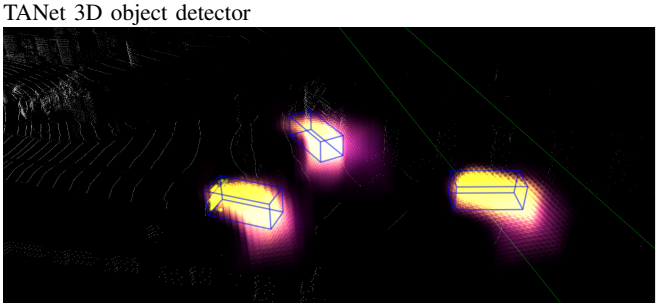
I. Introduction

3D Object Detection (3D OD) is the problem that aims to detect objects in the 3D world, providing the coordinates relative to the sensor and true world sizes of the detected objects. In contrast to 2D Object Detection which is commonly based on the appearance of objects in an image or video frame, in 3D OD the size of objects is not distorted by projections and rigid objects retain their size on every data frame. However, to obtain the data that can accurately represent objects in 3D requires specialized sensors. Even though color images [1]–[5] or stereo images [6]–[8] can be utilized to obtain 2D-like object detections and use those to estimate the corresponding 3D bounding boxes, the detection accuracy and speed of such methods is inferior to that of methods, such as VoxelNet [9], SECOND [10], PointPillars [11] and TANet [12], utilizing point cloud data captured by Lidar sensors.

3D Multiple Object (3D MOT) tracking aims to not only find the position of objects in the 3D world, but also assign a unique ID to each of them for the entire duration that an object appears in the data sequence. 3D MOT can be performed in a single stage, where the object detection in each frame and objects associations in successive frames are performed by the same model [13]–[15], or in two stages, where the detections are generated by a 3D OD model and the object associations in successive frames are performed separately.



(a) 3D bounding box with the uncertainty provided by the Variational TANet 3D object detector



(b) 3D bounding boxes with uncertainties provided to the Kalman filter of the uncertainty-aware AB3DMOT tracker

Fig. 1: Examples of a 3D bounding boxes with raw uncertainties obtained by the proposed Variational TANet (top figure), and the uncertainties provided to the Kalman filter of the proposed Uncertainty-aware AB3DMOT (bottom figure). Yellow color represents higher probability, while the purple color represents lower probability. Blue boxes correspond to the mean predictions.

Most 3D OD methods exploit object detections provided by powerful object detectors, commonly based on deep neural networks providing point estimates of their outputs. While recent advances in deep learning have led to remarkable results, this approach strongly restricts the ability of the object association step to exploit possible uncertainties in the object detections. Moreover, there has been some evidence that the predictive uncertainty of accurate deep learning models does not necessarily correlate with their confidence in their outputs [16], [17]. Uncertainty estimation in neural networks allows using the uncertainty in predictions to perform better decision-

making, which is important in critical fields, such as medical image analysis or autonomous driving. The practical application of uncertainty estimation includes 3D Object Tracking [18], [19], 3D Object Detection [20]–[22], 3D Human Pose Tracking [23], Steering Angle Prediction [24], providing better prediction and control than without the use of uncertainty.

In this paper, we propose an uncertainty-aware 3D object tracking pipeline. We show how the TANet 3D object detector [12] can be formulated and trained as a Variational Neural Network [25] to generate object detections with uncertainty in two ways, i.e., internally with variance estimation, and externally with covariance estimation. We combine them with a two-stage 3D MOT method called AB3DMOT [26] that utilizes a 3D Kalman filter [27] for state prediction and Hungarian algorithm [28] for objects' association. We modify AB3DMOT to use the predicted output uncertainty of the 3D OD model in the 3D Kalman Filter and show that this improves the Multi Object Tracking Accuracy (MOTA), F1 score and Mostly Lost (ML) metrics.

The remainder of the paper is structured as follows. Section II describes related and prior work. In Section III we describe the proposed approach, including the Variational TANet model and its training and the proposed uncertainty-aware AB3DMOt. Section IV outlines the experimental protocol and provides experimental results. Section V concludes this paper[1].

II. Related Work

3D OD based on Lidar point clouds cannot be performed with standard CNNs, as the point cloud data consists of a set of irregular 3D point positions. In order to introduced point cloud data as inputs to Deep Learning methods, one needs to structure it first, with voxelization being the most popular way of doing so. Voxelization is performed by selecting a region of interest in the 3D space and splitting it using a grid of same-sized cuboid shapes, called voxels. VoxelNet [18] creates a 3D grid of voxels and processes points inside each voxel to create voxel features, which are later processed by 3D and 2D convolutional layers. To accelerate the process, PointPillars [11] uses voxels with the maximum vertical size called pillars resulting in a 2D grid that can be used as input to a 2D CNN model, which is much faster than using 3D convolutions. TANet [12] introduces the Triple Attention module for extracting better features from pillars and a Coarse-to-Fine regression module that produces coarse detections first and then refines them with an additional subnetwork.

The fastest 2D Multiple Object Tracking (MOT) method called SORT [29] utilizes 2D Kalman Filter [27] and Hungarian algorithm [28] to associate the 2D object detections provided by a Deep Learning model. The 3D version of SORT, called AB3DMOT [26], performs better than SORT in 3D MOT by utilizing a 3D Kalman Filter instead of a 2D one. In [18], the use of uncertainty for 3D SOT is investigated by training the voxel-based 3D OD method SECOND [10] with an additional branch to predict the model's uncertainty, Then, the predicted uncertainty is used in the Kalman filter's measurement noise leading to better tracking results. The use of predicted uncertainty, instead of assuming a unit-Gaussian distribution over the measurements expressed by using the identity matrix, for measurement noise is also supported by [30].

Uncertainty estimation in neural networks can be performed using different ways to model uncertainty. The four main categories of uncertainty estimation methods defined in [31] are: Deterministic Methods [18], [32] which use a single network pass with uncertainty estimated from one of the network branches or by analyzing the behavior of the network; Bayesian Neural Networks (BNNs) [33], [34] that sample different models from the corresponding weight distribution and process the input using them resulting in a set of predictions, mean and variance of which are used to estimating the predictive distribution; Ensemble Methods [35], [36], that can also be seen as BNNs with Categorical distribution; and Test-Time Data Augmentation methods [37]–[39] that apply different augmentations to the input to create a set of slightly different inputs that are processed by the same model.

While Deterministic Methods require a single pass of the network leading to lower inference time, they usually perform worse than the rest of the methods as they can be seen as a point estimation of statistically better Bayesian approaches. Variational Neural Networks [25], [40] are similar to BNNs, but consider a Gaussian distribution over the outputs of each layer, the mean and variance of which are generated by the corresponding sub-layers. The output uncertainty is calculated by sampling different values from the corresponding Gaussian distributions, resulting in multiple predictions for the same input.

III. Uncertainty-aware 3D Object Tracking

A. Variational TANet

TANet [12] is a pillar-based 3D Object Detection model that improves PointPillars by introducing the Triple Attention module to extract better pillar (voxel) features and by using the Coarse-to-Fine fully convolutional network to process the voxel pseudo image and generate the bounding box predictions. TANet is an anchor-based method [41], which means that the output from the Coarse-to-Fine network is a voxel-wise prediction with position and sizes predicted as offsets from the corresponding anchor. This output is then processed by a Non-maximum-suppression module to select the most likely boxes and decoded based on the anchors to create the final 3D bounding boxes.

We train a Variational TANet (VTANet) by replacing the fully convolutional Coarse-to-Fine module with a Variational Neural Network (VNN) [25], [40] based version of it. The training is performed using the same loss function and training procedure, except we use multiple model samples per each data input and train for the mean of the predictions of all samples.

We propose two ways for the VTANet model to provide its estimated uncertainty to the subsequent tracker. The

[1]Our code is available at gitlab.au.dk/maleci/opendr/ua-ab3dmot

first, namely internal, way computes mean and variance of predictions of the Variational Coarse-to-Fine network and provides the variance through the decoding stage by treating the values as Gaussian random variables. The outputs of the CNN are voxel-wise in the form of (x, y, z, w, l, h, r) with respect to an anchor of the same data format, where (x, y, z) is the relative position vector, (w, l, h) is the relative size vector and r is the relative rotation angle. For a set of predictions from $(x_i, y_i, z_i, w_i, l_i, h_i, r_i), i \in [1, S]$, where S is the number of samples, we compute the mean prediction $(m_x, m_y, m_z, m_w, m_l, m_h, m_r)$ and the corresponding variance $(v_x, v_y, v_z, v_w, v_l, v_h, v_r)$. The anchor $(x_a, y_a, z_a, w_a, l_a, h_a, r_a)$ that corresponds to the prediction is used to create the decoded predictions $(\hat{x}, \hat{y}, \hat{z}, \hat{w}, \hat{l}, \hat{h}, \hat{r})$ as follows:

$$
\begin{aligned}
&d = \sqrt{l_a^2 + w_a^2}, \\
&(\hat{x}, \hat{y}) = (m_x, m_y)d + (x_a, y_a), \\
&(\hat{w}, \hat{l}, \hat{h}) = (e^{m_w} w_a, e^{m_l} l_a, e^{m_h} h_a), \\
&\hat{r} = m_r + r_a, \\
&\hat{z} = m_z h_a + z_a + h_a/2 - \hat{h}/2.
\end{aligned} \tag{1}
$$

The corresponding variances $(v_{\hat{x}}, v_{\hat{y}}, v_{\hat{z}}, v_{\hat{w}}, v_{\hat{l}}, v_{\hat{h}}, v_{\hat{r}})$ are calculated as follows:

$$
\begin{aligned}
&d = \sqrt{l_a^2 + w_a^2}, \\
&\mathrm{v}_{\exp}(m, v) = e^{2m+2v} - e^{2m+v}, \\
&(v_{\hat{x}}, v_{\hat{y}}) = (v_x, v_y)d^2, \\
&(v_{\hat{w}}, v_{\hat{l}}, v_{\hat{h}}) = \\
&\quad (\mathrm{v}_{\exp}(\hat{w}, v_w)w_a^2, \mathrm{v}_{\exp}(\hat{l}, v_l)l_a^2, \mathrm{v}_{\exp}(\hat{h}, v_h)h_a^2) \\
&\hat{r} = v_r, \\
&v_{\hat{z}} = v_z h_a^2,
\end{aligned} \tag{2}
$$

where $\mathrm{v}_{\exp}(m, v)$ is the variance of the exponent of the Gaussian random variable.

The second, namely external, way to compute uncertainty runs the full pipeline of TANet to create S 3D bounding box predictions, where S is the number of neural network parameter samples. These predictions are then grouped by finding the best association for each predicted object, similarly to the Hungarian algorithm on the tracking step. Consider the set of predictions $P^S = \{p_i \mid i \in [1, \ldots, S]\}$ where each element p_i is a set of 3D bounding boxes $p_i = \{b_k \mid k \in [1, \ldots, K_i]\}$, where K_i is the number of predicted boxes for the sample i. For different neural network parameter samples, some boxes with high uncertainty may or may not appear, which results in slightly different values of K_i. We define the association set $A(P)$ as a set of box groups $\{g_q\}$ where each group consists of 3D bounding boxes from different samples with the closest distances to each other, but no more than 1 meter. The groups are created by iterating through all boxes for each sample and checking the smallest distance to the average position of each existing group. If the distance is lower than 1 meter, the object is added to the selected group, and otherwise it creates a new group:

$$
\begin{aligned}
&\forall i \in [1..S] \, \forall k \in [1..K_i] \\
&\hat{G}(b_k) = \underset{g_q}{\operatorname{argmin}} \; |\mathrm{avg}(\{b^{\mathrm{x,y,z}} \in g_q\}) - b_k^{\mathrm{x,y,z}}| \\
&G(b_k) = \begin{cases} \hat{G}(b_k), & \text{if } |\mathrm{avg}(\{b^{\mathrm{x,y,z}} \in \hat{G}(b_k)\}) - b_k^{\mathrm{x,y,z}}| \leq 1, \\ g_{\text{new}}, & \text{otherwise,} \end{cases} \\
&g_q = \{b_k \mid G(b_k) = q, \; i \in [1..S], \; k \in [1..K_i]\},
\end{aligned} \tag{3}
$$

where $b^{x,y,z}$ is a position of the bounding box b and $\mathrm{avg}(\cdot)$ is the averaging function. For each group g_q, we compute the mean bounding box and the covariance matrix in the form of a 7×7 matrix for position (x, y, z), size (w, h, l) and rotation α.

B. Initialization of Variational TANet by pretrained model

Instead of training the VTANet model from scratch, we can initialize it based on an already trained TANet model which provides point estimates. This can be done based on the fact that infinitely small variance in a Gaussian distribution transforms it into a Dirac delta distribution, with all distributional mass placed on the mean value. By following the reverse process, we initialize the VTANet model with means from the corresponding pretrained TANet parameters, and the variance weights are set to be small values by either filling them with constant values or by using Xavier normal or uniform initialization [42]. Small variance values will not deviate too much from the initial pretrained values, but allow for training the whole model together and to find the optimal variance and mean parameters.

C. Uncertainty-Aware AB3DMOT

As shown in [18], [30], the Kalman filter benefits from providing actual uncertainties instead of assuming a unit-Gaussian distribution over the measurements. We follow this approach and provide the modified variance diagonal matrices or covariance matrices from the predicted uncertainties to the Kalman filter. We modify the uncertainty matrices provided by the detector with a linear transformation:

$$
\hat{\Sigma} = \alpha I + \beta \Sigma, \tag{4}
$$

where $\hat{\Sigma}$ is the uncertainty provided to the Kalman filter, Σ is the uncertainty predicted by the VTANet object detector, α is a hyperparameter used to control the contribution of the base uncertainty to $\hat{\Sigma}$, and β is a hyperparameter that controls the contribution of the predicted uncertainty to $\hat{\Sigma}$. This transformation aims to provide a degree of freedom for the Kalman filter in using the predicted object detections while utilizing the actual uncertainties in the predictions. Using the values $\alpha = 1$ and $\beta = 0$ leads to the standard AB3DMOT exploiting unit-Gaussian distribution measurement noise in the Kalman filter. Fig. 1 shows the difference between the uncertainty from the detector and the modified uncertainty provided to the Kalman filter.

TABLE I: Tracking results on KITTI tracking dataset.

Model	**Uncertainty method**	**Training Samples**	**Inference Samples**	**Tracking Parameters**	**MOTA%↑**	**F1%↑**	**ML%↓**
Voxel von-Mises [18]	deterministic	-	-	SORT [29]	-	55.10	30.60
TANet [12]	-	-	-	AB3DMOT [26]	68.71	85.69	8.58
IVTANet + UA-AB3DMOT	covar	2	4	$\alpha = 0.6,\ \beta = 5$	**72.30**	**87.34**	**7.74**
IVTANet + UA-AB3DMOT	covar	2	4	$\alpha = 0,\ \beta = 1$	72.13	87.26	7.74
IVTANet + UA-AB3DMOT	covar	2	4	$\alpha = 1,\ \beta = 0$	72.05	87.22	7.74
VTANet + UA-AB3DMOT	covar	3	3	$\alpha = 0.6,\ \beta = 5$	69.63	86.46	7.95
VTANet + UA-AB3DMOT	covar	3	3	$\alpha = 0,\ \beta = 1$	69.48	86.39	8.16
VTANet + UA-AB3DMOT	covar	3	4	$\alpha = 0.6,\ \beta = 5$	69.42	86.38	9.00
VTANet + UA-AB3DMOT	covar	3	3	$\alpha = 1,\ \beta = 0$	69.15	86.31	9.00
VTANet + UA-AB3DMOT	covar	3	4	$\alpha = 1,\ \beta = 0$	68.86	85.04	9.21
VTANet + UA-AB3DMOT	covar	2	4	$\alpha = 0.6,\ \beta = 5$	68.54	86.13	8.16
VTANet + UA-AB3DMOT	var	3	4	$\alpha = 0.5,\ \beta = 5$	68.46	85.91	7.95

IV. Experiments

We train TANet and VTANet models using the standard training procedure on KITTI [43] dataset described in [12]. VTANet models are trained with sample count in $[1, \ldots, 4]$ range and each model is evaluated with AB3DMOT on KITTI tracking dataset with every number of samples from the same range. We train the internal uncertainty VTANet models that provide the variance values and the external uncertainty models that provide the covariance values. Each model is evaluated with the different configuration of uncertainty transformation parameters α and β, including $\alpha = 0$, $\beta = 1$ for the predicted uncertainty only, $\alpha = 1$, $\beta = 0$ to not use the predicted uncertainty and different combinations of $\alpha \in [0, 1]$ and $\beta \in \{0.1, 1, 5, 10, 50\}$. Additionally, we train IVTANet models which are initialized with a pretrained TANet values for means and small variance weights and trained using the same training procedure as VTANet models. We compare the performance of the proposed method with that of the Voxel von-Mises method [18], which is the only method we found in the literature incorporating uncertainty estimation in 3D object tracking. This method employs SECOND [10] for 3D OD and SORT [29] for objects associations.

Table I shows the performance of the competing models on the KITTI tracking dataset based on three performance metrics, namely the Multi Object Tracking Accuracy (MOTA), F1 score, and Mostly Lost (ML). This table also provides information on the hyperparameter values used for different model, i.e., the type of adopted uncertainty estimation, the number of network parameter samples used during training and inference, and the tracking method and hyperparameter values.

As can be seen, the proposed uncertainty-aware 3D MOT method with a VTANet trained from scratch, i.e., VTANet + UA-AB3DMOT, provides better MOTA, F1 and ML scores compared to the classic 3D MOT approach combining TANet with AB3DMOT, i.e., TANet + AB3DMOT, due to the use of uncertainty. The external way to compute uncertainty as covariance provides better results compared to the internal one providing variance across all configurations, and therefore, only a sub-set of configurations is provided in the results. The Voxel von-Mises method [18] provides much worse results due to the use of inferior methods for each of its processing steps, i.e., it adopts the less accurate 3D detector SECOND [10], the less suited for 3D object detection tracker SORT [29] and a deterministic uncertainty estimation method that does not rely on a BNN or an Ensemble method during training. The best MOTA, F1 and ML scores are achieved by the configurations of the proposed method which employ a VTANet pretrained using a TANet providing point estimates. Those are denoted by IVTANet + UA-AB3DMOT in Table I. As can be seen in the results, the use of an IVTANet 3D object detector improves performance compared to the classic 3D MOT approach even without exploiting the estimated uncertainty (i.e., when $\alpha = 1$ and $\beta = 0$). Overall, the highest performance is achieved by using a VTANet obtained by training a pretrained TANet combined with the AB3DMOT model exploiting the estimated uncertainty (IVTANet + UA-AB3DMOT). It is interesting to see that the use of the same linear combination values ($\alpha = 0.6$ and $\beta = 5$) leads to the best performance when both VTANet and IVTANet 3D object detectors are used.

V. Conclusions

In this paper, we proposed an uncertainty-aware 3D object tracking pipeline that utilizes a Variational Neural Network-based version of TANet 3D object detector to generate predictions with uncertainty in two different ways, i.e., internally with variance estimation and externally with covariance estimation. Predictions from the detector are introduced into the uncertainty-aware tracker that utilizes the estimated uncertainty to determine the measurement noise in the Kalman filter, leading to an improvement in MOTA, F1 and ML metrics. We use a pretrained TANet model to initialize mean weights of a Variational TANet and initialize the variance weights with small values. The resulting model is trained using the regular training procedure and leads to a much better tracking performance compare to the models trained from scratch.

Acknowledgement

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun, "Monocular 3d object detection for autonomous driving," in *CVPR*, 2016, pp. 2147–2156.

[2] Tong He and Stefano Soatto, "Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors," in *AAAI*, 2019, pp. 8409–8416.

[3] Bin Xu and Zhenzhong Chen, "Multi-level fusion based 3d object detection from monocular images," in *CVPR*, 2018, pp. 2345–2353.

[4] Zengyi Qin, Jinglu Wang, and Yan Lu, "Monogrnet: A geometric reasoning network for monocular 3d object localization," in *AAAI*, 2019, pp. 8851–8858.

[5] Thomas Roddick, Alex Kendall, and Roberto Cipolla, "Orthographic feature transform for monocular 3d object detection," in *BMVC*, 2019, p. 285.

[6] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark E. Campbell, and Kilian Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *CVPR*, 2019, pp. 8445–8453.

[7] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger, "Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving," in *ICLR*, 2020.

[8] Chengyao Li, Jason Ku, and Steven L. Waslander, "Confidence guided stereo 3d object detection with split depth estimation," *IROS*, pp. 5776–5783, 2020.

[9] Yin Zhou and Oncel Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *CVPR*, 2018, pp. 4490–4499.

[10] Yan Yan, Yuxing Mao, and Bo Li, "SECOND: sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, pp. 3337, 2018.

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," in *CVPR*, 2019.

[12] Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai, "Tanet: Robust 3d object detection from point clouds with triple attention," in *AAAI*, 2020, pp. 11677–11684.

[13] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl, "Center-based 3d object detection and tracking," *CoRR*, vol. abs/2006.11275, 2020.

[14] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krähenbühl, Trevor Darrell, and Fisher Yu, "Joint monocular 3d vehicle detection and tracking," in *CVF*, 2019, pp. 5389–5398.

[15] Martin Simon, Karl Amende, Andrea Kraus, Jens Honer, Timo Sämann, Hauke Kaulbersch, Stefan Milz, and Horst-Michael Gross, "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds," in *CVPR*, 2019, pp. 1190–1199.

[16] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger, "On calibration of modern neural networks," in *ICML*, 2017, pp. 1321–1330.

[17] Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Ann Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic, "Revisiting the Calibration of Modern Neural Networks," in *NeurIPS*, 2021.

[18] Yuanxin Zhong, Minghan Zhu, and Huei Peng, "Uncertainty-aware voxel based 3d object detection and tracking with von-mises loss," *arXiv:2011.02553*, 2020.

[19] Jianren Wang, Siddharth Ancha, Yi-Ting Chen, and David Held, "Uncertainty-aware self-supervised 3d data association," in *RSJ*, 2020, pp. 8125–8132.

[20] Di Feng, Lars Rosenbaum, and Klaus Dietmayer, "Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection," in *ITSC*, 2018, pp. 3266–3273.

[21] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," in *CVPR*, 2019, pp. 12677–12686.

[22] Gregory P. Meyer and Niranjan Thakurdesai, "Learning an uncertainty-aware object detector for autonomous driving," in *IROS*, 2020, pp. 10521–10527.

[23] Ben Daubney and Xianghua Xie, "Tracking 3d human pose with large root node uncertainty," in *CVPR*, 2011, pp. 1321–1328.

[24] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza, "A general framework for uncertainty estimation in deep learning," *RA-L*, vol. 5, no. 2, pp. 3153–3160, 2020.

[25] Illia Oleksiienko, Dat Thanh Tran, and Alexandros Iosifidis, "Variational neural networks," *arxiv:2207.01524*, 2022.

[26] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," *IROS*, 2020.

[27] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.

[28] H. W. Kuhn, "The hungarian method for the assignment problem," *Nav. Res. Logist.*, vol. 2, no. 1-2, pp. 83–97, 1955.

[29] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Tozeto Ramos, and Ben Upcroft, "Simple online and realtime tracking," in *IEEE International Conference on Image Processing*, 2016, pp. 3464–3468.

[30] Rebecca L. Russell and Christopher Reale, "Multivariate uncertainty in deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7937–7943, 2022.

[31] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna M. Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu, "A survey of uncertainty in deep neural networks," *arxiv:2107.03342*, 2021.

[32] Murat Sensoy, Lance Kaplan, and Melih Kandemir, "Evidential deep learning to quantify classification uncertainty," in *NeurIPS*, 2018, p. 3183–3193.

[33] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra, "Weight Uncertainty in Neural Networks," in *JMLR*, 2015, vol. 37, pp. 1613–1622.

[34] Martin Magris and Alexandros Iosifidis, "Bayesian learning for neural networks: an algorithmic survey," *Artificial Intelligence Review*, 2023.

[35] Ian Osband, John Aslanides, and Albin Cassirer, "Randomized prior functions for deep reinforcement learning," in *NeurIPS*, 2018, vol. 31, pp. 8626–8638.

[36] Matias Valdenegro-Toro, "Deep sub-ensembles for fast uncertainty estimation in image classification," *arxiv:1910.08168*, 2019.

[37] Guotai Wang, Wenqi Li, Sébastien Ourselin, and Tom Vercauteren, "Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation," in *BrainLes*. 2018, vol. 11384, pp. 61–72, Springer.

[38] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren, "Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks," *Neurocomputing*, vol. 338, pp. 34–45, 2019.

[39] Ibrahem Kandel and Mauro Castelli, "Improving convolutional neural networks performance for image classification using test time augmentation: a case study using MURA dataset," *Health Inf. Sci. Syst.*, vol. 9, no. 1, pp. 33, 2021.

[40] Illia Oleksiienko, Dat Thanh Tran, and Alexandros Iosifidis, "Variational neural networks implementation in pytorch and jax," *Software Impacts*, vol. 14, pp. 100431, 2022.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, ," in *Advances in Neural Information Processing Systems*, 2015, vol. 28.

[42] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *AISTATS*, 2010, vol. 9, pp. 249–256.

[43] Andreas Geiger, Philip Lenz, and Raquel Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*, 2012, pp. 3354–3361.

7.5 CoVIO: Online Continual Learning for Visual-Inertial Odometry

The appended paper [98] follows.

CoVIO: Online Continual Learning for Visual-Inertial Odometry

Niclas Vödisch[1] Daniele Cattaneo[1] Wolfram Burgard[2] Abhinav Valada[1]
[1]University of Freiburg [2]University of Technology Nuremberg

Abstract

Visual odometry is a fundamental task for many applications on mobile devices and robotic platforms. Since such applications are oftentimes not limited to predefined target domains and learning-based vision systems are known to generalize poorly to unseen environments, methods for continual adaptation during inference time are of significant interest. In this work, we introduce CoVIO for online continual learning of visual-inertial odometry. CoVIO effectively adapts to new domains while mitigating catastrophic forgetting by exploiting experience replay. In particular, we propose a novel sampling strategy to maximize image diversity in a fixed-size replay buffer that targets the limited storage capacity of embedded devices. We further provide an asynchronous version that decouples the odometry estimation from the network weight update step enabling continuous inference in real time. We extensively evaluate CoVIO on various real-world datasets demonstrating that it successfully adapts to new domains while outperforming previous methods. The code of our work is publicly available at http://continual-slam.cs.uni-freiburg.de.

1. Introduction

Reliable estimation of a robot's motion based on its onboard sensors is a fundamental requirement for many downstream tasks including localization and navigation. Devices such as inertial measurement units (IMU) or inertial navigation systems (INS) provide a way to directly measure the robot's motion based on acceleration and GNSS readings. An alternative is to use visual odometry (VO) leveraging image data from monocular or stereo cameras. Such VO methods have been successfully used in UAVs [4], mobile applications [24], and even mars rovers [21]. Similar to other vision tasks, learning-based VO has gained increasing attention as the learnable high-level features can circumvent problems in textureless regions [27, 28] or in the presence of dynamic objects [1] where classical handcrafted methods suffer. However, learning-based VO lacks the ability to generalize to unseen domains, hindering their open-world deployment. Recently, adaptive VO [18] has opened a new avenue of research, *e.g.*, by using continual learning (CL) methodologies to enhance VO during inference time [30].

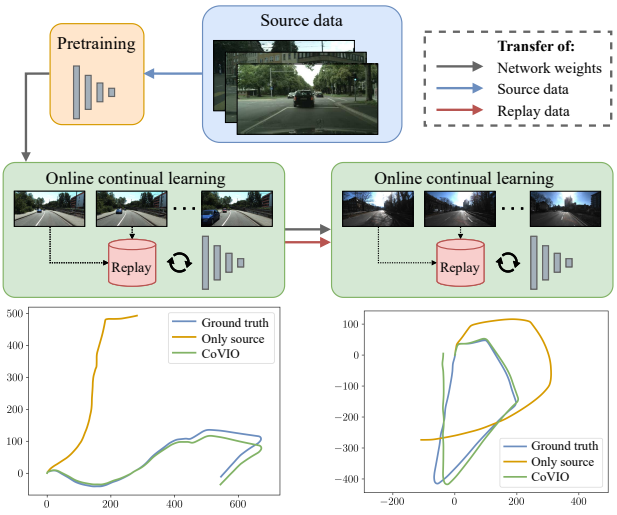


Figure 1. We propose CoVIO for online continual learning of visual-inertial odometry. After pretraining on a source domain that is then discarded, CoVIO further updates the network weights during inference on a target domain. Using experience replay CoVIO successfully mitigates catastrophic forgetting.

Most commonly, learning-based VO leverages monocular depth estimation as an auxiliary task [13, 18, 30] and exploits an unsupervised joint training scheme of a PoseNet, estimating the camera motion between two frames, and a DepthNet, estimating depth from a single image [6]. Due to the unsupervised nature of this approach, learning-based VO can be continuously trained also during inference time. In addition to classical domain adaptation [2], where knowledge is transferred from a single source to a single target domain, the recent study on continual SLAM [30] also investigates a sequential multi-domain setting as illustrated in Fig. 1. The authors introduce CL-SLAM, which fuses adaptive VO with a graph-based SLAM backend. To avoid catastrophic forgetting, *i.e.*, overfitting to the current domain while losing the ability to perform well on past domains, CL-SLAM employs a dual-network architecture comprising an expert and a generalizer for both efficient domain

adaptation and knowledge retention combined with experience replay. However, the previously proposed CL-SLAM suffers from three main drawbacks: First, network weights are transferred from the generalizer to the expert upon the start of a new evaluation sequence, *i.e.*, a human supervisor decides when new data should be classified as a domain change. Second, the utilized replay buffer of the generalizer is of infinite size and, thus, does not consider the limited storage capacity of real-world applications. Finally, since every received frame triggers an update of the network weights before yielding the VO estimate, real-time usage is difficult to achieve on low-power devices such as embedded hardware in robots.

In this work, we propose a novel adaptive visual-inertial odometry estimation method called CoVIO that explicitly addresses all of the aforementioned drawbacks of CL-SLAM. Similar to Kuznietsov *et al*. [11], we consider a source-free setting, *i.e.*, experience replay does not include data from the source domain used for pretraining. In particular, the contributions of this work can be summarized as follows:

1. We replace the dual-network architecture with a single network addressing both domain adaptation and knowledge retention but simplifying the overall architecture and reducing the GPU memory footprint. Additionally, this resolves the issue of transferring network weights without domain classification.
2. We propose a fixed-size replay buffer that maximizes image diversity and addresses the limited storage capacity of embedded devices.
3. We present an asynchronous version of CoVIO that separates the core motion estimation from the network update step allowing true continuous inference
4. We perform extensive evaluations of CoVIO on various datasets, both publicly available and in-house, demonstrating its efficacy compared to other visual odometry methods.
5. We release the code of our work and trained models at http://continual-slam.cs.uni-freiburg.de.

2. Related Work

In this section, we provide a brief introduction to continual and lifelong learning and summarize previous methods for domain adaptation of learning-based visual odometry.

Continual Learning: Deep learning-based models are commonly trained for a specific task, which is defined a priori, using a fixed set of training data. During inference, the model is then employed on previously unseen data from the same domain without further updates of the network weights. However, in many real-world scenarios, this assumption does not hold true, *e.g.*, the initially used training data might not well represent the data seen during inference, thus leading to a domain gap and suboptimal performance. Additionally, the objective of the task can change over time. Continual learning (CL) and lifelong learning [26] aim to overcome these challenges by enabling a method to continually learn additional tasks given new training data. In contrast to vanilla domain adaptation [2], CL methods should maintain the capability to solve previously learned tasks, *i.e.*, avoiding catastrophic forgetting. Ideally, learning a task also yields improved performance on previous tasks (*positive backward transfer*) as well as on yet unknown future tasks (*positive forward transfer*) [16]. The majority of CL approaches can be categorized into three strategies. First, experience replay directly tackles catastrophic forgetting from a data-driven perspective. For instance, both CoMoDA [10] and CL-SLAM [30] store images in a replay buffer and combine online data with replay samples when updating the network weights. Second, regularization techniques such as knowledge distillation [29] preserve information on a more abstract feature level. Finally, architectural methods prevent forgetting by using certain network structures, *e.g.*, LSTMs [13] and dual-network architectures [30], or by directly freezing internal model parameters. Online continual learning [17, 31] describes an extension of CL by considering a setting, where the model is continuously updated on a stream of data during inference time. Online CL also includes scenarios, which gradually change from one domain to another [25]. In this work, we employ online CL with experience replay for learning-based visual-inertial odometry estimation.

Adaptive Visual Odometry: Online adaptation of learning-based visual odometry (VO) and simultaneous localization and mapping (SLAM) aims to enhance performance on the fly allowing robotic systems to operate more reliably in previously unseen environments. Most commonly, learning-based VO relies on monocular depth estimation as an auxiliary task enabling joint training of a DepthNet and a PoseNet in an unsupervised manner [6]. In an early work on adaptive VO, Luo *et al*. [18] accumulate images from an online camera stream and leverage the unsupervised training scheme to update both networks. Different to experience replay in CL, the buffer of accumulated images is emptied after the update step, *i.e.*, each sample is only seen once. Li *et al*. [13] propose an architectural CL technique, replacing the standard convolutional layers with LSTM variants to prevent forgetting. During deployment, the networks are continuously trained using only the online data. In a follow-up work by the same authors [14], the PoseNet is substituted with optical flow-based point matching. Similarly, GeoRefine [9] combines online depth refinement with dense visual mapping. While the DepthNet is updated following the aforementioned works, GeoRefine uses a non-adaptive odometry and tracking module based on optical flow. Loo *et al*. [15] propose an adaptive visual SLAM system that com-

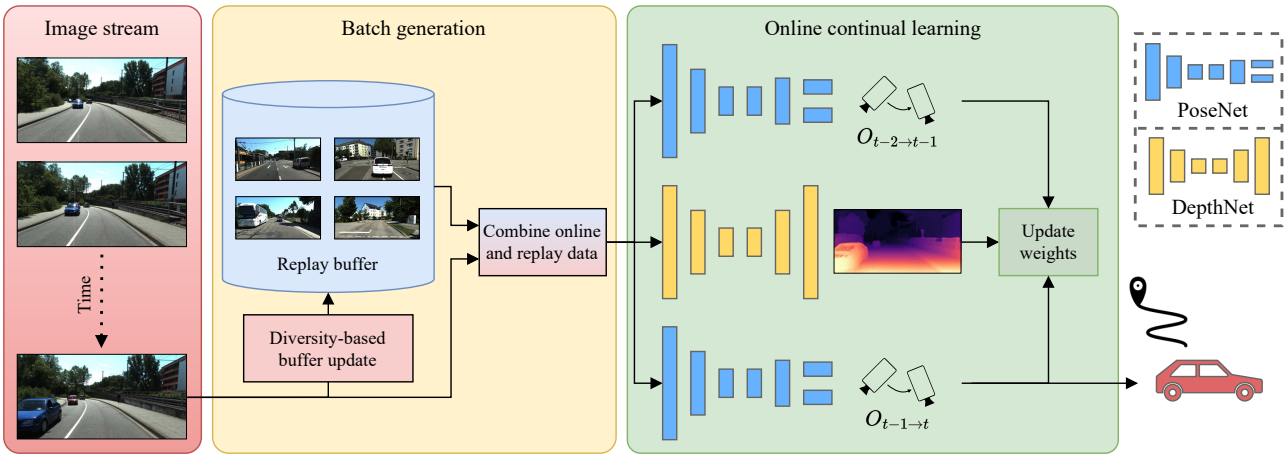


Figure 2. Our proposed CoVIO performs online continual learning on a stream of RGB images leveraging unsupervised monocular depth estimation as an auxiliary task. In each update step, the image triplet consisting of the current and the two previous frames is combined with samples from a replay buffer and then fed to the networks to update their weights via backpropagation. The estimated camera motion between the previous and the current image corresponds to the generated VO output. The replay buffer is optionally updated if the current frame is sufficiently different from the existing content.

bines experience replay with a variant of elastic weight consolidation (EWC) to further regularize the weight updates of both the DepthNet and the PoseNet. Finally, to avoid catastrophic forgetting in a multi-domain adaptation setting, CL-SLAM [30] exploits a dual-network architecture, which is composed of an expert to perform effective online adaptation to the new domain and a generalizer to retain previously acquired knowledge by leveraging experience replay. In this work, we propose an adaptive method for visual-inertial odometry built on CL-SLAM that explicitly addresses its shortcomings as outlined in Sec. 1.

3. Technical Approach

In the following sections, we first describe the network architecture along with the pretraining procedure on a source domain. Then, we introduce CoVIO and provide detailed explanations of all contributions.

3.1. Network Architecture and Pretraining

In this section, we detail the network architecture of our proposed CoVIO and the loss functions that we employ during the initial training phase.

Network Architecture: We build our network following the common scheme of unsupervised monocular depth estimation leveraging two separate networks that we refer to as DepthNet and PoseNet as depicted in Fig. 2. Similar to CL-SLAM [30], for an image triplet $\{\mathbf{I}_{t-2}, \mathbf{I}_{t-1}, \mathbf{I}_t\}$ we use Monodepth2 [6] to jointly predict a dense depth map $\mathbf{D}_{t-1}$ of the center image and the camera motion with respect to both neighboring frames, *i.e.*, $\mathbf{O}_{t-2\to t-1}$ and $\mathbf{O}_{t-1\to t}$. In CoVIO, we then output the latter as the VO estimate. In particular, we use an implementation comprising two separate ResNet-18 [8] encoders for the DepthNet and the PoseNet.

Source Domain Pretraining: To initialize CoVIO, we perform unsupervised training on a source domain $\mathcal{S}$ in an offline manner. In detail, we exploit the photometric reprojection loss $\mathcal{L}_{pr}$ and the image smoothness loss $\mathcal{L}_{sm}$ to train the DepthNet and the PoseNet [6]. We additionally supervise the PoseNet with scalar velocity readings from the vehicle's IMU [7]. The applied velocity supervision term $\mathcal{L}_{vel}$ enforces metric scale-aware odometry estimates. Thus, our total loss is composed of three terms:

$$\mathcal{L} = \mathcal{L}_{pr} + \gamma\mathcal{L}_{sm} + \lambda\mathcal{L}_{vel}, \tag{1}$$

with weighting factors γ and λ.

3.2. Online Continual Learning

After pretraining on a source domain $\mathcal{S}$, we use CoVIO to perform online continual learning on an unseen target domain $\mathcal{T}$. As illustrated in Fig. 2, each new RGB image triggers the following steps:

(1) Create a data triplet comprising the new frame $\mathbf{I}_t$ and the two previous frames $\mathbf{I}_{t-1}$ and $\mathbf{I}_{t-2}$ along with the corresponding IMU readings.
(2) Check whether this triplet should be added to the replay buffer using the proposed diversity-based update mechanism.
(3) Sample from the replay buffer and combine the samples with the previously generated data triplet.
(4) Estimate the depth map $\mathbf{D}_{t-1}$ and the camera motions $\mathbf{O}_{t-2\to t-1}$ and $\mathbf{O}_{t-1\to t}$.

(5) Compute the loss defined in Eq. (1) and update the network weights via backpropagation.
(6) Repeat steps (4) and (5) for c iterations.
(7) Output $\mathbf{O}_{t-1\to t}$ as the odometry estimate.

In the following, we provide more details on the proposed replay buffer and the online continual learning strategy of CoVIO. Finally, we propose an asynchronous version of CoVIO that separates the motion estimation from the network update step allowing continuous inference.

3.2.1 Replay Buffer

As outlined in Sec. 1, previous works [10, 30] typically assumed an infinitely sized replay buffer without considering the limited storage capacity on robotic platforms or mobile devices. To address this issue, we use a replay buffer with a fixed maximum size and propose an image diversity-based update mechanism that is comprised of two steps shown in Fig. 3. First, determine whether to add the current online data into the replay buffer and, second, if adding the data results in exceeding the predefined buffer size, select a sample that will be removed from the buffer.

Inspired by the loop closure detection in visual SLAM [12, 30], we interpret the cosine similarity between image feature maps as a distance measure between two frames $\mathbf{I}_1$ and $\mathbf{I}_2$:

$$\text{sim}_{\cos} = \cos\left(\text{feat}(\mathbf{I}_1), \text{feat}(\mathbf{I}_2)\right), \tag{2}$$

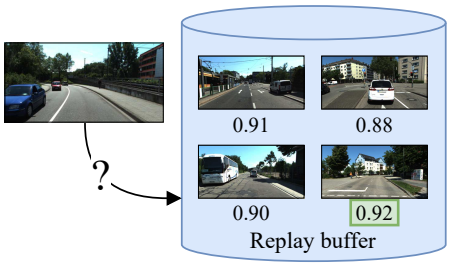
where $\text{feat}(\cdot)$ denotes the respective image features. In order to determine whether adding a new sample would increase the diversity of the replay buffer, we compute its cosine similarity with respect to all samples that are already in the buffer and take the maximum value.

$$\text{sim}_{\mathbf{B}}(\mathbf{I}_t) = \max_{\mathbf{I}_i \in \mathbf{B}} \cos\left(\text{feat}(\mathbf{I}_t), \text{feat}(\mathbf{I}_i)\right), \tag{3}$$

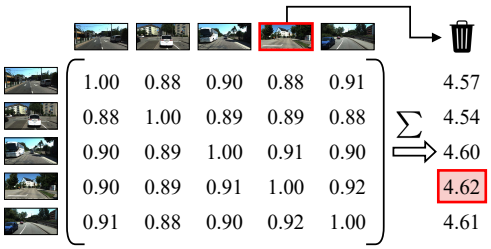
where $\mathbf{I}_i \in \mathbf{B}$ refers to the current content of the buffer. If $\text{sim}_{\mathbf{B}}(\mathbf{I}_t) < \theta_{th}$, the data triplet associated with $\mathbf{I}_t$ is added to the replay buffer. In case this results in a buffer size larger than the allowed size, we have to remove a sample from the buffer. Instead of using random sampling, we remove the sample that yields maximal diversity within the remaining samples. Formally, we remove the following sample:

$$\arg\max_{\mathbf{I}_i \in \mathbf{B}} \sum_{\mathbf{I}_j \in \mathbf{B}} \cos\left(\text{feat}(\mathbf{I}_i), \text{feat}(\mathbf{I}_j)\right) \tag{4}$$

As described in the next section, we do not update the encoder weights of CoVIO. Therefore, to avoid the overhead of a separate network, we use the encoder of the DepthNet to generate image features.



(a) An image is added to the replay buffer if the cosine similarity to the most similar image in the buffer is below a threshold, *e.g.*, $\theta_{th} = 0.95$. Here, the image will be added since $0.92 < \theta_{th}$.



(b) If adding a new image results in exceeding the allowed size of the replay buffer, the image that is the most similar with respect to all other images is removed. The table shows the cosine distance between two frames.

Figure 3. Diversity-based update mechanism of the replay buffer, separated in (a) adding and (b) removing a sample.

3.2.2 Adaptive Visual-Inertial Odometry

After the replay buffer has been updated, we construct a batch $\mathbf{b}_t$ consisting of the data triplet of the current image $\mathbf{I}_t$ and N samples from the replay buffer.

$$\mathbf{b}_t = \{\mathbf{I}_t, \mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_N\} \tag{5}$$

To query the samples from the buffer, we use a uniform probability distribution across all samples and avoid selecting the same sample multiple times if the current size of the buffer is greater than the requested number of samples. To further increase diversity, we augment the replay images in terms of brightness, contrast, saturation, and hue value. Next, the batch $\mathbf{b}_t$, comprising RGB images and velocity measurements, is fed to the DepthNet to estimate a dense depth map of the center images and to the PoseNet to estimate the camera motion with respect to both neighboring frames. Following the same procedure as during pretraining (see Sec. 3.1), we then compute the loss $\mathcal{L}$ defined in Eq. (1) and perform backpropagation to update the network weights. Following McCraith *et al.* [22], we do not update the weights of the encoders but only of the decoders.

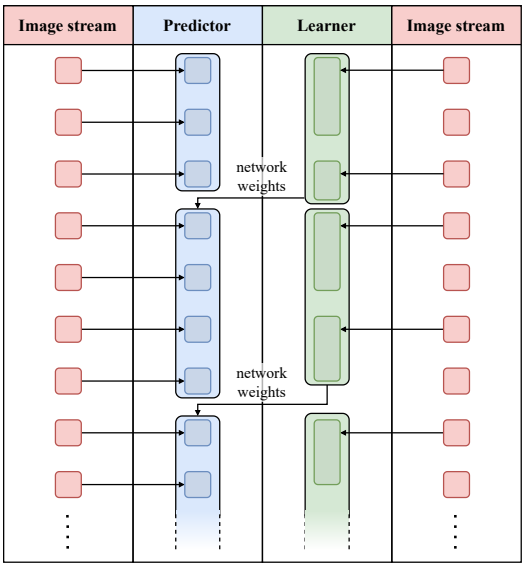


Figure 4. Illustration of the asynchronous variant of CoVIO. While the predictor generates visual odometry estimates in real time, the learner updates the network weights via backpropagation. After a given number of update cycles, the network weights are transferred from the learner to the predictor.

3.2.3 Asynchronous CoVIO

Finally, we propose an asynchronous variant of CoVIO to address true continuous inference on robotic platforms in a real-time capable setting. Since multiple update iterations c can result in a situation, in which the network update takes longer than the frame rate of the input camera stream, we also design a version that decouples the VO estimation from the CL updates. As illustrated in Fig. 4, the *predictor* continuously generates VO estimates for each incoming image. The *learner* contains a copy of the network that is updated using the previously introduced online CL strategy but disregards images if the update step takes longer than the time until the next frame is available. Compared to caching frames, this strategy ensures that always the latest information is used to update the network. Then, after a given number of update cycles, the network weights are transferred from the *learner* to the *predictor*. We include implementations in both ROS and ROS2 in our published code base.

4. Experimental Evaluation

In this section, we present extensive experimental results on the efficiency and efficacy of our proposed CoVIO, compared to previous works. We further conduct multiple ablation studies to demonstrate the effect of newly introduced hyperparameters and to justify certain design choices.

Throughout all experiments, we report the translation error t_{err} (in %) and the rotation error r_{err} (in °/m) as proposed by Geiger *et al*. [5]. These metrics evaluate the error as a function of the trajectory length. To ensure a fair comparison with the base work CL-SLAM [30], we further utilize the set of network weights that is provided by the authors and was pretrained on the Cityscapes Dataset [3]. We also follow CL-SLAM and only consider new frames when the IMU measures a driven distance of at least $0.2\,\mathrm{m}$.

4.1. Datasets

We employ our method on various datasets simulating a diverse set of environments. In particular, we initialize CoVIO with network weights trained on Cityscapes [3] and perform online continual learning on sequences from the KITTI odometry benchmark [5], the Oxford RobotCar Dataset [20], and in-house data.

Cityscapes: The Cityscapes Dataset [3] 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. In this work, we use network weights pretrained on Cityscapes that are provided by Vödisch *et al*. [30].

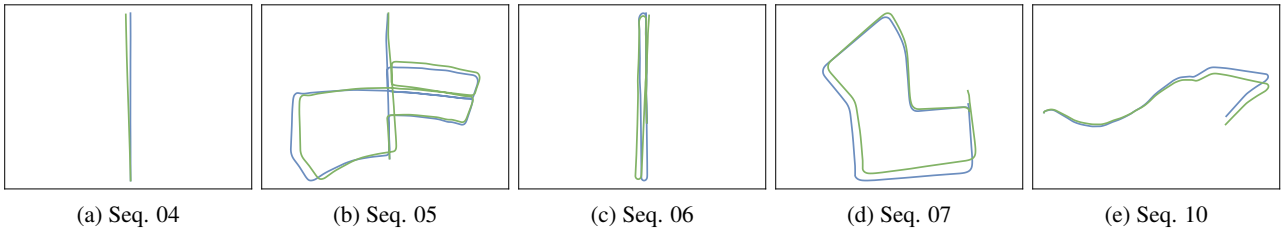
KITTI: The KITTI Dataset [5] is a pioneering autonomous driving dataset that was recorded in Karlsruhe, Germany. For continual learning of new domains, we use images and ground truth poses of multiple sequences from the odometry benchmark and combine them with the respective IMU data from the raw dataset.

Oxford RobotCar: The Oxford RobotCar Dataset [20] provides multiple recordings of the same route that were captured across one year. We use the included RGB images and the IMU data. To compute the error metrics, we exploit the separately released RTK ground truth positions [19].

In-House: Finally, we employ CoVIO on an in-house dataset recorded in Freiburg, Germany. Our robotic platform includes forward-facing RGB cameras and an inertial navigation system (INS), that we use to compute the velocity supervision loss.

4.2. Evaluation of Online Continual Learning

In this section, we conduct a series of experiments including both simple online domain adaptation from a source $\mathcal{S}$ to a target domain $\mathcal{T}$ and online continual learning from $\mathcal{S}$ to a sequence of target domains $\{\mathcal{T}_1, \mathcal{T}_2, \dots\}$. Based on the ablation studies in Sec. 4.3, we use a replay buffer size of $|\mathbf{B}| = 100$, an update batch size of $|\mathbf{b}_t| = 3$, and $c = 5$ backpropagation steps allowing a fair comparison with the base work CL-SLAM [30]. We set a similarity threshold of $\theta_{th} = 0.95$ for the diversity-based update scheme of the replay buffer. For the loss weights, we follow CL-SLAM and use $\gamma = 0.001$ and $\lambda = 0.05$. To compare with other methods, we do not use the asynchronous version but the same learning scheme as in CL-SLAM.



(a) Seq. 04 (b) Seq. 05 (c) Seq. 06 (d) Seq. 07 (e) Seq. 10

Figure 5. Online continual learning results on the KITTI odometry benchmark after pretraining on the Cityscapes dataset.

Table 1. Comparison of continual odometry estimation on the KITTI odometry benchmark.

Method	Seq. 04		Seq. 05		Seq. 06		Seq. 07		Seq. 10	
	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
ORB-SLAM [23]	0.62	0.11	2.51	0.25	7.80	0.35	1.53	0.35	2.96	0.52
Only target	10.72	1.69	34.55	11.88	15.20	5.62	12.77	6.80	55.27	9.50
DeepSLAM [12]	5.22	2.27	4.04	1.40	5.99	1.54	4.88	2.14	10.77	4.45
Only source	28.94	4.64	46.13	19.20	49.57	20.79	37.75	25.42	30.91	15.28
CL-SLAM [30]	4.37	**0.51**	4.30	1.01	2.53	0.63	**2.10**	**0.83**	11.18	1.74
CoVIO *(ours)*	**2.11**	0.53	**2.88**	**0.94**	**2.13**	**0.47**	3.19	1.26	**3.71**	**1.55**

Comparison of the translation and rotation errors of our CoVIO with baseline methods evaluated on the KITTI odometry benchmark. "Only target" and DeepSLAM are trained on sequences {00, 01, 02, 08, 09} without further adaptation. "Only source", CL-SLAM, and CoVIO are trained on Cityscapes. Both CL-SLAM and CoVIO perform online adaptation on the respective KITTI sequence. The values of CL-SLAM and "only target" are reported by Vödisch *et al.* [30]. The errors of the paths predicted by ORB-SLAM are based on ground truth scaling and hence not directly comparable to the other methods. The smallest and second smallest errors across the methods producing metric predictions are shown in **bold** and underlined.

4.2.1 Cityscapes to KITTI

We use our proposed CoVIO to perform online continual adaptation from Cityscapes to KITTI and compare its performance to other methods shown in Tab. 1. In detail, we compare with the traditional ORB-SLAM [23] as well as the following learning-based methods: "Only target" and DeepSLAM [12] are trained on the KITTI sequences {00, 01, 02, 08, 09} without further adaptation; "only source", CL-SLAM [30], and CoVIO are trained on Cityscapes with online adaptation to KITTI for both CL-SLAM and CoVIO. Generally, the difference between "only source" and "only target" demonstrates the domain gap that online adaptation aims to overcome. Our proposed CoVIO outperforms the base method CL-SLAM on the majority of sequences and also improves performance compared to offline training on the target domain. We visualize the predicted and ground truth odometry in Fig. 5. Note that, unlike CL-SLAM and DeepSLAM, we do not include loop closures in CoVIO.

We further perform online continual learning on all sequences in a sequential manner, *i.e.*, after pretraining on Cityscapes, adapt to sequence 04, then sequence 05, etc., and list the results in Tab. 2. In particular, we compute the translation and rotation errors after each step on all sequences to determine both forward and backward transfer, *i.e.*, the effect on previous and yet unseen future sequences. Since all the sequences of a dataset could be considered to represent similar domains, *e.g.*, the same camera parameters and comparable environments, we observe a general trend of positive forward transfer. Although the performance on previous sequences cannot be fully retained, CoVIO successfully mitigates catastrophic forgetting compared to the initial performance after pretraining on the source domain.

4.2.2 Cityscapes to In-House Data

Next, we utilize CoVIO to estimate visual odometry on an in-house dataset, after pretraining on Cityscapes. In Fig. 6, we provide a qualitative comparison of CoVIO to CL-SLAM [30] with disabled loop closure detection, no online adaptation, and the measured GNSS position. Since we do not have access to highly accurate RTK readings, we omit computing error metrics for this dataset. However, as demonstrated in Fig. 6, CoVIO is able to maintain accurate odometry tracking for a longer distance than CL-SLAM.

4.2.3 Cityscapes to KITTI and RobotCar

We further investigate the capability of CoVIO to retain knowledge in a multi-target setting. In detail, we perform the same experiment as conducted by CL-SLAM [30]. After initialization on Cityscapes, we sequentially deploy CoVIO to KITTI sequence 09, a sequence from RobotCar, KITTI sequence 10, and another sequence from RobotCar.

Table 2. Continual odometry estimation results on the KITTI odometry benchmark.

Sequence	Images	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}	t_{err}	r_{err}
		Cityscapes ⟶		Seq. 04 ⟶		Seq. 05 ⟶		Seq. 06 ⟶		Seq. 07 ⟶		Seq. 10	
Seq. 04	269	28.94	4.64	2.11	0.53	7.66	7.05	8.21	1.48	7.88	3.41	9.80	3.82
Seq. 05	2676	46.13	19.20	59.51	16.99	2.85	1.05	8.49	3.77	6.84	3.64	13.73	5.36
Seq. 06	1099	49.57	20.79	65.39	22.33	20.01	10.83	3.08	1.16	7.77	4.25	5.76	1.92
Seq. 07	993	37.75	25.42	67.67	29.85	7.26	4.93	7.13	3.38	6.05	3.53	9.60	5.33
Seq. 10	1127	30.91	15.28	35.37	10.18	11.13	9.48	5.08	2.47	17.53	7.73	2.65	1.15

We continually employ CoVIO on five KITTI sequences after initialization on Cityscapes. The number of images corresponds to the number of update batches of a sequence. The cells highlighted in gray denote the results of the current adaptation step. Along one row, we can measure forward and backward transfer.

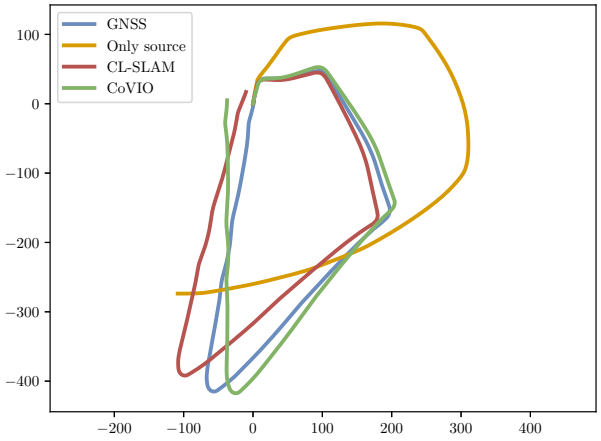


Figure 6. Continual odometry estimation results on in-house data.

For further details on the RobotCar sequences, we refer the reader to [30]. In Tab. 3, we report the adaptation quality (AQ) and the retention quality (RQ) as introduced by Vödisch *et al*. [30]. Broadly, the AQ score measures the ability of a method to adapt to a previously unseen environment, whereas the RQ measures the ability to retain long-term knowledge when being redeployed to a previously seen domain. Compared to CL-SLAM, CoVIO improves the AQ score and, with a high margin, the RQ with respect to the translation error. Although RQ_{rot} suffers from a small decrease, the absolute rotation errors on the four considered sequences are smaller than those of CL-SLAM, hence smaller differences between with and without an intermediate domain influence the RQ more strongly.

4.3. Ablation Study

In this section, we present the results of various ablation studies substantiating the design choices for the sizes of the update batch $\mathbf{b}_t$ and the replay buffer $\mathbf{B}$. We further demonstrate that CoVIO is less sensitive to the number of back-propagation steps per update batch than a previous method. In the following studies, we always report the translation and rotation errors of sequences 04 and 06.

Table 3. Comparison of adaptation and retention quality.

Previous sequences	Current sequence	CL-SLAM [30] t_{err}	CL-SLAM [30] r_{err}	CoVIO *(ours)* t_{err}	CoVIO *(ours)* r_{err}
c_t	k_9	**2.50**	**0.37**	3.89	1.49
c_t	r_1	28.94	5.63	**6.62**	**2.61**
$c_t \rightarrow r_1$	k_9	**3.24**	**0.54**	4.09	1.18
$c_t \rightarrow k_9$	r_1	30.13	5.87	**11.00**	**3.44**
$\Rightarrow AQ_{trans}$ / AQ_{rot}		0.85	0.98	**0.94**	**0.99**
$c_t \rightarrow k_9 \rightarrow r_1$	k_{10}	4.85	1.59	**1.86**	**0.70**
$c_t \rightarrow k_9 \rightarrow r_1 \rightarrow k_{10}$	r_2	20.50	4.77	**5.66**	**3.99**
$c_t \rightarrow k_9$	k_{10}	7.48	1.63	**1.43**	**0.73**
$c_t \rightarrow k_9 \rightarrow r_1$	r_2	16.41	4.58	**7.67**	**3.42**
$\Rightarrow RQ_{trans}$ / RQ_{rot} $\times 10^{-3}$		-7.30	**-0.42**	**7.89**	-1.53

Adaptation quality (AQ) and retention quality (RQ) [30] with respect to the translation and rotation errors. c_t denotes the Cityscapes training set, k_9 and k_{10} refer to sequences 09 and 10 of the KITTI odometry benchmark, and r_1 and r_2 correspond to sequences [30] from RobotCar. The values of CL-SLAM are reported by the authors. The best scores in each category are shown in **bold**.

4.3.1 Size of the Update Batch

We first investigate the effect of varying sizes of the update batch, *i.e*., the number of replay samples. In Tab. 4, we list the errors for batch sizes $\mathbf{b}_t = \{1, 2, 3, 4, 5\}$ given an unlimited replay buffer. Note that $\mathbf{b} = 1$ implies that experience replay is disabled. Therefore, this strategy corresponds to CL-SLAM [30] for source-to-target domain adaptation. Generally, the translation error can be reduced by using replay data, whereas the rotation error is the smallest when only training with the current sample. As we deem the translation error more important in terms of mapping and localization accuracy, we select $\mathbf{b} = 3$.

4.3.2 Size of the Replay Buffer

In the next study, we restrict the size of the replay buffer to address both scalability of the method and the limited storage capacity on mobile devices and robotic platforms. In detail, we report the error for buffer sizes $|\mathbf{B}| = \{10, 25, 50, 100, \infty\}$ in Tab. 5. For all the buffers of limited

Table 4. Ablation study on the size of the update batch.

Batch size	Seq. 04		Seq. 06	
	t_{err}	r_{err}	t_{err}	r_{err}
1	3.56	**0.15**	2.30	**0.18**
2	2.97	0.59	**1.81**	0.50
3	**2.79**	0.54	1.98	0.59
4	2.89	0.73	1.99	0.54
5	2.89	0.63	2.46	0.70

In this study, we use a replay buffer of infinite size. Batch sizes greater than 1 imply using replay data in addition to the online image, *i.e.*, the first row corresponds to the strategy of CL-SLAM. The smallest and second smallest errors are shown in **bold** and underlined, respectively.

Table 5. Ablation study on the size of replay buffer.

Buffer size	Diversity update	Seq. 04		Seq. 06	
		t_{err}	r_{err}	t_{err}	r_{err}
∞		2.79	0.54	1.98	0.59
100		2.62	0.52	**1.75**	0.48
100	✓	**2.11**	0.53	2.13	**0.47**
50		2.64	0.42	2.72	0.91
50	✓	**2.11**	0.53	2.13	**0.47**
25		2.51	0.40	2.42	0.77
25	✓	**2.11**	0.53	2.20	0.50
10		2.82	**0.33**	2.08	0.64
10	✓	2.12	0.56	2.21	0.59

In this study, we use a batch size of 3. The effectively used buffer size of sequence 04 is the same for 25, 50, and 100. Similarly, in sequence 06 the same number of samples is added when buffer sizes of 50 and 100 are available. The first row corresponds to the strategy of CL-SLAM. The smallest and second smallest errors are shown in **bold** and underlined, respectively.

size, we both enable and disable our proposed diversity-based updating mechanism. Interestingly, the positive effect of enforcing a high diversity is more pronounced for sequence 04, where CoVIO generally yields smaller translation errors with fewer samples in the buffer. It should further be noted that due to the length of the sequences, the diversity-based buffer contains the same samples for $|\mathbf{B}_{\text{Seq. 04}}| = \{25, 50, 100\}$ and $|\mathbf{B}_{\text{Seq. 06}}| = \{50, 100\}$. For CoVIO, we select $|\mathbf{B}| = 100$ to account for the increased storage requirements in a multi-target setting.

4.3.3 Number of Update Cycles

Lastly, we report the sensitivity of CoVIO with respect to the number of backpropagation steps c for a single update batch. As shown in Tab. 6, more steps decrease both the errors confirming the results of CL-SLAM [30]. However, in contrast to the performance reported for CL-SLAM, CoVIO is noticeably less sensitive and already yields relatively small errors for $c = 1$. We conclude that this is caused by using experience replay also for the online learner. To enable direct comparison with CL-SLAM, we use the same value $c = 5$.

Table 6. Ablation study on the number of update cycles.

Update cycles	Seq. 04		Seq. 06	
	t_{err}	r_{err}	t_{err}	r_{err}
1	3.13	1.65	4.07	1.14
2	2.80	1.06	2.62	0.64
3	2.49	0.77	2.61	0.63
4	2.29	0.61	2.31	0.52
5	2.11	0.53	**2.13**	**0.47**
6	**1.98**	**0.50**	2.15	0.49

For a fair comparison, we use the same number of update cycles $c = 5$ as CL-SLAM [30]. The smallest and second smallest errors are shown in **bold** and underlined, respectively.

5. Conclusion

In this paper, we presented CoVIO for online continual learning of visual-inertial odometry. CoVIO exploits an unsupervised training scheme during inference time and thus seamlessly adapts to new domains. In particular, we explicitly address the shortcomings of previous works by designing a lightweight network architecture and by proposing a novel fixed-size replay buffer that maximizes image diversity. Using experience replay, CoVIO successfully mitigates catastrophic forgetting while achieving efficient online domain adaptation. We further provide an asynchronous version of CoVIO separating the core motion estimation from the network update step, hence allowing true continuous inference in real time. In extensive evaluations, we demonstrated that CoVIO outperforms competitive baselines. We also made the code and models publicly available to facilitate future research. Future work will focus on extending this work to a multi-task setup for robotic vision systems.

Acknowledgment

This work was funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 871449-OpenDR. We thank Kürsat Petek for the extensive discussions concerning the replay buffer and for supporting the collection of our in-house data. We further thank Ahmet Selim Çanakçı for helping us with the implementation of the asynchronous version of CoVIO.

References

[1] Borna Bešić and Abhinav Valada. Dynamic object removal and spatio-temporal RGB-D inpainting via geometry-aware adversarial learning. *IEEE Transactions on Intelligent Vehicles*, 7(2):170–185, 2022. 1

[2] Borna Bešić, Nikhil Gosala, Daniele Cattaneo, and Abhinav Valada. Unsupervised domain adaptation for LiDAR panop-

tic segmentation. *IEEE Robotics and Automation Letters*, 7(2):3404–3411, 2022. 1, 2

[3] 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 *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 3213–3223, 2016. 5

[4] Changhong Fu, Adrian Carrio, and Pascual Campoy. Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. In *International Conference on Unmanned Aircraft Systems*, pages 957–962, 2015. 1

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 5

[6] Clement Godard, Oisin Mac Aodha, Michael Firman, and Gabriel Brostow. Digging into self-supervised monocular depth estimation. In *International Conference on Computer Vision*, pages 3827–3837, 2019. 1, 2, 3

[7] Vitor Guizilini, Rareș Ambruș, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3D packing for self-supervised monocular depth estimation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, 2020. 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 770–778, 2016. 3

[9] Pan Ji, Qingan Yan, Yuxin Ma, and Yi Xu. Georefine: Self-supervised online depth refinement for accurate dense mapping. In *European Conference on Computer Vision*, pages 360–377, 2022. 2

[10] Yevhen Kuznietsov, Marc Proesmans, and Luc Van Gool. CoMoDA: Continuous monocular depth adaptation using past experiences. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021. 2, 4

[11] Yevhen Kuznietsov, Marc Proesmans, and Luc Van Gool. Towards unsupervised online domain adaptation for semantic segmentation. In *European Conference on Computer Vision*, pages 261–271, 2022. 2

[12] Ruihao Li, Sen Wang, and Dongbing Gu. DeepSLAM: A robust monocular SLAM system with unsupervised deep learning. *IEEE Transactions on Industrial Electronics*, 68(4):3577–3587, 2021. 4, 6

[13] Shunkai Li, Xin Wang, Yingdian Cao, Fei Xue, Zike Yan, and Hongbin Zha. Self-supervised deep visual odometry with online adaptation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, 2020. 1, 2

[14] Shunkai Li, Xin Wu, Yingdian Cao, and Hongbin Zha. Generalizing to the open world: Deep visual odometry with online adaptation. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 13179–13188, 2021. 2

[15] Shing Yan Loo, Moein Shakeri, Sai Hong Tang, Syamsiah Mashohor, and Hong Zhang. Online mutual adaptation of deep depth prediction and visual slam. *arXiv preprint arXiv:2111.04096*, 2021. 2

[16] David Lopez-Paz and Marc' Aurelio Ranzato. Gradient episodic memory for continual learning. In *Conference on Neural Information Processing Systems*, volume 30, 2017. 2

[17] Mayank Lunayach, James Smith, and Zsolt Kira. Lifelong wandering: A realistic few-shot online continual learning setting. *arXiv preprint arXiv:2206.07932*, 2022. 2

[18] Hongcheng Luo, Yang Gao, Yuhao Wu, Chunyuan Liao, Xin Yang, and Kwang-Ting Cheng. Real-time dense monocular SLAM with online adapted depth prediction network. *IEEE Transactions on Multimedia*, 21(2):470–483, 2019. 1, 2

[19] Will Maddern, Geoffrey Pascoe, Matthew Gadd, Dan Barnes, Brian Yeomans, and Paul Newman. Real-time kinematic ground truth for the oxford robotcar dataset. *arXiv preprint arXiv:2002.10152*, 2020. 5

[20] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000km: The Oxford RobotCar dataset. *International Journal of Robotics Research*, 36(1):3–15, 2017. 5

[21] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007. 1

[22] Robert McCraith, Lukas Neumann, Andrew Zisserman, and Andrea Vedaldi. Monocular depth estimation with self-supervised instance adaptation. *arXiv preprint arXiv:2004.05821*, 2020. 4

[23] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 6

[24] Thomas Schöps, Jakob Engel, and Daniel Cremers. Semi-dense visual odometry for ar on a smartphone. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 145–150, 2014. 1

[25] Abu Md Niamul Taufique, Chowdhury Sadman Jahan, and Andreas Savakis. Unsupervised continual learning for gradually varying domains. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 3739–3749, 2022. 2

[26] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Conference on Neural Information Processing Systems*, volume 8, 1995. 2

[27] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Deep auxiliary learning for visual localization and odometry. In *IEEE International Conference on Robotics and Automation*, pages 6939–6946, 2018. 1

[28] Abhinav Valada, Noha Radwan, and Wolfram Burgard. Incorporating semantic and geometric priors in deep pose regression. *Workshop on learning and inference in robotics: Integrating structure, priors and models at robotics: Science and systems (RSS)*, 1(3), 2018. 1

[29] Francisco Rivera Valverde, Juana Valeria Hurtado, and Abhinav Valada. There is more than meets the eye: Self-supervised multi-object detection and tracking with sound by distilling multimodal knowledge. In *IEEE/CVF Conference Computer Vision and Pattern Recognition*, pages 11607–11616, 2021. 2

[30] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Continual SLAM: Beyond lifelong simultaneous localization and mapping through continual learning.

In Aude Billard, Tamim Asfour, and Oussama Khatib, editors, *Robotics Research*, pages 19–35, Cham, 2023. Springer Nature Switzerland. 1, 2, 3, 4, 5, 6, 7, 8

[31] Jianren Wang, Xin Wang, Yue Shang-Guan, and Abhinav Gupta. Wanderlust: Online continual object detection in the real world. In *International Conference on Computer Vision*, pages 10809–10818, 2021. 2