





Receding Moving Object Segmentation in 3D LiDAR Data Using Sparse 4D Convolutions

Benedikt Mersch , Graduate Student Member, IEEE, Xieyuanli Chen , Ignacio Vizzo , Student Member, IEEE, Lucas Nunes , Graduate Student Member, IEEE, Jens Behley , and Cyrill Stachniss , Member, IEEE

Abstract—A key challenge for autonomous vehicles is to navigate in unseen dynamic environments. Separating moving objects from static ones is essential for navigation, pose estimation, and understanding how other traffic participants are likely to move in the near future. In this work, we tackle the problem of distinguishing 3D LiDAR points that belong to currently moving objects, like walking pedestrians or driving cars, from points that are obtained from non-moving objects, like walls but also parked cars. Our approach takes a sequence of observed LiDAR scans and turns them into a voxelized sparse 4D point cloud. We apply computationally efficient sparse 4D convolutions to jointly extract spatial and temporal features and predict moving object confidence scores for all points in the sequence. We develop a receding horizon strategy that allows us to predict moving objects online and to refine predictions on the go based on new observations. We use a binary Bayes filter to recursively integrate new predictions of a scan resulting in more robust estimation. We evaluate our approach on the SemanticKITTI moving object segmentation challenge and show more accurate predictions than existing methods. Since our approach only operates on the geometric information of point clouds over time, it generalizes well to new, unseen environments, which we evaluate on the Apollo dataset.

Index Terms—Semantic scene understanding, deep learning methods.

I. INTRODUCTION

DISTINGUISHING moving from static objects in 3D LiDAR data is a crucial task for autonomous systems and required for planning collision-free trajectories and navigating safely in dynamic environments. Moving object segmentation can improve localization [5], [7], planning [34], mapping [5], scene flow estimation [2], [15], [37], or the prediction of future states [38], [40]. There are mapping approaches that identify if observed points are *potentially moving* or *have moved* throughout the mapping process [1], [7], [16], [28]. On the contrary,

Manuscript received February 24, 2022; accepted June 7, 2022. Date of publication June 15, 2022; date of current version June 27, 2022. This letter was recommended for publication by Associate Editor Y. Yang and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy under Grant EXC-2070-390732324-PhenoRob. (Corresponding author: Benedikt Mersch.)

Benedikt Mersch, Xieyuanli Chen, Ignacio Vizzo, Lucas Nunes, and Jens Behley are with the University of Bonn, 53113 Bonn, Germany (e-mail: mersch@igg.uni-bonn.de; xieyuanli.chen@igg.uni-bonn.de; ivizzo@uni-bonn.de; lucas.nunes@uni-bonn.de; jens.behley@igg.uni-bonn.de).

Cyrill Stachniss is with the University of Bonn, 53113 Bonn, Germany, and also with the Department of Engineering Science, University of Oxford, OX1 2JD Oxford, U.K. (e-mail: cyrill.stachniss@igg.uni-bonn.de).

Digital Object Identifier 10.1109/LRA.2022.3183245

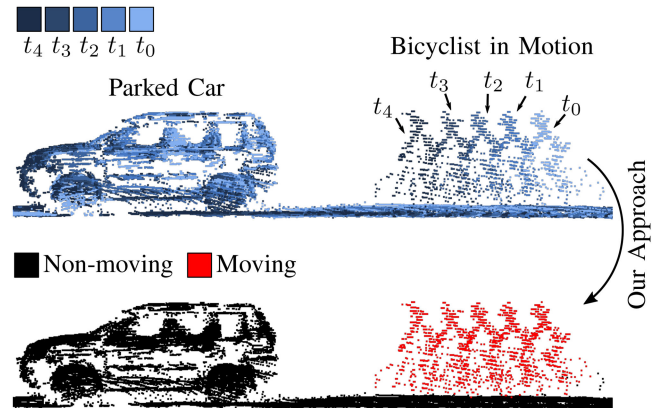


Fig. 1. Given a sequence of point clouds, our method identifies that the points belonging to the bicyclist move in space over time. *Top*: Input sequence with points colorized (blue) with respect to the time step. The darker the blue color, the newer the scan. *Bottom*: Our method successfully predicts the bicyclist as moving (red) and the parked car as static (black).

identifying objects that are *actually moving* within a short time horizon are of interest for online navigation [34], can improve scene flow estimation between two consecutive point clouds [2], [15], [37], or support predicting a future state of the environment [40].

In this work, we focus on the task of segmenting moving objects online using a limited time horizon of observations. Given a sequence of 3D LiDAR scans, we predict for each point if it belongs to a moving object, for example bicyclists or driving cars, or a static, i.e., non-moving one, like parked cars, buildings, or trees.

In contrast to the task of semantic segmentation, moving object segmentation in 3D LiDAR data does not require a complex notion of semantic classes with extensive labeling to supervise learning-based methods or to evaluate their performance. Instead, the goal is to predict if a local point cloud structure moves throughout space and time or remains static as visualized in Fig. 1. In general, the task requires the extraction of temporal information from the LiDAR sequence to decide which points are moving and which are not. Previous works tackled this problem by extracting temporal information from residual range images [5] or bird's eye view (BEV) images [26], typically using a 2D convolutional neural network (CNN). The back-projection from these 2D representations to the 3D space often requires post-processing like k-nearest neighbor (KNN) clustering [5], [9], [12], [25] to avoid labels bleeding into points that are close in

the image space but distant in 3D. Other approaches can identify objects that have moved in 3D space directly during mapping [1] or with a clustering and tracking approach [6]. Nevertheless, these offline methods often rely on having access to all LiDAR observations in the sequence.

The main contribution of this paper is a novel approach that predicts moving objects online for a short sequence of LiDAR scans. We exploit sparse 4D convolutions to jointly extract spatio-temporal features from the input point cloud sequence. The outputs of our network are moving object confidence scores for the points in each input scan. Since we directly predict in a voxelized sparse 4D space, we do not require any back-projection and clustering to retrieve per-point predictions. Our method operates in a sliding window fashion and appends a new observed scan to the input sequence while discarding the oldest one. By doing so, our method can include new observations into the estimation as they arrive. We implemented a binary Bayes filter to fuse these predictions and in this way increase the robustness to false predictions. Since our method uses only the spatial point information over time, it is class agnostic and generalizes well on unseen data.

In sum, we make three key claims: Our approach i) segments moving objects in LiDAR data more accurately compared to existing methods, ii) generalizes well to unseen environments without additional domain adaptation techniques, and iii) improves the results by integrating online new observations. We back explicitly up these three claims by our experimental evaluation. The code of this paper as well as our pre-trained models will be available at.¹

II. RELATED WORK

We can group LiDAR-based moving object segmentation methods with respect to their definition of dynamic objects. Besides map cleaning methods [17], [28], there are also mapping approaches that remove objects that have moved throughout the mapping process from the data before fusing them with the map [1], [7], [16]. For example, Wang *et al.* [39] apply graph-based clustering to segment objects that could move in 3D LiDAR data. Ruchti and Burgard [30] use a deep neural network to predict dynamic probabilities for each point in a range image before fusing them with a map. In contrast to that, Thomas *et al.* [34] proposed a self-supervised method for classifying indoor LiDAR points into dynamic labels. The authors explicitly distinguish between short-term and long-term movable objects to treat them differently in localization and planning. Arora *et al.* [1] explore ground segmentation with ray-casting to coarsely remove dynamic objects in LiDAR scans. Other researchers encode non-static objects into the map by estimating multi-modal states [33]. Recently, Chen *et al.* [6] propose a pipeline to automatically label moving objects offline. They first use an occupancy-based method to find dynamic point candidates and further identify moving objects by sequential clustering and tracking. Instead of removing all long-term changes caused by objects that have moved, our method segments motion online

and focuses on objects that are actually moving within a limited time horizon.

Previously, scene-flow methods first classify moving points and then estimate separate flows for static and moving objects between two point clouds [2], [15], [37]. In more detail, Baur *et al.* [2] estimate the 3D scene flow between two point clouds composed of a rigid body motion for static and a per-point flow for moving objects. They use a self-supervised motion segmentation signal based on the discrepancy between per-point flow and rigid body motion for training their network. Even though moving object segmentation can be a by-product of scene flow estimation, most methods only consider two subsequent frames which could be a too short time horizon for classifying slowly moving objects.

Other methods primarily focus on segmenting moving objects online using a larger time horizon. To cope with the computational effort of 3D point cloud sequences, projection-based methods have been proposed. Chen *et al.* [5] developed LMNet, which exploits existing single-scan semantic segmentation networks that get residual range images as additional inputs to extract temporal information. Recently, Mohapatra *et al.* [26] introduced a method using BEV images for moving object segmentation and achieve faster runtime but inferior performance compared to LMNet. Projection-based methods often suffer from information loss or back-projection artifacts and require additional steps like kNN clustering [5], [9], [12], [25]. In contrast, our method directly predicts in 4D space and does not require any post-processing techniques.

Extracting temporal information from sequential point cloud data is gaining more attention in research since it allows to increase temporal consistency for classification tasks or to predict future states of the environment. To fuse independent semantic single-scan predictions, Dewan and Burgard [10] use a binary Bayes filter by propagating previous predictions to the next scan using scene flow. In contrast, Duerr *et al.* [12] optimize a recurrent neural network to temporally align range image features from a single-scan semantic segmentation network. Some works project the spatial information into 2D representations like range images [5], [12], [19], [24] or BEV images [23], [26], [40] and then apply 2D or 3D convolutions to reduce the computational burden of jointly processing 4D data. Besides point-based methods [13], [14], [21] for processing point cloud sequences, representing point clouds as sparse tensors can also circumvent the back-projection issue and makes it possible to apply sparse convolutions efficiently. For example, Shi *et al.* [32] propose SpSequenceNet for 4D semantic segmentation which processes two LiDAR frames with sparse 3D convolutions and combines their temporal information with a cross-frame global attention module. To apply convolutions across time, Choy *et al.* [8] propose Minkowski networks for semantic segmentation using sparse 4D convolutions on temporal RGB-D data.

In this paper, we propose a novel moving object segmentation method that jointly applies sparse 4D convolutions on a sequence of LiDAR point clouds building on top of the Minkowski engine [8]. Unlike previous methods, we operate online and do not need a pre-built map representation. Whereas most classification methods output one prediction for each frame, we

¹ [Online] Available: <https://github.com/PRBonn/4DMOS>

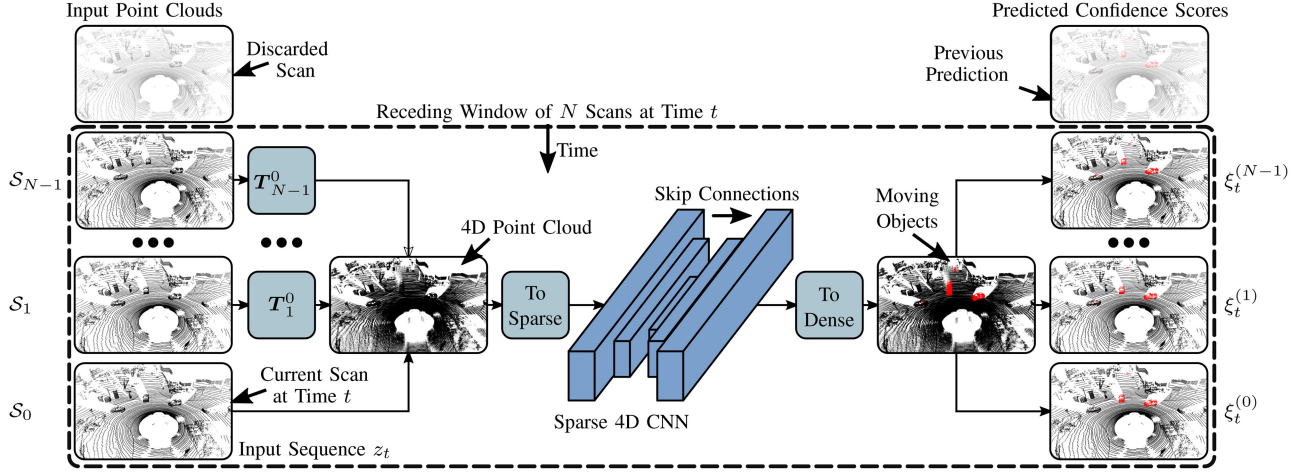


Fig. 2. Overview of our approach operating with a receding horizon strategy. We transform all scans of the considered receding window to the current viewpoint. Next, we aggregate all points and create a sparse 4D point cloud. We apply sparse 4D convolutions to jointly extract spatio-temporal features. Our final layer predicts moving object confidence scores for all points in the input sequence.

propose to predict moving objects using a receding horizon strategy. This allows us to incorporate new observations in an online fashion and refine predictions more robustly by Bayesian filtering.

III. OUR APPROACH

Given a point cloud sequence $\mathcal{S} = \{\mathcal{S}_j\}_{j=0}^{N-1}$ of N LiDAR scans $\mathcal{S}_j = \{\mathbf{p}_i \in \mathbb{R}^4\}_{i=0}^{M_j-1}$ with M_j points represented as homogeneous coordinates, i.e., $\mathbf{p}_i = [x_i, y_i, z_i, 1]^\top$, the goal of our approach is to predict, which points are actually moving in the input sequence \mathcal{S} . We denote the current scan as \mathcal{S}_0 and index the previous past scans from 1 to $N-1$.

As shown in Fig. 2, we first transform the past point clouds $\mathcal{S}_1, \dots, \mathcal{S}_{N-1}$ to the viewpoint of the current scan \mathcal{S}_0 and create a sparse 4D tensor, see Section III-A. We extract spatio-temporal features with a sparse convolutional architecture and predict confidence scores of being actually moving for each point in the sequence, see Section III-B. As soon as we obtain a new LiDAR scan, we shift the prediction window as explained in Section III-C. The receding horizon strategy allows to recursively update the estimation by fusing later predictions for the same scan in the sequence in a binary Bayes filter, see Section III-D.

A. Input Representation

The first step is to locally align all past point clouds $\mathcal{S}_1, \dots, \mathcal{S}_{N-1}$ in the sequence \mathcal{S} to the viewpoint of the current LiDAR scan \mathcal{S}_0 . In this work, we assume to have access to *estimated* relative pose transformations \mathbf{T}_j^{j-1} between scans \mathcal{S}_{j-1} and \mathcal{S}_j . Odometry estimation is a standard task for autonomous vehicles and can be efficiently solved on-board with an online SLAM system like SuMa [4] and further improved by integrating information from an inertial measurement unit [31] or by using wheel encoders. Our approach is agnostic to the source of odometry information and a local consistency is sufficient such that obtaining this data is not a problem in practice. We represent

the relative transformations between scans $\mathbf{T}_1^0, \dots, \mathbf{T}_N^{N-1}$ as transformation matrices, i.e., $\mathbf{T}_j^{j-1} \in \mathbb{R}^{4 \times 4}$. Further, we denote the j^{th} scan transformed to the current viewpoint by

$$\mathcal{S}^{j \rightarrow 0} = \{\mathbf{T}_j^0 \mathbf{p}_i\}_{\mathbf{p}_i \in \mathcal{S}_j} \quad \text{with} \quad \mathbf{T}_j^0 = \prod_{k=0}^{j-1} \mathbf{T}_{j-k}^{j-k-1}. \quad (1)$$

The motivation behind locally aligning the scans in the sequence is that our CNN should focus on local point patterns that move in space over time and for that, pose information helps. We also provide an experimental analysis on the effect of the pose alignment in Section IV-E. After applying the transformations, we aggregate the aligned scans into a 4D point cloud by converting from homogeneous coordinates to cartesian coordinates and by adding the time as an additional dimension resulting in coordinates $[x_i, y_i, z_i, t_i]^\top$ for point \mathbf{p}_i .

Since outdoor point clouds obtained from a LiDAR sensor are sparse by nature, we quantize the 4D point cloud into a sparse voxel grid with a fixed resolution in time Δt and space Δs . We use a sparse tensor to represent the voxel grid and store the indices and associated features of non-empty voxels only. Sparse tensors are more memory efficient compared to dense voxel grids since they only store information about the voxels that are actually occupied by points. The sparse representation allows us to use spatio-temporal CNNs efficiently since common dense 4D convolutions become intractable on large scenes.

B. Sparse 4D Convolutions

Using the sparse input representation discussed in Section III-A, we can apply time- and memory-efficient sparse 4D convolutions to jointly extract spatio-temporal features from the sparse 4D occupancy grid and predict a moving object confidence score for each point. To this end, we use the Minkowski engine [8] for sparse convolutions. Sparse convolutions operate on the sparse tensor and define kernel maps that specify how the kernel weights connect the input and output coordinates.

The main advantage of sparse convolutions is the computational speed-up compared to dense convolutions.

We use a sparse convolutional network developed for 4D semantic segmentation on RGB-D data and adapt it for moving object segmentation on LiDAR data. More specifically, we use a modified MinkUNet14 [8], which is a sparse equivalent of a residual bottleneck architecture with strided sparse convolutions for downsampling the feature maps and strided sparse transpose convolutions for upsampling. The skip connections in a UNet fashion [29] help to maintain details and fine-grained predictions. We reduce the number of feature channels in the network resulting in a model with 1.8 M parameters, which is comparably low compared to the moving object segmentation baseline LM-Net [5] with SalsaNext [9] (6.7 M) or RangeNet++ [25] (50 M) backbones. The last layer of our network is a 4D sparse convolution with a softmax that predicts moving object confidence scores between 0 and 1 for each point.

In contrast to 4D semantic segmentation methods that use RGB values as input features [8], we initialize voxels occupied by at least one point with a constant feature of 0.5. Therefore, our input is a sparse 4D occupancy grid only storing voxels occupied by a point. This makes it easier to deploy the approach in new environments without estimating the distribution of coordinates or intensity values to standardize the input data as done for semantic segmentation [25]. The generalization capability of our approach is further investigated in Section IV-C.

C. Receding Horizon Strategy

The fully sparse convolutional architecture introduced in Section III-B jointly predicts moving object confidence scores for all points in the input sequence. At inference time, one option would be to divide the input data into fixed, non-overlapping intervals and to predict each sub-sequence once.

Instead, we propose a different strategy and develop a receding horizon strategy for moving object segmentation. When the LiDAR sensor obtains the next point cloud, we add it to the input sequence and discard the oldest scan resulting in a first in, first out queue, see Fig. 2. The main advantage is that we can re-estimate moving objects based on new observations and therefore increase the time horizon used for prediction. It is a natural idea to use multiple observations to reduce the uncertainty of semantic estimations and has been well investigated in mapping algorithms like SuMa++ [7]. It is still rarely used for online segmentation, and we propose a method to improve the online moving object segmentation.

D. Binary Bayes Filter

Since our proposed method predicts moving objects in N scans at once, the receding horizon strategy leads to a re-estimation of the previously predicted $N-1$ scans. These multiple predictions from different time steps allow refining the estimation of moving objects based on new observations. We propose to fuse them recursively using a binary Bayes filter. This makes it possible to increase the time horizon used for segmentation and helps to predict slowly moving objects that only moved a small distance within the initial time horizon.

The Bayesian fusion reduces the number of false positives and negatives that arise due to occlusions or noisy measurements.

More formally, for a scan \mathcal{S}_j , we can estimate moving objects at time t by fusing all predicted moving object confidence scores from previously observed point cloud sequences $z_{0:t}$ that contain the scan \mathcal{S}_j . The term z_t denotes the observed input point cloud sequence $\mathcal{S}_0, \dots, \mathcal{S}_{N-1}$ with \mathcal{S}_0 recorded at time t . We want to estimate the joint probability distribution of the moving state $m^{(j)}$ of all points up to time t denoted by

$$p\left(m^{(j)} \mid z_{0:t}^{(j)}\right) = \prod_i p\left(m_i^{(j)} \mid z_{0:t}^{(j)}\right), \quad (2)$$

where $m_i^{(j)} \in \{0, 1\}$ is the state of point $\mathbf{p}_i \in \mathcal{S}_j$ being moving in the scan \mathcal{S}_j . For better readability, we will from now on consider a single point \mathbf{p}_i in point cloud \mathcal{S}_j and omit the superscript j without loss of generality.

We apply Bayes' rule to the per-point probability distribution $p(m_i \mid z_{0:t})$ in (2) and follow the standard derivation of the recursive binary Bayes filter [36]. Using the log-odds notation $l(x) = \log \frac{p(x)}{1-p(x)}$ commonly used in occupancy grid mapping, we finally end up with

$$l(m_i \mid z_{0:t}) = \begin{cases} l(m_i \mid z_{0:t-1}) + l(m_i \mid z_t) - l(m_i), & \text{if } t \in \mathcal{T} \\ l(m_i \mid z_{0:t-1}), & \text{otherwise,} \end{cases} \quad (3)$$

with \mathcal{T} being the set of time steps in which we observe point \mathbf{p}_i in the input sequence z_t . Whereas $l(m_i \mid z_{0:t-1})$ is a recursive term including all predictions for the point i up to time $t-1$, the term $l(m_i \mid z_t)$ denotes the log-odds of the probability to be moving at time t . Note that if we do not observe the point \mathbf{p}_i at time t , there is no prediction and we do not update the recursive term $l(m_i \mid z_{0:t-1})$. The prior probability $p_0 \in (0, 1)$ in the last part $l(m_i) = \log \frac{p_0}{1-p_0}$ provides a measure of the innovation introduced by a new prediction. For moving object segmentation, the prior determines how much a predicted moving point in a single scan influences the final prediction. We will investigate different priors in Section IV-E.

At time t , our network outputs moving object confidence scores $\xi_t^{(j)} \in \{\xi_{t,i}^{(j)}\}_{i=0}^{M_j-1}$ with $\xi_{t,i} \in (0, 1)$ for each point cloud \mathcal{S}_j with M_j points in the current input sequence z_t . We can interpret the predicted confidence score $\xi_{t,i}$ for a single point \mathbf{p}_i given the input sequence z_t as posterior probability reading

$$\xi_{t,i} = p(m_i = 1 \mid z_t). \quad (4)$$

The log-odds expression of the confidence score in (3) is then given as

$$l(m_i \mid z_t) = \log \frac{\xi_{t,i}}{1 - \xi_{t,i}}. \quad (5)$$

Fig. 3 illustrates the non-overlapping strategy in the upper part and our proposed receding horizon strategy with a binary Bayes filter to fuse multiple predictions in the lower part. We obtain the final prediction by converting the recursively estimated per-point log-odds $l(m_i \mid z_{0:t-1})$ to confidence score using $p(x) = \log \frac{l(x)}{1+l(x)}$. If the confidence is larger than 0.5, we predict the point to be moving and otherwise non-moving.

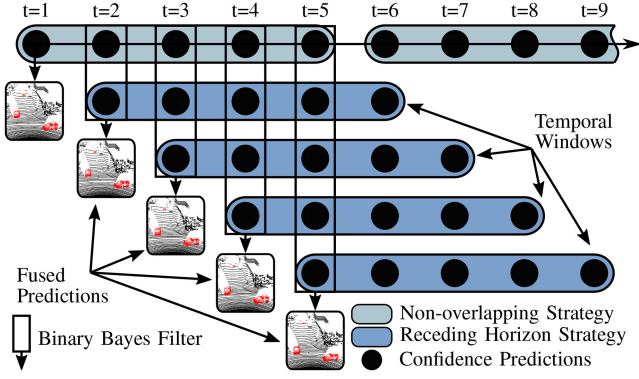


Fig. 3. Overview of our proposed binary Bayes filter. At $t=5$, the non-overlapping strategy uses the five per-scan confidence predictions, whereas our receding horizon strategy integrates the next observation at $t=6$ by shifting the temporal window. Our binary Bayes filter then fuses multiple moving object confidence scores to improve the prediction.

IV. EXPERIMENTAL EVALUATION

The main focus of this work is a method to segment actually moving objects in 3D LiDAR data by exploiting consecutive scans in an online fashion. Additionally, we carry out the prediction using a receding horizon strategy and integrate new predictions recursively in a binary Bayes filter.

We present our experiments to show the capabilities of our method and to support our three key claims: Our approach i) segments moving objects in LiDAR data more accurately compared to existing methods, ii) generalizes well to unseen environments without additional domain adaptation techniques, and iii) improves the results by integrating online new observations.

A. Experimental Setup

For our experimental evaluation, we train all models on the SemanticKITTI [3] dataset. We use sequences 00–07 and 09–10 for training, 08 for validation, and 11–21 for testing. During training, we optimize the model with a binary cross-entropy loss for all points in the input sequence and a learning rate of 0.0001 and a weight decay of 0.0001 with the Adam optimizer [18]. If not stated differently in the experiments, our input point clouds sequences contain $N=10$ input scans with a temporal resolution of $\Delta t = 0.1$ s. The spatial voxel size for quantization is $\Delta s = 0.1$ m. To increase the diversity of the training data and to avoid overfitting, we follow the data augmentation of Nunes *et al.* [27] and apply random rotations, shifting, flipping, jittering, and scaling to all points in the same 4D point cloud. We train all networks for less than 60 epochs and keep the model with the best performance on the validation set. We follow the receding horizon strategy presented in Section III-C and combine predictions with the binary Bayes filter proposed in Section III-D using a prior of $p_0=0.25$.

For quantitative evaluation, we report the standard intersection-over-union (IoU) metric [11] for the moving class given by

$$\text{IoU}_{\text{MOS}} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \quad (6)$$

TABLE I
PERFORMANCE ON SEMANTICKITTI [3] MOVING OBJECT SEGMENTATION BENCHMARK [5]. BASELINE RESULTS TAKEN FROM [6]. BEST RESULT IN BOLD

	IoU _{MOS} [%]
SalsaNext [9] (movable classes)	4.4
SceneFlow [20]	4.8
SpSequenceNet [32]	43.2
LMNet [5]	58.3
KPConv [35]	60.9
Ours, $N=10$ Scans, $\Delta t=0.1$ s, $p_0=0.25$	65.2
LMNet+AutoMOS+Extra [6]	62.3

with true positive TP, false positive FP, and false negative FN classifications of moving points.

To evaluate the generalization capability of our approach across environments, we additionally test it on another dataset without the use of domain adaptation techniques. We follow the setup of Chen *et al.* [6] and use the Apollo-ColumbiaParkMapData [22] dataset sequence 2 (frames 22300–24300) and sequence 3 (frames 3100–3600) annotated the same way as SemanticKITTI. Note that SemanticKITTI and Apollo both use Velodyne HDL-64E LiDAR scanners, but they are mounted on a different car at a different height and recorded data in a different environment.

B. Moving Object Segmentation Performance

Our first experiment evaluates the performance of our model on the SemanticKITTI [3] moving object segmentation benchmark [5]. The results support the first claim about segmenting moving objects more accurately compared to existing methods that are published and open-source. For a fair comparison, we follow the setup from LMNet [5] and use the provided SemanticKITTI poses estimated with an online SLAM system [4]. We report the result on the hidden test set in Table I and compare it to additional baselines provided by Chen *et al.* [5].

One can see that single-scan segmentation with SalsaNext [9] and predicting all movable classes as moving leads to low performance of 4.4% IoU_{MOS}. The same applies to estimating scene flow and thresholding the flow vectors to determine if an object moves. The online multi-scan semantic segmentation methods SpSequenceNet [32], LMNet [5], and KPConv [35] show improved results up to 60.9% IoU_{MOS}, see Table I. Our method can outperform all baselines with an IoU_{MOS} of 65.2%, which demonstrates the effectiveness of our approach. Our performance is also better than LMNet+AutoMOS+Extra [6], which additionally uses automatically generated moving object labels for training. This emphasizes the strength of our result.

C. Generalization Capabilities

The next experiment evaluates our method's ability to generalize across different environments. It supports our second claim that the approach generalizes well on unseen data. We test our model on the Apollo dataset without using any domain adaptation techniques or re-training and compare to baselines that use different levels of domain adaptation. LMNet [5] uses the same SemanticKITTI [3] sequences for training, whereas

TABLE II
PERFORMANCE ON APOLLO [22] DATASET. BEST RESULT IN BOLD

	IoUMOS [%]
LMNet [5]	16.9
LMNet+AutoMOS [6]	45.7
LMNet+AutoMOS+Fine-Tuned [6]	65.9
Ours, $N=10$ Scans, $\Delta t=0.1$ s, $p_0=0.25$	73.1

LMNet+AutoMOS [6] is LMNet trained on an automatically labeled training set of Apollo. LMNet+AutoMOS+Fine-Tuned [6] is a model pre-trained on SemanticKITTI and fine-tuned on Apollo, see [6] for details. The results in Table II suggest that for the baselines, domain adaptation like re-training or fine-tuning improves the results with a maximum IoUMOS of 65.9%. Our method yields the highest IoUMOS of 73.1% without any additional steps, which shows that the approach is well capable of predicting moving objects in an unknown environment.

We hypothesize that extracting moving object features in a sparse 4D occupancy grid is advantageous since the method does not use any sensor-specific information like intensity or RGB values. Directly operating in 4D space also makes the network less prone to overfitting to a specific sensor location as in the case of range-images, where moving objects are usually found in certain areas of the image. We also do not use information about semantic classes whose distribution can differ between environments.

D. Receding Horizon Strategy and Fusion

This section backs up our third claim that the proposed receding horizon strategy combined with a binary Bayes filter improves the MOS results by integrating online new observations. We investigate the effect of using different numbers of input scans N and temporal resolutions Δt for prediction as well as fusing with different prior probabilities p_0 in the Bayesian fusion presented in Section III-D.

We compare models trained on $N = 2, 5$, and 10 input and output scans as well as a model that predicts a single output scan. Since the combination of a receding horizon strategy and the Bayesian fusion of multiple beliefs allows us to use information from a larger time horizon, we additionally compare to two variant setups using $N = 5$ input and output frames but with a different temporal resolution. One uses a resolution of $\Delta t = 0.2$ s resulting in a time horizon of 0.8 s, the other one processes 1.2 s of scans that are $\Delta t = 0.3$ s apart. For comparison, the method using $N = 2$ scans with a resolution of $\Delta t = 0.1$ s has a time horizon of 0.2 s, the one with $N = 5$ scans a horizon of 0.4 s and the model using $N = 10$ scans looks at 0.9 s of data. We visualize the time horizons and temporal resolutions for each variant in Fig. 4 as colored dots on a timeline sampled at 10 Hz.

The Bayesian prior p_0 in (3) serves to compute the difference between the new predicted log-odds and the initially expected log-odds. Therefore, modifying the prior influences the contribution of new observed moving objects to the updated prediction.

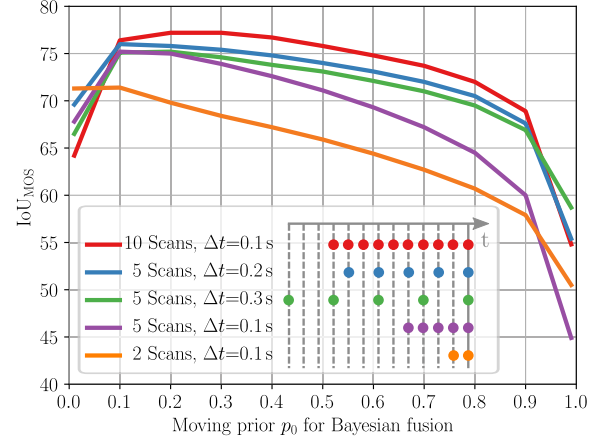


Fig. 4. Comparison of IoUMOS on the SemanticKITTI [3] validation set using different moving object priors for the binary Bayes filter. The colors indicate variants of our approach using different time horizons and resolutions. The colored dots on the timeline visualize which past scans we input to the model for prediction at time t .

Fig. 4 shows the IoUMOS on the SemanticKITTI validation set for different priors. With a small prior (e.g. $p_0 = 0.01$), we fuse predicted moving objects more aggressively leading to more true positives, but also an increased number of false positives since inconsistent predictions are not filtered out. A large prior (e.g. $p_0 = 0.99$) results in a conservative fusion where objects are only predicted to be moving if all predictions agree. We found that a moving object prior between 0.1 and 0.3 works best for the SemanticKITTI validation sequence. We experienced that for a lot of slowly moving objects in the scene, setting a lower prior helps to keep them in the final prediction even if they have not been predicted moving from all available instances in time. We achieve the best result with a model using $N = 10$ input and output frames and a Bayesian prior of $p_0 = 0.25$, which is the setup for the experiments presented in Section IV-B and Section IV-C.

In general, a combination of processing more scans and fusing multiple predictions with the receding horizon strategy works best for achieving a larger time horizon resulting in better moving object segmentation. Our approach also works with fewer scans but with a larger temporal resolution, which reduces the computational effort. The models using $N = 5$ input scans with a larger temporal resolution of $\Delta t = 0.2$ s and $\Delta t = 0.3$ s between scans outperform the model with the same number of processed scans but a smaller resolution of $\Delta t = 0.1$ s, which is the actual sensor frame rate. This shows that the extended time horizon achieved with a larger temporal resolution leads to better segmented slowly moving objects since their motion is more visible in the sequence.

E. Ablation Study

To further support our third claim and show the effectiveness of individual proposed components of our approach, we train different variants of our network and evaluate their performance on the validation set. We train all models for up to 60 epochs and report the best IoUMOS on the validation set

TABLE III
ABLATION STUDY ON DIFFERENT VARIANTS OF OUR APPROACH WITH AND WITHOUT THE PROPOSED RECEDING HORIZON STRATEGY AND BAYESIAN FUSION (BF) USING A PRIOR $p_0 = 0.25$. WE DENOTE THE TEMPORAL RESOLUTION BETWEEN SCANS BY Δt . BEST RESULTS IN BOLD

	# Inputs	# Outputs	Poses?	Δt	IoU _{MOS} [%]	
					w/o BF	w/ BF
[A]	5	5	✓	0.1 s	69.1	74.5
[B]	5	5	✓	0.2 s	71.8	75.6
[C]	5	5	✓	0.3 s	71.6	74.9
[D]	5	5	✗	0.1 s	35.6	39.9
[E]	5	1	✓	0.1 s	66.5	-
[F]	2	2	✓	0.1 s	64.9	69.0
[G]	10	10	✓	0.1 s	74.3	77.2

during training, see Table III. For all methods, we compare two prediction strategies: First, a non-overlapping strategy that divides the input sequence into sub-sequences and predicts each sub-sequence independently, see the upper part in Fig. 3. Second, our receding horizon strategy proposed in Section III-C, which generates multiple predictions for the same scan and fuses them in a binary Bayes filter (again using a prior of $p_0 = 0.25$). We visualize this combination in the lower part of Fig. 3.

In general, we see an improvement of up to 5.4 percentage points of IoU_{MOS} for all models using the proposed receding horizon strategy. More precisely, using the binary Bayes filter with model [A] reduces the number of false negatives by 8.2% and the number of false positives by 18.9%. This indicates that the proposed approach successfully integrates more observations into the estimation and is more robust to false predictions due to occlusions or noisy measurements. If we compare the performance of model [A] using $N = 5$ scans which are $\Delta t = 0.1$ s apart to the networks trained with larger temporal resolutions of $\Delta t = 0.2$ s [B] and $\Delta t = 0.3$ s [C], we again see that the results can be further improved by considering a larger time horizon, see also Section IV-D. If the point clouds are not transformed into a common viewpoint, the method [D] is still able to infer moving objects but at a reduced performance of IoU_{MOS} = 39.9% with Bayesian fusion. This is because the network needs to infer both the ego-motion of the sensor as well as the relative motion of the objects. When only training to predict a single output scan [E], the result is worse and fusing more predictions is not possible since no additional predictions are available. Next, one can see that our method can also achieve moving object segmentation with two scans only [F] but the performance is worse. The best performing model [G] takes $N = 10$ input scans with a temporal resolution of $\Delta t = 0.1$ s and fuses the predictions resulting in an IoU_{MOS} of 77.2%. The results show that we achieve a better moving object segmentation by increasing the time horizon with a combination of processing more scans, increasing the temporal resolution, and using the proposed receding horizon strategy with a binary Bayes filter.

F. Qualitative Results

Finally, we illustrate that our method predicts actually moving objects in 3D space without the need for geometric

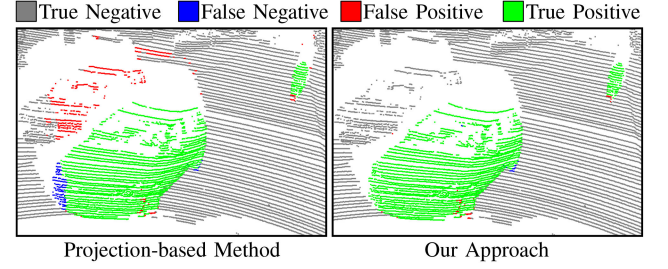


Fig. 5. Qualitative comparison of segmentation accuracy. *Left*: Prediction by range image-based LMNet [5] after kNN post-processing. *Right*: Our sparse voxel-based approach without further post-processing. Best viewed in color.

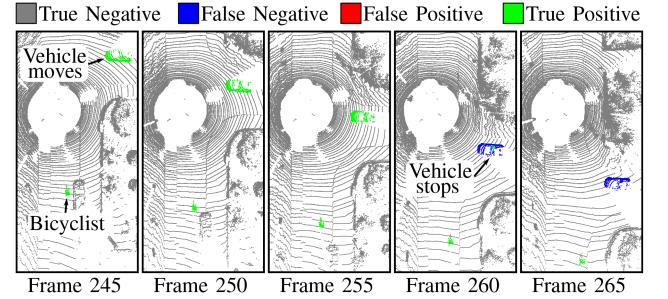


Fig. 6. Change in moving object segmentation if an object stops moving. Best viewed in color.

post-processing like clustering. We use the model from Section IV-B trained on $N = 10$ scans with a temporal resolution of $\Delta t = 0.1$ s. In Fig. 5, we show the segmentation of scan 1638 from the SemanticKITTI [3] validation sequence 08. We compare the range image-based method LMNet [5] to our sparse voxel-based approach. One can see that despite the geometric-based kNN post-processing, the baseline still shows artifacts and bleeding labels behind the moving car illustrated as red-colored false positives whereas our method directly predicts in the 3D space without boundary effects.

Next, Fig. 6 shows how the prediction changes for a scene in the validation set in which a vehicle stops moving. Since our method does not use any semantic understanding of objects, it only reasons about how the points move in space for the given time horizon. Since the vehicle stops moving to yield at the intersection, our method's prediction changes from moving to static. The bicyclist in the back is successfully classified as moving. Note that this results in a false negative indicated in blue since the ground truth SemanticKITTI labels consider if an object has moved throughout the data collection and not based on recent movement.

G. Runtime

With our unoptimized Python implementation, the network requires on average 0.078 s for predicting moving objects in 10 scans and 0.047 s for 5 scans both using an NVIDIA RTX A5000. Our binary Bayes filter only adds a small overhead of 0.008 s on average for fusing 10 predictions and 0.004 s for fusing 5 predictions.

V. CONCLUSION

In this paper, we present a novel approach to segment moving objects in 3D LiDAR data. Our method jointly predicts moving objects for all scans in the input sequence and operates using a receding horizon strategy. We report improved performance on the SemanticKITTI moving object segmentation benchmark and show that the approach generalizes well on unseen data. Our proposed receding horizon strategy in combination with a binary Bayes filter allows us to extend the time horizon used for segmenting moving objects and to increase the robustness to false positive and false negative predictions. Currently, we estimate odometry and moving object segmentation separately which can be jointly optimized in future work.

REFERENCES

- [1] M. Arora, L. Wiesmann, X. Chen, and C. Stachniss, "Mapping the static parts of dynamic scenes from 3D LiDAR point clouds exploiting ground segmentation," in *Proc. Euro. Conf. Mobile Robot.*, 2021, pp. 1–6.
- [2] S. A. Baur, D. J. Emmerichs, F. Moosmann, P. Pinggera, B. Ommer, and A. Geiger, "SLIM: Self-supervised LiDAR scene flow and motion segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 13106–13116.
- [3] J. Behley *et al.*, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9296–9306.
- [4] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robot.: Sci. Syst.*, 2018.
- [5] X. Chen *et al.*, "Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6529–6536, Oct. 2021.
- [6] X. Chen *et al.*, "Automatic labeling to generate training data for online LiDAR-based moving object segmentation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 6107–6114, Jul. 2022.
- [7] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4530–4537.
- [8] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3070–3079.
- [9] T. Cortinhal, G. Tzelepis, and E. Aksoy, "SalsaNext: Fast semantic segmentation of LiDAR point clouds for autonomous driving," in *Proc. Adv. Vis. Comput. 15th Int. Symp.*, San Diego, CA, USA, Oct. 2020, pp. 207–222.
- [10] A. Dewan and W. Burgard, "DeepTemporalSeg: Temporally consistent semantic segmentation of 3D LiDAR scans," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2624–2630.
- [11] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.
- [12] H. W. F. Duerr, M. Pfaller, and J. Beyerer, "LiDAR-based recurrent 3D semantic segmentation with temporal memory alignment," in *Proc. Int. Conf. 3D Vis.*, 2020, pp. 781–790.
- [13] H. Fan, X. Yu, Y. Ding, Y. Yang, and M. Kankanhalli, "PSTNet: Point spatio-temporal convolution on point cloud sequences," in *Proc. Int. Conf. Learn. Representations*, 2021.
- [14] H. Fan and Y. Yang, "PointRNN: Point recurrent neural network for moving point cloud processing," 2019, *arXiv:1910.08287*.
- [15] Z. Gojcic, O. Litany, A. Wieser, L. J. Guibas, and T. Birdal, "Weakly supervised learning of rigid 3D scene flow," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5688–5699.
- [16] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, pp. 189–206, 2013.
- [17] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10758–10765.
- [18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
- [19] A. Laddha, S. Gautam, G. P. Meyer, and C. Vallespi-Gonzalez, "RV-FuseNet: Range view based fusion of time-series LiDAR data for joint 3D object detection and motion forecasting," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 7060–7066.
- [20] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 529–537.
- [21] X. Liu, M. Yan, and J. Bohg, "MeteorNet: Deep learning on dynamic 3D point cloud sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9245–9254.
- [22] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-Net: Towards learning based LiDAR localization for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 6382–6391.
- [23] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3569–3577.
- [24] B. Mersch, X. Chen, J. Behley, and C. Stachniss, "Self-supervised point cloud prediction using 3D spatio-temporal convolutional networks," in *Proc. Conf. Robot Learn.*, 2022, pp. 1444–1454.
- [25] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet : Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4213–4220.
- [26] S. Mohapatra *et al.*, "LiMoSeg: Real-time bird's eye view based LiDAR motion segmentation," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, 2022, pp. 828–835.
- [27] L. Nunes, R. Marcuzzi, X. Chen, J. Behley, and C. Stachniss, "SegContrast: 3D point cloud feature representation learning through self-supervised segment discrimination," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 2116–2123, Apr. 2022.
- [28] F. Pomerleau, P. Krüsiand, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3D map maintenance in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 3712–3719.
- [29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, 2015, pp. 234–241.
- [30] P. Ruchti and W. Burgard, "Mapping with dynamic-object probabilities calculated from single 3D range scans," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 6331–6336.
- [31] T. Shan *et al.*, "Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.
- [32] H. Shi, G. Lin, H. Wang, T. Y. Hung, and Z. Wang, "SpSequenceNet: Semantic segmentation network on 4D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4573–4582.
- [33] C. Stachniss and W. Burgard, "Mobile robot mapping and localization in non-static environments," in *Proc. Nat. Conf. Artif. Intell.*, 2005, pp. 1324–1329.
- [34] H. Thomas, B. Agro, M. Gridseth, J. Zhang, and T. D. Barfoot, "Self-supervised learning of lidar segmentation for autonomous indoor navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 14047–14053.
- [35] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6410–6419.
- [36] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.
- [37] I. Tishchenko, S. Lombardi, M. R. Oswald, and M. Pollefeys, "Self-supervised learning of non-rigid residual flow and ego-motion," in *Proc. Int. Conf. 3D Vis.*, 2020, pp. 150–159.
- [38] M. Toyungyernsub, M. Itkina, R. Senanayake, and M. J. Kochenderfer, "Double-prong ConvLSTM for spatiotemporal occupancy prediction in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 13931–13937.
- [39] D. Z. Wang, I. Posner, and P. Newman, "What could move? Finding cars, pedestrians and bicyclists in 3D laser data," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 4038–4044.
- [40] P. Wu, S. Chen, and D. N. Metaxas, "MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11382–11392.